Differentiating through the Fréchet Mean

Aaron Lou [*1] Isay Katsman [*1] Qingxuan Jiang [*1] Serge Belongie [1] Ser-Nam Lim [2] Christopher De Sa [1]

Abstract

Recent advances in deep representation learning on Riemannian manifolds extend classical deep learning operations to better capture the geometry of the manifold. One possible extension is the Fréchet mean, the generalization of the Euclidean mean; however, it has been difficult to apply because it lacks a closed form with an easily computable derivative. In this paper, we show how to differentiate through the Fréchet mean for arbitrary Riemannian manifolds. Then, focusing on hyperbolic space, we derive explicit gradient expressions and a fast, accurate, and hyperparameter-free Fréchet mean solver. This fully integrates the Fréchet mean into the hyperbolic neural network pipeline. To demonstrate this integration, we present two case studies. First, we apply our Fréchet mean to the existing Hyperbolic Graph Convolutional Network, replacing its projected aggregation to obtain state-of-the-art results on datasets with high hyperbolicity. Second, to demonstrate the Fréchet mean's capacity to generalize Euclidean neural network operations, we develop a hyperbolic batch normalization method that gives an improvement parallel to the one observed in the Euclidean setting[1].

1. Introduction

Recent advancements in geometric representation learning have utilized hyperbolic space for tree embedding tasks (Nickel & Kiela, 2017; 2018; Yu & De Sa, 2019). This is due to the natural non-Euclidean structure of hyperbolic

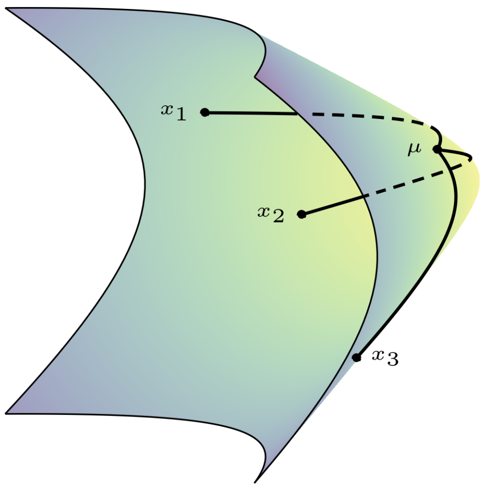

Figure 1. Depicted above is the Fréchet mean, μ, of three points, x_1, x_2, x_3 in the Lorentz model of hyperbolic space. As one can see, the Fréchet mean conforms with the geometry of the hyperboloid and is vastly different from the standard Euclidean mean.

space, in which distances grow exponentially as one moves away from the origin. Such a geometry is naturally equipped to embed trees, since if we embed the root of the tree near the origin and layers at successive radii, the geometry of hyperbolic space admits a natural hierarchical structure. More recent work has focused specifically on developing neural networks that exploit the structure of hyperbolic space (Ganea et al., 2018; Tifrea et al., 2019; Chami et al., 2019; Liu et al., 2019).

A useful structure that has thus far not been generalized to non-Euclidean neural networks is that of the Euclidean mean. The (trivially differentiable) Euclidean mean is necessary to perform aggregation operations such as attention (Vaswani et al., 2017), and stability-enhancing operations such as batch normalization (Ioffe & Szegedy, 2015), in the context of Euclidean neural networks. The Euclidean mean extends naturally to the Fréchet mean in non-Euclidean geometries (Fréchet, 1948). However, unlike the Euclidean mean, the Fréchet mean does not have a closed-form solution, and its computation involves an argmin operation that cannot be easily differentiated. This makes the important operations we are able to perform in Euclidean space hard to

[*]Equal contribution [1]Department of Computer Science, Cornell University, NY, Ithaca, USA [2]Facebook AI, NY, New York, USA. Correspondence to: Aaron Lou <al968@cornell.edu>, Isay Katsman <isk22@cornell.edu>, Qingxuan Jiang <qj46@cornell.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

[1]Our PyTorch implementation of the differentiable Fréchet mean can be found at https://github.com/CUVL/Differentiable-Frechet-Mean.

generalize to their non-Euclidean counterparts. In this paper, we extend the methods in Gould et al. (2016) to differentiate through the Fréchet mean, and we apply our methods to downstream tasks. Concretely, our paper's contributions are that:

- We derive closed-form gradient expressions for the Fréchet mean on Riemannian manifolds.
- For the case of hyperbolic space, we present a novel algorithm for quickly computing the Fréchet mean and a closed-form expression for its derivative.
- We use our Fréchet mean computation in place of the neighborhood aggregation step in Hyperbolic Graph Convolution Networks (Chami et al., 2019) and achieve state-of-the-art results on graph datasets with high hyperbolicity.
- We introduce a fully differentiable Riemannian batch normalization method which mimics the procedure and benefit of standard Euclidean batch normalization.

2. Related Work

Uses of Hyperbolic Space in Machine Learning. The usage of hyperbolic embeddings first appeared in Kleinberg (2007), in which the author uses them in a greedy embedding algorithm. Later analyses by Sarkar (2011) and Sala et al. (2018) demonstrate the empirical and theoretical improvement of this approach. However, only recently in Nickel & Kiela (2017; 2018) was this method extended to machine learning. Since then, models such as those in Ganea et al. (2018); Chami et al. (2019); Liu et al. (2019) have leveraged hyperbolic space operations to obtain better embeddings using a hyperbolic version of deep neural networks.

Fréchet Mean. The Fréchet mean (Fréchet, 1948), as the generalization of the classical Euclidean mean, offers a plethora of applications in downstream tasks. As a mathematical construct, the Fréchet mean has been thoroughly studied in texts such as Karcher (1977); Charlier (2013); Bacák (2014).

However, the Fréchet mean is an operation not without complications; the general formulation requires an argmin operation and offers no closed-form solution. As a result, both computation and differentiation are problematic, although previous works have attempted to resolve such difficulties.

To address computation, Gu et al. (2019) show that a Riemannian gradient descent algorithm recovers the Fréchet mean in linear time for products of Riemannian model spaces. However, without a tuned learning rate, it is too hard to ensure performance. Brooks et al. (2019) instead use the Karcher Flow Algorithm (Karcher, 1977); although this method is manifold-agnostic, it is slow in practice. We address such existing issues in the case of hyperbolic space by providing a fast, hyperparameter-free algorithm for computing the Fréchet mean.

