Latent Variable Modelling with Hyperbolic Normalizing Flows

Avishek Joey Bose [1 2] **Ariella Smofsky** [1 2] **Renjie Liao** [3 4] **Prakash Panangaden** [1 2] **William L. Hamilton** [1 2]

Abstract

The choice of approximate posterior distributions plays a central role in stochastic variational inference (SVI). One effective solution is the use of normalizing flows to construct flexible posterior distributions. However, one key limitation of existing normalizing flows is that they are restricted to the Euclidean space and are ill-equipped to model data with an underlying hierarchical structure. To address this fundamental limitation, we present the first extension of normalizing flows to hyperbolic spaces. We first elevate normalizing flows to hyperbolic spaces using coupling transforms defined on the tangent bundle, termed Tangent Coupling ($\mathcal{TC}$). We further introduce Wrapped Hyperboloid Coupling ($\mathcal{WHC}$), a fully invertible and learnable transformation that explicitly utilizes the geometric structure of hyperbolic spaces, allowing for expressive posteriors while being efficient to sample from. We demonstrate the efficacy of our novel normalizing flow over hyperbolic VAEs and Euclidean normalizing flows. Our approach achieves improved performance on density estimation, as well as reconstruction of real-world graph data, which exhibit a hierarchical structure. Finally, we show that our approach can be used to power a generative model over hierarchical data using hyperbolic latent variables.

1. Introduction

Stochastic variational inference (SVI) methods provide an appealing way of scaling probabilistic modeling to large scale data. These methods transform the problem of computing an intractable posterior distribution to finding the best approximation within a class of tractable probability distributions (Hoffman et al., 2013). Using tractable classes of approximate distributions, *e.g.*, mean-field, and Bethe approximations, facilitates efficient inference, at the cost of limiting the expressiveness of the learned posterior.

[1]McGill University [2]Mila [3]University of Toronto [4]Vector Institute. Correspondence to: Joey Bose <joey.bose@mail.mcgill.ca>.

Preprint. Under Review.

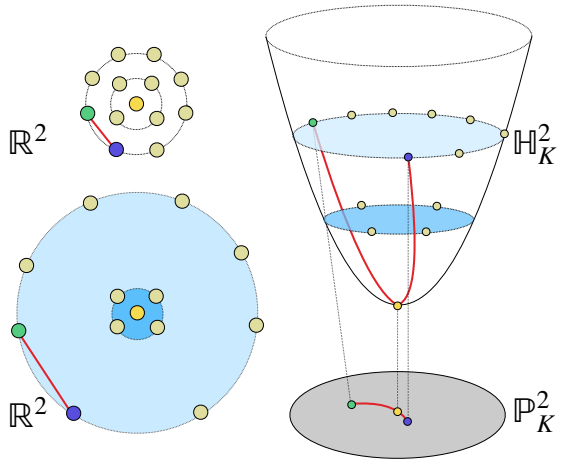

Figure 1. The shortest path between a given pair of node embeddings in $\mathbb{R}^2$ and hyperbolic space as modelled by the Lorentz model $\mathbb{H}^2_K$ and Poincaré disk $\mathbb{P}^2_K$. Unlike Euclidean space, distances between points grow exponentially as you move away from the origin in hyperbolic space, and thus the shortest paths between points in hyperbolic space go through the origin, giving rise to hierarchical, tree-like structure.

In recent years, the power of these SVI methods has been further improved by employing *normalizing flows*, which greatly increase the flexibility of the approximate posterior distribution. Normalizing flows involve learning a series of invertible transformations, which are used to transform a sample from a simple base distribution to a sample from a richer distribution (Rezende & Mohamed, 2015). Indeed, flow-based posteriors enjoy many advantages such as efficient sampling, exact likelihood estimation, and low-variance gradient estimates when the base distribution is reparametrizable, making them ideal for modern machine learning problems. There have been numerous advances in normalizing flow construction in Euclidean spaces from RealNVP (Dinh et al., 2017), NAF (Huang et al., 2018), and FFJORD (Grathwohl et al., 2018) to name a few.

However, current normalizing flows are restricted to Euclidean space, and as a result, these approaches are ill-equipped to model data with an underlying hierarchical structure. Many real-world datasets, such as ontologies, social networks, sentences in natural language, and evolutionary relationships between biological entities in phylogenetics exhibit rich hierarchical or tree-like structure. Hierarchical data of this kind can be naturally represented in

hyperbolic spaces, *i.e.*, non-Euclidean spaces with constant negative curvature (Figure 1). But Euclidean normalizing flows fail to incorporate these structural inductive biases, since Euclidean space cannot embed deep hierarchies without suffering from high distortion (Sarkar, 2011). Furthermore, sampling from densities defined on Euclidean space will inevitability generate points that do not lie on the underlying hyperbolic space.

Present work. To address this fundamental limitation, we present the first extension of normalizing flows to hyperbolic spaces. Prior works have considered learning models with hyperbolic parameters (Liu et al., 2019b; Nickel & Kiela, 2018) as well as variational inference with hyperbolic latent variables (Nagano et al., 2019; Mathieu et al., 2019), but our work represents the first approach to allow flexible density estimation in hyperbolic space.

To define our normalizing flows we leverage the Lorentz model of hyperbolic geometry and introduce two new forms of coupling, *Tangent Coupling* ($\mathcal{TC}$) and *Wrapped Hyperboloid Coupling* ($\mathcal{WHC}$). These define flexible and invertible transformations capable of transforming sampled points in the hyperbolic space. We derive the change of volume associated with these transformations and show that it can be computed efficiently with $\mathcal{O}(n)$ cost, where n is the dimension of the hyperbolic space. We empirically validate our proposed normalizing flows on structured density estimation, reconstruction and generation tasks on hierarchical data, highlighting the utility of our proposed approach.

2. Background on Hyperbolic Geometry

Within the Riemannian geometry framework, hyperbolic spaces are manifolds with constant negative curvature K and are of particular interest for embedding hierarchical structures. There are multiple models of n-dimensional hyperbolic space, such as the hyperboloid $\mathbb{H}^n_K$, also known as the Lorentz model, or the Poincaré ball $\mathbb{P}^n_K$. Figure 1 illustrates some key properties of $\mathbb{H}^2_K$ and $\mathbb{P}^2_K$, highlighting how distances grow exponentially as you move away from the origin and how the shortest paths between distant points tend to go through the origin, giving rise to a hierarchical or tree-like structure. In the next section, we briefly review the Lorentz model of hyperbolic geometry. We are not assuming a background in Riemannian geometry, though Appendix A and Ratcliffe (1994) are of use to the interested reader. Henceforth, for notational clarity, we use boldface font to denote points on the hyperboloid manifold.

2.1. Lorentz Model of Hyperbolic Geometry

An n-dimensional hyperbolic space, $\mathbb{H}^n_K$, is the unique, complete, simply-connected n-dimensional Riemannian manifold of constant negative curvature, K. For our purposes, the Lorentz model is the most convenient representation of hyperbolic space, since it is equipped with relatively simple explicit formulas and useful numerical stability properties (Nickel & Kiela, 2018). We choose the 2D Poincaré disk $\mathbb{P}^2_1$ to visualize hyperbolic space because of its conformal mapping to the unit disk. The Lorentz model embeds hyperbolic space $\mathbb{H}^n_K$ within the $n+1$-dimensional Minkowski space, defined as the manifold $\mathbb{R}^{n+1}$ equipped with the following inner product:

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} := -x_0 y_0 + x_1 y_1 + \cdots + x_n y_n, \tag{1}$$

which has the type $\langle \cdot, \cdot \rangle_{\mathcal{L}} : \mathbb{R}^{n+1} \times \mathbb{R}^{n+1} \to \mathbb{R}$. It is common to denote this space as $\mathbb{R}^{1,n}$ to emphasize the distinct role of the zeroth coordinate. In the Lorentz model, we model hyperbolic space as the (upper sheet of) the hyperboloid embedded in Minkowski space. It is a remarkable fact that though the Lorentzian metric (Eq. 1) is indefinite, the induced Riemannian metric $g_{\mathbf{x}}$ on the unit hyperboloid is positive definite (Ratcliffe, 1994). The n-Hyperbolic space with constant negative curvature K with origin $\mathbf{o} = (1/K, 0, \ldots, 0)$, is a Riemannian manifold $(\mathbb{H}^n_K, g_{\mathbf{x}})$ where

$$\mathbb{H}^n_K := \{x \in \mathbb{R}^{n+1} : \langle \mathbf{x}.\mathbf{x} \rangle_{\mathcal{L}} = 1/K,\ x_0 > 0,\ K < 0\}.$$

Equipped with this, the induced distance between two points $(\mathbf{x}, \mathbf{y})$ in $\mathbb{H}^n_K$ is given by

$$d(\mathbf{x}, \mathbf{y})_{\mathcal{L}} := \frac{1}{\sqrt{-K}} \operatorname{arccosh}(-K \langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}). \tag{2}$$

The tangent space to the hyperboloid at the point $\mathbf{p} \in \mathbb{H}^n_K$ can also be described as an embedded subspace of $\mathbb{R}^{1,n}$. It is given by the set of points satisfying the orthogonality relation with respect to the Minkowski inner product [1],

$$\mathcal{T}_{\mathbf{p}} \mathbb{H}^n_K := \{u : \langle u, \mathbf{p} \rangle_{\mathcal{L}} = 0\}. \tag{3}$$

Of special interest are vectors in the tangent space at the origin of $\mathbb{H}^n_K$ whose norm under the Minkowski inner product is equivalent to the conventional Euclidean norm. That is $v \in \mathcal{T}_{\mathbf{o}} \mathbb{H}^n_K$ is a vector such that $v_0 = 0$ and $||\mathbf{v}||_{\mathcal{L}} := \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{\mathcal{L}}} = ||\mathbf{v}||_2$. Thus *at the origin* the partial derivatives with respect to the ambient coordinates, $\mathbb{R}^{n+1}$, define the covariant derivative.

Projections. Starting from the extrinsic view by which we consider $\mathbb{R}^{n+1} \supset \mathbb{H}^n_K$, we may project any vector $x \in \mathbb{R}^{n+1}$ on to the hyperboloid using the shortest Euclidean distance:

$$\text{proj}_{\mathbb{H}^n_K}(x) = \frac{x}{\sqrt{-K}||x||_{\mathcal{L}}}. \tag{4}$$

[1] It is also equivalently known as the Lorentz inner product.

Furthermore, by definition a point on the hyperboloid satisfies $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = 1/K$ and thus when provided with n co-ordinates $\hat{x} = (x_1, \ldots, x_n)$ we can always determine the missing coordinate to get a point on $\mathbb{H}^n_K$:

$$x_0 = \sqrt{||\hat{x}||_2^2 + \frac{1}{K}}. \tag{5}$$

Exponential Map. The exponential map takes a vector, v, in the tangent space of a point $\mathbf{x} \in \mathbb{H}^n_K$ to a point on the manifold—i.e., $\mathbf{y} = \exp^K_{\mathbf{x}}(v) : \mathcal{T}_{\mathbf{x}}\mathbb{H}^n_K \to \mathbb{H}^n_K$ by moving a unit length along the *geodesic*, γ (straightest parametric curve), uniquely defined by $\gamma(0) = \mathbf{x}$ with direction $\gamma'(0) = v$. The closed form expression for the exponential map is then given by

$$\exp^K_{\mathbf{x}}(v) = \cosh\left(\frac{||v||_{\mathcal{L}}}{R}\right)\mathbf{x} + \sinh\left(\frac{||v||_{\mathcal{L}}}{R}\right)\frac{Rv}{||v||_{\mathcal{L}}}, \tag{6}$$

where we used the *generalized radius* $R = 1/\sqrt{-K}$ in place of the curvature.