Some works have addressed the differentiation issue by circumventing it, instead relying on pseudo-Fréchet means. In Law et al. (2019), the authors utilize a novel squared Lorentzian distance (as opposed to the canonical distance for hyperbolic space) to derive explicit formulas for the Fréchet mean in pseudo-hyperbolic space. In Chami et al. (2019), the authors use an aggregation method in the tangent space as a substitute. Our work, to the best of our knowledge, is the first to provide explicit derivative expressions for the Fréchet mean on Riemannian manifolds.

Differentiating through the argmin. Theoretical foundations of differentiating through the argmin operator have been provided in Gould et al. (2016). Similar methods have subsequently been used to develop differentiable optimization layers in neural networks (Amos & Kolter, 2017; Agrawal et al., 2019).

Given that the Fréchet mean is an argmin operation, one might consider utilizing the above differentiation techniques. However, a naïve application fails, as the Fréchet mean's argmin domain is a manifold, and Gould et al. (2016) deals specifically with Euclidean space. Our paper extends this theory to the case of general Riemannian manifolds, thereby allowing the computation of derivatives for more general argmin problems, and, in particular, for computing the derivative of the Fréchet mean in hyperbolic space.

3. Background

In this section, we establish relevant definitions and formulas of Riemannian manifolds and hyperbolic spaces. We also briefly introduce neural network layers in hyperbolic space.

3.1. Riemannian Geometry Background

Here we provide some of the useful definitions from Riemannian geometry. For a more in-depth introduction, we refer the interested reader to our Appendix C or texts such as Lee (2003) and Lee (1997).

Manifold and tangent space: An n-dimensional manifold $\mathcal{M}$ is a topological space that is locally homeomorphic to $\mathbb{R}^n$. The tangent space $T_x\mathcal{M}$ at x is defined as the vector space of all tangent vectors at x and is isomorphic to $\mathbb{R}^n$. We assume our manifolds are smooth, i.e. the maps are diffeomorphic. The manifold admits local coordinates $(x_1, \dots, x_n)$ which form a basis $(dx_1, \dots, dx_n)$ for the tangent space.

Riemannian metric and Riemannian manifold: For a manifold $\mathcal{M}$, a Riemannian metric $\rho = (\rho_x)_{x \in \mathcal{M}}$ is a

Table 1. Summary of operations in the Poincaré ball model and the hyperboloid model ($K < 0$)

	Poincaré Ball	**Hyperboloid**
Manifold	$\mathbb{D}_K^n = \{x \in \mathbb{R}^n : \langle x, x\rangle_2 < -\frac{1}{K}\}$	$\mathbb{H}_K^n = \{x \in \mathbb{R}^{n+1} : \langle x, x\rangle_{\mathcal{L}} = \frac{1}{K}\}$
Metric	$g_x^{\mathbb{D}_K} = (\lambda_x^K)^2 g^{\mathbb{E}}$ where $\lambda_x^K = \frac{2}{1+K\|x\|_2^2}$ and $g^{\mathbb{E}} = I$	$g_x^{\mathbb{H}_K} = \eta$, where η is I except $\eta_{0,0} = -1$
Distance	$d_{\mathbb{D}}^K(x, y) = \frac{1}{\sqrt{\lvert K\rvert}} \cosh^{-1}\left(1 - \frac{2K\|x-y\|_2^2}{(1+K\|x\|_2^2)(1+K\|y\|_2^2)}\right)$	$d_{\mathbb{H}}^K(x, y) = \frac{1}{\sqrt{\lvert K\rvert}} \cosh^{-1}(K\langle x, y\rangle_{\mathcal{L}})$
Exp map	$\exp_x^K(v) = x \oplus_K \left(\tanh\left(\sqrt{\lvert K\rvert}\frac{\lambda_x^K\|v\|_2}{2}\right)\frac{v}{\sqrt{\lvert K\rvert}\|v\|_2}\right)$	$\exp_x^K(v) = \cosh(\sqrt{\lvert K\rvert}\lvert\lvert v\rvert\rvert_{\mathcal{L}})x + v\frac{\sinh(\sqrt{\lvert K\rvert}\lvert\lvert v\rvert\rvert_{\mathcal{L}})}{\sqrt{\lvert K\rvert}\lvert\lvert v\rvert\rvert_{\mathcal{L}}}$
Log map	$\log_x^K(y) = \frac{2}{\sqrt{\lvert K\rvert}\lambda_x^K}\tanh^{-1}(\sqrt{\lvert K\rvert}\|-x \oplus_K y\|_2)\frac{-x\oplus_K y}{\|-x\oplus_K y\|_2}$	$\log_x^K(y) = \frac{\cosh^{-1}(K\langle x,y\rangle_{\mathcal{L}})}{\sinh(\cosh^{-1}(K\langle x,y\rangle_{\mathcal{L}}))}(y - K\langle x, y\rangle_{\mathcal{L}}x)$
Transport	$PT_{x\to y}^K(v) = \frac{\lambda_x^K}{\lambda_y^K}\operatorname{gyr}[y, -x]v$	$PT_{x\to y}^K(v) = v - \frac{K\langle y,v\rangle_{\mathcal{L}}}{1+K\langle x,y\rangle_{\mathcal{L}}}(x + y)$

Table 2. Summary of hyperbolic counterparts of Euclidean operations in neural networks

Operation	**Formula**
Matrix-vector multiplication	$A \otimes^K x = \exp_0^K(A\log_0^K(x))$
Bias translation	$x \oplus^K b = \exp_x(PT_{0\to x}^K(b))$
Activation function	$\sigma^{K_1,K_2}(x) = \exp_0^{K_1}(\sigma(\log_0^{K_2}(x)))$

smooth collection of inner products $\rho_x : T_x\mathcal{M}\times T_x\mathcal{M} \to \mathbb{R}$ on the tangent space of every $x \in \mathcal{M}$. The resulting pair $(\mathcal{M}, \rho)$ is called a Riemannian manifold. Note that ρ induces a norm in each tangent space $T_x\mathcal{M}$, given by $\|\vec{v}\|_\rho = \sqrt{\rho_x(\vec{v}, \vec{v})}$ for any $\vec{v} \in T_x\mathcal{M}$. We oftentimes associate ρ to its matrix form (ρ_{ij}) where $\rho_{ij} = \rho(dx_i, dx_j)$ when given local coordinates.

Geodesics and induced distance function: For a curve $\gamma : [a, b] \to \mathcal{M}$, we define the length of γ to be $L(\gamma) = \int_a^b \|\gamma'(t)\|_\rho\, dt$. For $x, y \in \mathcal{M}$, the distance $d(x, y) = \inf L(\gamma)$ where γ is any curve such that $\gamma(a) = x, \gamma(b) = y$. A geodesic γ_{xy} from x to y, in our context, should be thought of as a curve that minimizes this length[2].