Logarithmic Map. As the inverse of the exponential map, the logarithmic map takes a point, $\mathbf{y}$, on the manifold back to the tangent space of another point $\mathbf{x}$ also on the manifold. In the Lorentz model this is defined as

$$\log^K_{\mathbf{x}}(\mathbf{y}) = \frac{\operatorname{arccosh}(\alpha)}{\sqrt{\alpha^2 - 1}}(\mathbf{y} - \alpha\mathbf{x}), \tag{7}$$

where $\alpha = K\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}$.

Parallel Transport. The parallel transport for two points $\mathbf{x}, \mathbf{y} \in \mathbb{H}^n_K$ is a map that carries the vectors in $v \in \mathcal{T}_{\mathbf{x}}\mathbb{H}^n_K$ to corresponding vectors at $v' \in \mathcal{T}_{\mathbf{y}}\mathbb{H}^n_K$ along the geodesic. That is vectors are connected between the two tangent spaces such that the covariant derivative is unchanged. Parallel transport is a map that preserves the metric, *i.e.*, $\langle \mathrm{PT}^K_{\mathbf{x}\to\mathbf{y}}(v), \mathrm{PT}^K_{\mathbf{x}\to\mathbf{y}}(v') \rangle_{\mathcal{L}} = \langle v, v' \rangle_{\mathcal{L}}$ and in the Lorentz model is given by

$$\begin{aligned}\mathrm{PT}^K_{\mathbf{x}\to\mathbf{y}}(v) &= v - \frac{\langle \log^K_{\mathbf{x}}(\mathbf{y}), v \rangle_{\mathcal{L}}}{d(\mathbf{x}, \mathbf{y})_{\mathcal{L}}}(\log^K_{\mathbf{x}}(\mathbf{y}) + \log^K_{\mathbf{y}}(\mathbf{x})) \\ &= v + \frac{\langle \mathbf{y}, v \rangle_{\mathcal{L}}}{R^2 - \langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}}(\mathbf{x} + \mathbf{y}),\end{aligned} \tag{8}$$

where α is as defined above. Another useful property is that the inverse parallel transport simply carries the vectors back along the geodesic and is simply defined as $(\mathrm{PT}^K_{\mathbf{x}\to\mathbf{y}}(v))^{-1} = \mathrm{PT}^K_{\mathbf{y}\to\mathbf{x}}(v)$.

2.2. Probability Distributions on Hyperbolic Spaces

Probability distributions can be defined on Riemannian manifolds, which include $\mathbb{H}^n_K$ as a special case. One transforms the infinitesimal volume element on the manifold to the corresponding volume element in $\mathbb{R}^n$ as defined by the co-ordinate charts. In particular, given the Riemannian manifold $\mathcal{M}(\mathbf{z})$ and its metric $g_{\mathbf{z}}$, we have $\int p(\mathbf{z}) d\mathcal{M}(\mathbf{z}) = \int p(\mathbf{z})\sqrt{|g_{\mathbf{z}}|}d\mathbf{z}$, where $d\mathbf{z}$ is the Lebesgue measure. We now briefly survey three distinct generalizations of the normal distribution to Riemannian manifolds.

Riemannian Normal. The first is the Riemannian normal (Pennec, 2006; Said et al., 2014), which is derived from maximizing the entropy given a Fréchet mean μ and a dispersion parameter σ. Specifically, we have $\mathcal{N}_{\mathcal{M}}(\mathbf{z}|\mu, \sigma^2) = \frac{1}{Z}\exp\left(-d_{\mathcal{M}}(\mu, \mathbf{z})^2/2\sigma^2\right)$, where $d_{\mathcal{M}}$ is the *induced distance* and Z is the normalization constant (Said et al., 2014; Mathieu et al., 2019).