Exponential and logarithmic map: For each point $x \in \mathcal{M}$ and vector $\vec{v} \in T_x\mathcal{M}$, there exists a unique geodesic $\gamma : [0, 1] \to \mathcal{M}$ where $\gamma(0) = x, \gamma'(0) = \vec{v}$. The exponential map $\exp_x : T_x\mathcal{M} \to \mathcal{M}$ is defined as $\exp_x(\vec{v}) = \gamma(1)$. Note that this is an isometry, i.e. $\|\vec{v}\|_\rho = d(x, \exp_x(\vec{v}))$. The logarithmic map $\log_x : \mathcal{M} \to T_x\mathcal{M}$ is defined as the inverse of $\exp_x$, although this can only be defined locally[3].

Parallel transport: For $x, y \in \mathcal{M}$, the parallel transport $PT_{x\to y} : T_x\mathcal{M} \to T_y\mathcal{M}$ defines a way of transporting the local geometry from x to y along the unique geodesic that preserves the metric tensors.

3.2. Hyperbolic Geometry Background

We now examine hyperbolic space, which has constant curvature $K < 0$, and provide concrete formulas for computation. The two equivalent models of hyperbolic space frequently used are the Poincaré ball model and the hyperboloid model. We denote $\mathbb{D}_K^n$ and $\mathbb{H}_K^n$ as the n-dimensional Poincaré ball and hyperboloid models with curvature $K < 0$, respectively.

3.2.1. Basic Operations

Inner products: We define $\langle x, y\rangle_2$ to be the standard Euclidean inner product and $\langle x, y\rangle_{\mathcal{L}}$ to be the Lorentzian inner product $-x_0y_0 + x_1y_1 + \cdots + x_ny_n$.

Gyrovector operations: For $x, y \in \mathbb{D}_K^n$, the Möbius addition (Ungar, 2008) is

$$x \oplus_K y = \frac{(1 - 2K\langle x, y\rangle_2 - K\|y\|_2^2)x + (1 + K\|x\|_2^2)y}{1 - 2K\langle x, y\rangle_2 + K^2\|x\|_2^2\|y\|_2^2} \tag{1}$$

This induces Möbius subtraction $\ominus_K$ which is defined as $x \ominus_K y = x \oplus_K -y$. In the theory of gyrogroups, the notion of the gyration operator (Ungar, 2008) is given by

$$\operatorname{gyr}[x, y]v = \ominus_K(x \oplus_K y) \oplus_K (x \oplus_K (y \oplus_K v)) \tag{2}$$

Riemannian operations on hyperbolic space: We summarize computations for the Poincaré ball model and the

[2]Formally, geodesics are curves with 0 acceleration w.r.t. the Levi-Civita connection. There are geodesics which are not minimizing curves, such as the larger arc between two points on a great circle of a sphere; hence this clarification is important.

[3]Problems in definition arise in the case of conjugate points (Lee, 1997). However, exp is a local diffeomorphism by the inverse function theorem.

hyperboloid model in Table 1.

3.3. Hyperbolic Neural Networks

Introduced in Ganea et al. (2018), hyperbolic neural networks provide a natural generalization of standard neural networks.

Hyperbolic linear layer: Recall that a Euclidean linear layer is defined as $f : \mathbb{R}^m \to \mathbb{R}^n$, $f = \sigma(Ax + b)$ where $A \in \mathbb{R}^{n \times m}$, $x \in \mathbb{R}^m$, $b \in \mathbb{R}^n$ and σ is some activation function.

With analogy to Euclidean layers, a hyperbolic linear layer $g : \mathbb{H}^m \to \mathbb{H}^n$ is defined by $g = \sigma^{K,K}(A \otimes^K x \oplus^K b)$, where $A \in \mathbb{R}^{n \times m}$, $x \in \mathbb{H}^m$, $b \in \mathbb{H}^n$, and we replace the operations by hyperbolic counterparts outlined in Table 2.

Hyperbolic neural networks are defined as compositions of these layers, similar to how conventional neural networks are defined as compositions of Euclidean layers.

4. A Differentiable Fréchet Mean Operation for General Riemannian Manifolds

In this section, we provide a few theorems that summarize our method of differentiating through the Fréchet mean.

4.1. Background on the Fréchet Mean

Fréchet mean and variance: On a Riemannian manifold $(\mathcal{M}, \rho)$, the Fréchet mean $\mu_{fr} \in \mathcal{M}$ and Fréchet variance $\sigma^2_{fr} \in \mathbb{R}$ of a set of points $\mathcal{B} = \{x^{(1)}, \cdots, x^{(t)}\}$ with each $x^{(l)} \in \mathcal{M}$ are defined as the solution and optimal values of the following optimization problem (Bacák, 2014):

$$\mu_{fr} = \operatorname*{arg\,min}_{\mu \in \mathcal{M}} \frac{1}{t} \sum_{l=1}^{t} d(x^{(l)}, \mu)^2 \tag{3}$$

$$\sigma^2_{fr} = \min_{\mu \in \mathcal{M}} \frac{1}{t} \sum_{l=1}^{t} d(x^{(l)}, \mu)^2 \tag{4}$$

In Appendix A, we provide proofs to illustrate that this definition is a natural generalization of Euclidean mean and variance.

The Fréchet mean can be further generalized with an arbitrary re-weighting. In particular, for positive weights $\{w_l\}_{l \in [t]}$, we can define the weighted Fréchet mean as:

$$\mu_{fr} = \operatorname*{arg\,min}_{\mu \in \mathcal{M}} \sum_{l=1}^{t} w_l \cdot d(x^{(l)}, \mu)^2 \tag{5}$$

This generalizes the weighted Euclidean mean to Riemannian manifolds.

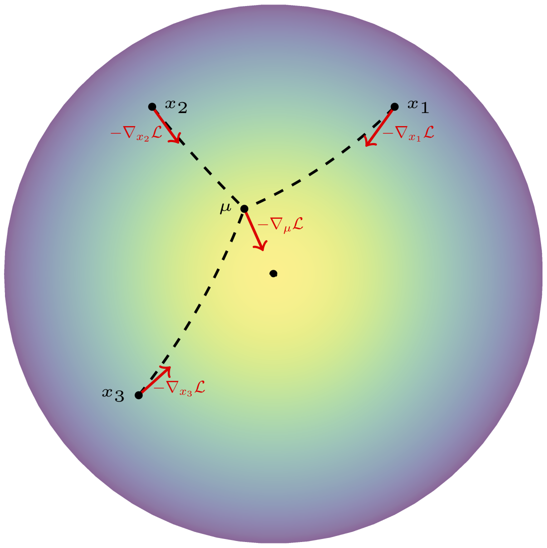
x_2
x_1
$-\nabla_{x_2}\mathcal{L}$
$-\nabla_{x_1}\mathcal{L}$
μ
$-\nabla_{\mu}\mathcal{L}$
x_3
$-\nabla_{x_3}\mathcal{L}$

Figure 2. Depicted above is the Fréchet mean, μ, of three points, x_1, x_2, x_3 in the Poincaré ball model of hyperbolic space, $\mathbb{D}^2_{-1}$, as well as the negative gradients (shown in red) with respect to the loss function $\mathcal{L} = \|\mu\|^2$.

4.2. Differentiating Through the Fréchet Mean

All known methods for computing the Fréchet mean rely on some sort of iterative solver (Gu et al., 2019). While backpropagating through such a solver is possible, it is computationally inefficient and suffers from numerical instabilities akin to those found in RNNs (Pascanu et al., 2013). To circumvent these issues, recent works compute gradients at the solved value instead of differentiating directly, allowing for full neural network integration (Chen et al., 2018; Pogančić et al., 2020). However, to the best of our knowledge, no paper has investigated backpropagation on a manifold-based convex optimization solver. Hence, in this section, we construct the gradient, relying on the fact that the Fréchet mean is an argmin operation.

4.2.1. Differentiating through the argmin Operation

Motivated by previous works on differentiating argmin problems (Gould et al., 2016), we propose a generalization which allows us to differentiate the argmin operation on the manifold. The full theory is presented in Appendix D.

4.2.2. Construction of the Fréchet mean derivative

Since the Fréchet mean is an argmin operation, we can apply the theorems in Appendix D to obtain gradients with respect to the input points. This operation (as well the resulting gradients) are visualized in Figure 2.

For the following theorems, we denote $\widetilde{\nabla}$ as the total derivative (or Jacobian) for notational convenience.

Theorem 4.1. *Let $\mathcal{M}$ be an n-dimensional Riemannian manifold, and let $\{x\} = (x^{(1)}, \dots, x^{(t)}) \in (\mathcal{M})^t$ be a set of data points with weights $w_1, \dots, w_t \in \mathbb{R}^+$. Let $f : (\mathcal{M})^t \times \mathcal{M} \to \mathcal{M}$ be given by $f(\{x\}, y) = \sum_{l=1}^{t} w_l \cdot d(x^{(l)}, y)^2$ and $\overline{x} = \mu_{fr}(\{x\}) = \arg\min_{y \in \mathcal{M}} f(\{x\}, y)$ be the Fréchet mean. Then with respect to local coordinates we have*

$$\widetilde{\nabla}_{x^{(i)}} \mu_{fr}(\{x\}) = -f_{YY}(\{x\}, \overline{x})^{-1} f_{X^{(i)}Y}(\{x\}, \overline{x}) \quad (6)$$

where the functions $f_{X^{(i)}Y}(\{x\}, y) = \widetilde{\nabla}_{x^{(i)}} \nabla_y f(\{x\}, y)$ and $f_{YY}(\{x\}, y) = \nabla^2_{yy} f(\{x\}, \overline{x})$ are defined in terms of local coordinates.

Proof. This is a special case of Theorem D.1 in the appendix. This is because the Fréchet objective function f is a twice differentiable real-valued function for specific $x^{(i)}$ and y (under our geodesic formulation); thus we obtain the desired formulation. The full explanation can be found in Remark D. □

While the above theorem gives a nice theoretical framework with minimal assumptions, it is in practice too unwieldy to apply. In particular, the requirement of local coordinates renders most computations difficult. We now present a version of the above theorem which assumes that the manifold is embedded in Euclidean space[4].

Theorem 4.2. *Assume the conditions and values in Theorem 4.1. Furthermore, assume $\mathcal{M}$ is embedded (as a Riemannian manifold) in $\mathbb{R}^m$ with $m \geq \dim \mathcal{M}$, then we can write*

$$\widetilde{\nabla}_{x^{(i)}} \mu_{fr}(\{x\}) = -f^p_{YY}(\{x\}, \overline{x})^{-1} f^p_{X^{(i)}Y}(\{x\}, \overline{x}) \quad (7)$$

where $f^p_{YY}(\{x\}, y) = \widetilde{\nabla}_y (\text{proj}_{T_{\overline{x}}\mathcal{M}} \circ \nabla_y f)(\{x\}, y)$, $f^p_{X^{(i)}Y}(\{x\}, y) = \widetilde{\nabla}_{x^{(i)}} (\text{proj}_{T_{\overline{x}}\mathcal{M}} \circ \nabla_y f)(\{x\}, y)$, and $\text{proj}_{T_{\overline{x}}\mathcal{M}} : \mathbb{R}^m \to T_{\overline{x}}\mathcal{M} \cong \mathbb{R}^n$ is the linear subspace projection operator.

Proof. Similar to the relationship between Theorem 4.1 and Theorem D.1, this is a special case of Theorem D.3 in the appendix. □

5. Hyperbolic Fréchet Mean

Although we have provided a formulation for differentiating through the Fréchet mean on general Riemannian manifolds, to properly integrate it in the hyperbolic setting we need to address two major difficulties:

1. The lack of a fast forward computation.

2. The lack of an explicit derivation of a backpropagation formula.

Resolving these difficulties will allow us to define a Fréchet mean neural network layer for geometric, and specifically hyperbolic, machine learning tasks.

5.1. Forward Computation of the Hyperbolic Fréchet Mean

Previous forward computations fall into one of two categories: (1) fast, inaccurate computations which aim to approximate the true mean with a pseudo-Fréchet mean, or (2) slow, exact computations. In this section we focus on outperforming methods in the latter category, since we strive to compute the exact Fréchet mean (pseudo-means warp geometry).

5.1.1. Former Attempts at Computing the Fréchet Mean

The two existing algorithms for Fréchet mean computation are (1) Riemannian gradient-based optimization (Gu et al., 2019) and (2) iterative averaging (Karcher, 1977). However, in practice both algorithms are slow to converge even for simple synthetic examples of points in hyperbolic space. To overcome this difficulty, which can cripple neural networks, we propose the following algorithm that is much faster in practice.

Algorithm 1 Poincaré model Fréchet mean algorithm

```
**Inputs**: $x^{(1)}, \cdots, x^{(t)} \in \mathbb{D}^n_K \subseteq \mathbb{R}^{n+1}$ and weights $w_1, \dots, w_t \in \mathbb{R}^+$.
**Algorithm**:
$y_0 = x^{(1)}$
Define $g(y) = \frac{2\,\text{arccosh}(1+2y)}{\sqrt{y^2+y}}$
for $k = 0, 1, \cdots, T$:
    for $l = 1, 2, \cdots, t$:
        $\alpha_l = w_l g\left(\frac{\|x^{(l)} - y_k\|^2}{(1+K\|x^{(l)}\|^2)(1+K\|y_k\|^2)}\right) \frac{1}{1+K\|x^{(l)}\|^2}$
    $a = \sum_{l=1}^{t} \alpha_l$ , $b = \sum_{l=1}^{t} \alpha_l x^{(l)}$ , $c = \sum_{l=1}^{t} \alpha_l \|x^{(l)}\|^2$
    $y_{k+1} = \left(\frac{(a-cK) - \sqrt{(a-cK)^2 + 4K \cdot \|b\|^2}}{2|K| \cdot \|b\|^2}\right) b$
return $y_T$
```

[4]We also present a more general way to take the derivative that drops this restriction via an exponential map-based parameterization in Appendix D.

5.1.2. Algorithm for Fréchet Mean Computation via First-order Bound

The core idea of our algorithm relies on the fact that the square of distance metric is a concave function for both the Poincaré ball and hyperboloid model. Intuitively, we select an initial "guess" and use a first-order bound to minimize the Fréchet mean objective. The concrete algorithm for the Poincaré ball model is given as Algorithm 1 above. Note that the algorithm is entirely hyperparameter-free and does not require setting a step-size. Additionally we introduce three different initializations:

1. Setting $y_0 = x^{(1)}$.
2. Setting $y_0 = x^{(\arg\max_i w_i)}$.
3. Setting y_0 to be the output of the first step of the Karcher flow algorithm (Karcher, 1977).

We tried these initializations for our test tasks (in which weights were equal, tasks described in Section 6), and found little difference between them in terms of performance. Even for toy tasks with varying weights, these three methods produced nearly the same results. However, we give them here for completeness.

Moreover, we can prove that the algorithm is guaranteed to converge.

Theorem 5.1. *Let $x^{(1)}, \cdots, x^{(t)} \in \mathbb{D}_K^n$ be t points[5] in the Poincaré ball, $w_1, \ldots, w_t \in \mathbb{R}^+$ be their weights, and let their weighted Fréchet mean be the solution to the following optimization problem.*

$$\mu_{fr} = \arg\min_{y \in \mathbb{D}_K^n} f(y) \tag{8}$$

$$\text{where } f(y) = \sum_{l=1}^{t} w_l \cdot d_{\mathbb{D}_K^n}(x^{(l)}, y)^2$$

$$= \sum_{l=1}^{t} \frac{w_l}{|K|} \operatorname{arccosh}^2 \left(1 - \frac{2K\|x^{(l)} - y\|^2}{(1 + K\|x^{(l)}\|^2)(1 + K\|y\|^2)}\right) \tag{9}$$

Then Algorithm 1 gives a sequence of points $\{y_k\}$ such that their limit $\lim_{k\to\infty} y_k = \mu_{fr}$ converges to the Fréchet mean solution.

Proof. See Theorem E.2 in the appendix. □

The algorithm and proof of convergence for the hyperboloid model are given in Appendix E.1 and are omitted here for brevity.

[5]Here we present the version for $K = -1$ for cleaner presentation. The generalization to arbitrary $K < 0$ is easy to compute, but clutters presentation.

5.1.3. Empirical Comparison to Previous Fréchet Mean Computation Algorithms

To demonstrate the efficacy of our algorithm, we compare it to previous approaches on randomly generated data. Namely, we compare against a naïve Riemannian Gradient Descent (RGD) approach (Udrişte, 1994) and against the Karcher Flow algorithm (Karcher, 1977). We test our Fréchet mean algorithm against these methods on synthetic datasets of ten on-manifold randomly generated 16-dimensional points. We run all algorithms until they are within $\epsilon = 10^{-12}$ of the true Fréchet mean in norm, and report the number of iterations this takes in Table 3 for both hyperboloid (H) and Poincaré (P) models of hyperbolic space. Note that we significantly outperform the other algorithms. We also observe that by allowing 200x more computation, a grid search on the learning hyperparameter[6] in RGD obtains nearly comparable or better results (last row of Table 3 for both models). However, we stress that this requires much more computation, and note that our algorithm produces nearly the same result while being **hyperparameter-free**.

Table 3. Empirical computation of the Fréchet mean; the average number of iterations, as well as runtime, required to become accurate within $\epsilon = 10^{-12}$ of the true Fréchet mean are reported. 10 trials are conducted, and standard deviation is reported. The primary baselines are the RGD (Udrişte, 1994) and Karcher Flow (Karcher, 1977) algorithms. (H) refers to hyperboloid and (P) refers to Poincaré.

		Iterations	Time (ms)[7]
H	RGD ($lr = 0.01$)	801.0$\pm$21.0	932.9$\pm$130.0
H	Karcher Flow	62.5$\pm$6.0	50.9$\pm$8.9
H	Ours	**13.7**$\pm$0.9	**6.1**$\pm$1.9
H	RGD + Grid Search on lr	27.7$\pm$0.8	5333.5$\pm$770.7
P	RGD ($lr = 0.01$)	773.8$\pm$22.1	1157.3$\pm$74.8
P	Karcher Flow	57.5$\pm$9.1	59.8$\pm$10.4
P	Ours	**13.4**$\pm$0.5	**9.1**$\pm$1.3
P	RGD + Grid Search on lr	10.5$\pm$0.5	6050.6$\pm$235.2

We also find that this convergence improvement translates to real world applications. Specifically, we find that for the graph link prediction experimental setting in Section 6.1.3, our forward pass takes anywhere from $\approx 15 - 25$ iterations, significantly outperforming the $1000+$ needed with RGD and ≈ 120 needed with Karcher Flow.

[6]The grid search starts from $lr = 0.2$ and goes to $lr = 0.4$ in increments of 0.01 for the Poincaré ball model, and from $lr = 0.2$ to 0.28 for the hyperboloid model (same increment).

[7]Experiments were run with an Intel Skylake Core i7-6700HQ 2.6 GHz Quad core CPU.

5.2. Backward Computation of the Hyperbolic Fréchet Mean

For the backward computation, we re-apply the general Riemannian theory for differentiating through the Fréchet mean in Section 4 to hyperbolic space. Since most autodifferentiation packages do not support manifold-aware higher order differentiation, we derive the gradients explicitly. We begin with the Poincaré ball model by setting $\mathcal{M} = \mathbb{D}_K^n$ and applying Theorem 4.2.

Theorem 5.2. *Let $x^{(1)}, \cdots, x^{(t)} \in \mathbb{D}_K^n \subseteq \mathbb{R}^n$ be t points in the Poincaré ball and $w_1, \ldots, w_t \in \mathbb{R}^+$ be the weights. Let their weighted Fréchet mean μ_{fr} be solution to the following optimization problem*

$$\mu_{fr}(x^{(1)}, \cdots, x^{(t)}) = \operatorname*{arg\,min}_{y \in \mathbb{D}_K^n} f(\{x\}, y) \tag{10}$$

$$\text{where } f(\{x\}, y) = \sum_{l=1}^{t} w_l \cdot d_{\mathbb{D}_K^n}(x^{(l)}, y)^2 =$$

$$\sum_{l=1}^{t} \frac{w_l}{|K|} \operatorname{arccosh}^2 \left(1 - \frac{2K||x^{(l)} - y||_2^2}{(1 + K||x^{(l)}||_2^2)(1 + K||y||_2^2)}\right) \tag{11}$$

Then the derivative of μ_{fr} with respect to $x^{(i)}$ is given by

$$\widetilde{\nabla}_{x^{(i)}} \mu_{fr}(\{x\}) = -f_{YY} f(\{x\}, \overline{x})^{-1} f_{X^{(i)}Y}(\{x\}, \overline{x}) \tag{12}$$

where $\overline{x} = \mu_{fr}(\{x\})$ and f_{YY}, $f_{X^{(i)}Y}$ are defined in Theorem 4.2 [8].

The full concrete derivation of the above terms for the geometry induced by this manifold choice is given in Appendix Theorem F.3.

Proof. This is a concrete application of Theorem 4.2. In particular since our manifold is embedded in $\mathbb{R}^n$ ($\mathbb{D}_K^n \subseteq \mathbb{R}^n$). Note that this is the total derivative in the ambient Euclidean space[9]. For the full proof see Theorem F.3 in the Appendix. $\square$

The derivation for the hyperboloid model is given in Appendix F.2.

6. Case Studies

To demonstrate the efficacy of our developed theory, we investigate the following test settings. In the first setting, we directly modify the hyperbolic aggregation strategy in Hyperbolic GCNs (Chami et al., 2019) to use our differentiable Fréchet mean layer. This was the original intent[10] but was not feasible without our formulation. In the second setting, we introduce Hyperbolic Batch Normalization (HBN) as an extension of the regular Euclidean Batch Normalization (EBN). When combined with hyperbolic neural networks (Ganea et al., 2018), HBN exhibits benefits similar to those of EBN with Euclidean networks.

6.1. Hyperbolic Graph Convolutional Neural Networks (HGCNs)

6.1.1. Original Framework

Introduced in Chami et al. (2019), Hyperbolic Graph Convolutional Networks (GCNs) provide generalizations of Euclidean GCNs to hyperbolic space. The proposed network architecture is based on three different layer types: feature transformation, activation, and attention-based aggregation.

Feature transformation: The hyperbolic feature transformation consists of a gyrovector matrix multiplication followed by a gyrovector addition.

$$h_i^l = (W^l \otimes^{K_{l-1}} x_i^{l-1}) \oplus^{K_{l-1}} b^l \tag{13}$$

Attention-based aggregation: Neighborhood aggregation combines local data at a node. It does so by projecting the neighbors using the logarithmic map at the node, averaging in the tangent space, and projecting back with the exponential map at the node. Note that the weights w_{ij} are positive and can be trained or defined by the graph adjacency matrix.

$$AGG^K(x_i) = \exp_{x_i}^K \left(\sum_{j \in \mathcal{N}(i)} w_{ij} \log_{x_i}^K x_j \right) \tag{14}$$

Activation: The activation layer applies a hyperbolic activation function.

$$x_i^l = \sigma^{\otimes^{K_{l-1}, K_l}}(y_i^l) \tag{15}$$

6.1.2. Proposed Changes

The usage of tangent space aggregation in the HGCN framework stemmed from the lack of a differentiable Fréchet mean operation. As a natural extension, we substitute our Fréchet mean in place of the aggregation layer.

6.1.3. Experimental Results

We use precisely the same architecture as in Chami et al. (2019), except we substitute all hyperbolic aggregation layers with our differentiable Fréchet mean layer. Furthermore,

[8] The projection operation is trivial since $\dim \mathbb{R}^n = \dim \mathbb{D}_K^n$.

[9] To transform Euclidean gradients into Riemannian ones, simply multiply by inverse of the matrix of the metric.

[10] We quote directly from the paper Chami et al. (2019): "An analog of mean aggregation in hyperbolic space is the Fréchet mean, which, however, has no closed form solution. Instead, we propose to..."

we test with precisely the same hyperparameters (learning rate, test/val split, and the like) as Chami et al. (2019) for a fair comparison. Our new aggregation allows us to achieve new state-of-the-art results on the Disease and Disease-M graph datasets (Chami et al., 2019). These datasets induce ideal test tasks for hyperbolic learning since they have very low Gromov δ-hyperbolicity (Adcock et al., 2013), which indicates the structure is highly tree-like. Our results and comparison to the baseline are given in Table 4. We run experiments for 5 trials and report the mean and standard deviation. Due to practical considerations, we only test with the Poincaré model[11]. For reference, the strongest baseline results with the hyperboloid model are reported from Chami et al. (2019) (note that we outperform these results as well). On the rather non-hyperbolic CoRA (Sen et al., 2008) dataset, our performance is comparable to that of the best baseline. Note that this is similar to the performance exhibited by the vanilla HGCN. Hence we conjecture that when the underlying dataset is not hyperbolic in nature, we do not observe improvements over the best Euclidean baseline methods.

Table 4. ROC AUC results for Link Prediction (LP) on various graph datasets, averaged over 5 trials (with standard deviations). Graph hyperbolicity values are also reported (lower δ is more hyperbolic). Results are given for models learning in Euclidean (E), Hyperboloid (H), and Poincaré (P) spaces. Note that the best Euclidean method is GAT (Veličković et al., 2018) and is shown below for fair comparison on CoRA. We highlight the best result only if our result gives a p-value < 0.01 after running a paired-significance t-test.

		Disease $\delta = 0$	Disease-M $\delta = 0$	CoRA $\delta = 11$
E	MLP	72.6±0.6	55.3±0.5	83.1±0.5
	GAT	69.8±0.3	69.5±0.4	**93.7**±0.1
H	HNN	75.1±0.3	60.9±0.4	89.0±0.1
	HGCN	90.8±0.3	78.1±0.4	92.9±0.1
P	HGCN	76.4±8.9	81.4±3.4	93.4±0.4
	Ours	**93.7**±0.4	**91.0**±0.6	92.9±0.4

6.2. Hyperbolic Batch Normalization

Euclidean batch normalization (Ioffe & Szegedy, 2015) is one of the most widely used neural network operations that has, in many cases, obviated the need for explicit regularization such as dropout (Srivastava et al., 2014). In particular, analysis demonstrates that batch normalization induces a smoother loss surface which facilitates convergence and yields better final results (Santurkar et al., 2018). Generalizing this for Riemannian manifolds is a natural extension, and such a computation would involve a differentiable Fréchet mean.

6.2.1. Theoretical Formulation and Algorithm

In this section we formulate Riemannian Batch Normalization as a natural extension of standard Euclidean Batch Normalization. This concept is, to the best of our knowledge, only touched upon by Brooks et al. (2019) in the specific instance of the manifold of positive semidefinite matrices. However, we argue in Appendix G that, unlike our method, their formulation is incomplete and lacks sufficient generality to be considered a true extension.

Algorithm 2 Riemannian Batch Normalization

```
**Training Input**: Batches of data points $\{x_1^{(t)}, \cdots, x_m^{(t)}\} \subseteq \mathcal{M}$ for $t \in [1, \dots, T]$, testing momentum $\eta \in [0, 1]$
**Learned Parameters**: Target mean $\mu' \in \mathcal{M}$, target variance $(\sigma')^2 \in \mathbb{R}$
**Training Algorithm**:
$\mu_{test} \leftarrow \text{FrechetMean}(\{x_1^{(1)}, \dots, x_m^{(1)}\})$
$\sigma_{test} \leftarrow 0$
for $t = 1, \dots, T$:
    $\mu = \text{FrechetMean}(\{x_1^{(t)}, \dots, x_m^{(t)}\})$
    $\sigma^2 = \frac{1}{m} \sum_{i=1}^{m} d(x_i^{(t)}, \mu)^2$
    $\mu_{test} = \text{FrechetMean}(\{\mu_{test}, \mu\}, \{\eta, 1 - \eta\})$
    $\sigma_{test} = \frac{(t-1)\sigma_{test} + \sigma}{t}$
    for $i = 1, \cdots, m$:
        $\tilde{x_i}^{(t)} \leftarrow \exp_{\mu'}\left(\frac{\sigma'}{\sigma} PT_{\mu \to \mu'}(\log_\mu x_i^{(t)})\right)$
    return normalized batch $\tilde{x_1}^{(t)}, \cdots, \tilde{x_m}^{(t)}$

**Testing Input**: Test data points $\{\overline{x_1}, \cdots, \overline{x_s}\} \subseteq \mathcal{M}$, final running mean $\mu_{test}$ and running variance $\sigma_{test}$
**Testing Algorithm**:
$\overline{\mu} = \text{FrechetMean}(\{\overline{x_1}, \cdots, \overline{x_s}\})$
$\overline{\sigma}^2 = \frac{1}{m} \sum_{i=1}^{m} d(\overline{x_i}, \overline{\mu})^2$
for $i = 1, \cdots, s$:
    $\tilde{\overline{x_i}} \leftarrow \exp_{\mu_{test}}\left(\frac{\sigma_{test}}{\overline{\sigma}} PT_{\overline{\mu} \to \mu_{test}}(\log_{\overline{\mu}} \overline{x_i})\right)$
return normalized batch $\tilde{\overline{x_1}}, \cdots, \tilde{\overline{x_s}}$
```

Our full algorithm is given in Algorithm 2. Note that in practice we use $\sqrt{\sigma^2 + \epsilon}$ in place of σ as in the original formulation to avoid division by zero.

6.2.2. Experimental Results

We apply Riemannian Batch Normalization (specifically for hyperbolic space) to the encoding Hyperbolic Neural Network (HNN) (Ganea et al., 2018) in the framework of

[11]The code for HGCN included only the Poincaré model implementation at the time this paper was submitted. Hence we use the Poincaré model for our experiments, although our contributions include derivations for both hyperboloid and Poincaré models.

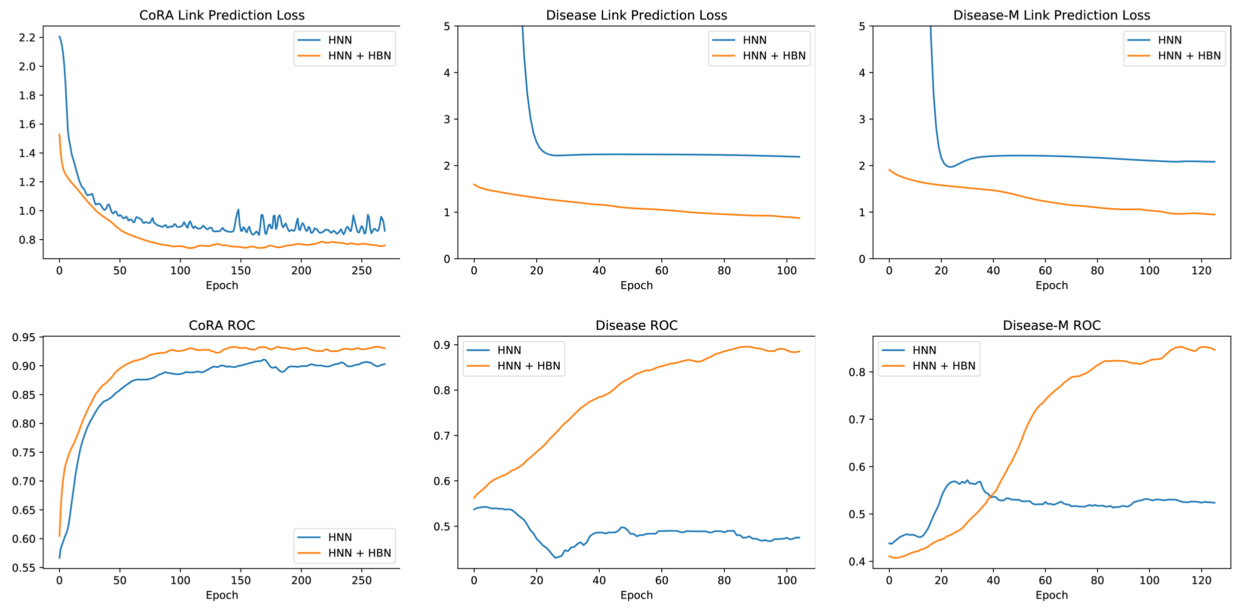

Figure 3. The graphs above correspond to a comparison of the HNN baseline, which uses a two-layer hyperbolic neural network encoder, and the baseline augmented with hyperbolic batch normalization after each layer. The columns correspond to the CoRA (Sen et al., 2008), Disease (Chami et al., 2019), and Disease-M (Chami et al., 2019) datasets, respectively. The top row shows the comparison in terms of validation loss, and the bottom row shows the comparison in terms of validation ROC AUC. The figures show that we converge faster and attain better performance in terms of both loss and ROC. Note that although CoRA is not hyperbolic (as previously mentioned), we find it encouraging that introducing hyperbolic batch normalization produces an improvement regardless of dataset hyperbolicity.

Chami et al. (2019). We run on the CoRA (Sen et al., 2008), Disease (Chami et al., 2019), and Disease-M (Chami et al., 2019) datasets and present the validation loss and ROC AUC diagrams in Figure 3.

In terms of both loss and ROC, our method results in both faster convergence and a better final result. These improvements are expected as they appear when applying standard batch normalization to Euclidean neural networks. So, our manifold generalization does seem to replicate the useful properties of standard batch normalization. Additionally, it is encouraging to see that, regardless of the hyperbolic nature of the underlying dataset, hyperbolic batch normalization produces an improvement when paired with a hyperbolic neural network.

7. Conclusion and Future Work

We have presented a fully differentiable Fréchet mean operation for use in any differentiable programming setting. Concretely, we introduced differentiation theory for the general Riemannian case, and for the demonstrably useful case of hyperbolic space, we provided a fast forward pass algorithm and explicit derivative computations. We demonstrated that using the Fréchet mean in place of tangent space aggregation yields state-of-the-art performance on link prediction tasks in graphs with tree-like structure. Additionally, we extended batch normalization (a standard Euclidean operation) to the realm of hyperbolic space. On a graph link prediction test task, we showed that hyperbolic batch normalization gives benefits similar to those experienced in the Euclidean setting.

We hope our work paves the way for future developments in geometric representation learning. Potential future work can focus on speeding up our computation of the Fréchet mean gradient, finding applications of our theory on manifolds beyond hyperbolic space, and applying the Fréchet mean to generalize more standard neural network operations.

8. Acknowledgements

We would like to acknowledge Horace He for his helpful comments regarding implementation. In addition, we would like to thank Facebook AI for funding equipment that made this work possible.

References

Adcock, A. B., Sullivan, B. D., and Mahoney, M. W. Tree-like structure in large social and information networks. *2013 IEEE 13th International Conference on Data Mining*, pp. 1–10, 2013.

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, pp. 9558–9570, 2019.

Amos, B. and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 136–145, 2017.

Bacák, M. Computing medians and means in hadamard spaces. *SIAM Journal on Optimization*, 24(3):1542–1566, 2014.

Brooks, D., Schwander, O., Barbaresco, F., Schneider, J.-Y., and Cord, M. Riemannian batch normalization for spd neural networks. In *Advances in Neural Information Processing Systems*, pp. 15463–15474, 2019.

Casado, M. L. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems*, pp. 9154–9164, 2019.

Chami, I., Ying, Z., Ré, C., and Leskovec, J. Hyperbolic graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 4869–4880, 2019.

Charlier, B. Necessary and sufficient condition for the existence of a fréchet mean on the circle. *ESAIM: Probability and Statistics*, 17:635–649, 2013.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, pp. 6571–6583, 2018.

Fréchet, M. Les éléments aléatoires de nature quelconque dans un espace distancié. In *Annales de l'institut Henri Poincaré*, volume 10, pp. 215–310, 1948.

Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In *Advances in Neural Information Processing Systems*, pp. 5345–5355, 2018.

Gould, S., Fernando, B., Cherian, A., Anderson, P., Cruz, R. S., and Guo, E. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *ArXiv*, abs/1607.05447, 2016.

Gu, A., Sala, F., Gunel, B., and Ré, C. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2019.

Gulcehre, C., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., Santoro, A., and de Freitas, N. Hyperbolic attention networks. In *International Conference on Learning Representations*, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pp. 448–456, 2015.

Karcher, H. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30(5):509–541, 1977.

Kleinberg, R. D. Geographic routing using hyperbolic space. *IEEE INFOCOM 2007 - 26th IEEE International Conference on Computer Communications*, pp. 1902–1909, 2007.

Law, M., Liao, R., Snell, J., and Zemel, R. Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning*, pp. 3672–3681, 2019.

Lee, J. *Riemannian Manifolds: An Introduction to Curvature*. Graduate Texts in Mathematics. Springer New York, 1997.

Lee, J. M. Introduction to smooth manifolds. *Graduate Texts in Mathematics*, 218, 2003.

Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. In *Advances in Neural Information Processing Systems*, pp. 8228–8239, 2019.

Nickel, M. and Kiela, D. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, pp. 6338–6347, 2017.

Nickel, M. and Kiela, D. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3779–3788, 2018.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013.

Pogančić, M. V., Paulus, A., Musil, V., Martius, G., and Rolinek, M. Differentiation of blackbox combinatorial solvers. In *International Conference on Learning Representations*, 2020.

Sala, F., Sa, C. D., Gu, A., and Ré, C. Representation tradeoffs for hyperbolic embeddings. *Proceedings of Machine Learning Research*, 80:4460–4469, 2018.

Santurkar, S., Tsipras, D., Ilyas, A., and Madry, A. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pp. 2483–2493, 2018.

Sarkar, R. 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.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., and Eliassi-Rad, T. Collective classification in network data. *AI Magazine*, 29:93–106, 2008.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Tifrea, A., Becigneul, G., and Ganea, O.-E. Poincaré glove: Hyperbolic word embeddings. In *International Conference on Learning Representations*, 2019.

Udrişte, C. *Convex functions and optimization methods on Riemannian manifolds*. Mathematics and Its Applications. Springer, Dordrecht, 1994. doi: 10.1007/978-94-015-8390-9.

Ungar, A. A. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194, 2008.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *ArXiv*, abs/1706.03762, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Wolter, F.-E. Distance function and cut loci on a complete riemannian manifold. *Archiv der Mathematik*, 32(1):92–96, 1979. doi: 10.1007/BF01238473.

Yu, T. and De Sa, C. M. Numerically accurate hyperbolic embeddings using tiling-based models. In *Advances in Neural Information Processing Systems*, pp. 2021–2031, 2019.