Restricted Normal. One can also restrict sampled points from the normal distribution in the ambient space to the manifold. One example is the Von Mises distribution on the unit circle and its generalized version, *i.e.*, Von Mises-Fisher distribution on the hypersphere (Davidson et al., 2018).

Wrapped Normal. Finally, we can define a wrapped normal distribution (Falorsi et al., 2019; Nagano et al., 2019), which is obtained by (1) sampling from $\mathcal{N}(0, I)$ and then transforming it to a point $v \in \mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$ by concatenating 0 as the zeroth coordinate; (2) parallel transporting the sample v from the tangent space at $\mathbf{o}$ to the tangent space of another point $\boldsymbol{\mu}$ on the manifold to obtain u; (3) mapping u from the tangent space to the manifold using the exponential map at $\boldsymbol{\mu}$. Sampling from such a distribution is straightforward and the probability density can be obtained via the change of variable formula,

$$\log p(\mathbf{z}) = \log p(v) - (n-1)\log\left(\frac{\sinh\left(\|u\|_{\mathcal{L}}\right)}{\|u\|_{\mathcal{L}}}\right), \tag{9}$$

where $p(\mathbf{z})$ is the wrapped normal distribution and $p(v)$ is the normal distribution in the tangent space of $\mathbf{o}$.

3. Normalizing Flows on Hyperbolic Spaces

We seek to define flexible and learnable distributions on $\mathbb{H}^n_K$, which will allow us to learn rich approximate posterior distributions for hierarchical data. To do so, we design a class of invertible parametric hyperbolic functions, $f_i : \mathbb{H}^n_K \to \mathbb{H}^n_K$. A sample from the approximate posterior can then be obtained by first sampling from a simple base distribution $\mathbf{z}_0 \sim p(\mathbf{z})$ defined on $\mathbb{H}^n_K$ and then applying a composition of functions $f_{i\in[j]}$ from this class: $\mathbf{z}_j = f_j \circ f_{j-1} \circ \cdots \circ f_1(\mathbf{z}_0)$.

In order to ensure effective and tractable learning, the class of functions f_i must satisfy three key desiderata:

1. Each function f_i must be invertible.
2. We must be able to efficiently sample from the final distribution, $\mathbf{z}_j = f_j \circ f_{j-1} \circ \cdots \circ f_1(\mathbf{z}_0)$.
3. We must be able to efficiently compute the associated change in volume —i.e., the Jacobian determinant, of the overall transformation.

Given these requirements, the final transformed distribution is given by the change of variables formula:

$$\log p(\mathbf{z}_j) = \log p(\mathbf{z}_0) - \sum_{i=1}^{k} \log \det \Big| \frac{\partial f_j}{\partial z_{j-1}} \Big|. \tag{10}$$

Functions satisfying desiderata 1-3 in Euclidean space are often termed *normalizing flows* (Appendix B), and our work extends this idea to hyperbolic spaces. In the following sections, we describe two flows of increasing complexity, Tangent Coupling ($\mathcal{TC}$) and Wrapped Hyperboloid Coupling ($\mathcal{WHC}$). The first approach lifts a standard Euclidean flow to the tangent space at the origin of the hyperboloid. The second approach modifies the flow to explicitly utilize hyperbolic geometry. Figure 2 illustrates synthetic densities as learned by our approach on $\mathbb{P}_1^2$.

3.1. Tangent Coupling

Similar to the Wrapped Normal distribution (Section 2.2), one strategy to define a normalizing flow on the hyperboloid is to use the tangent space at the origin. That is, we first sample a point from our base distribution—which we define to be a Wrapped Normal—and use the logarithmic map at the origin to transport it to the corresponding tangent space. Once we arrive at the tangent space we are free to apply any Euclidean flow before finally projecting back to the manifold using the exponential map. This approach leverages the fact that the tangent bundle of a hyperbolic manifold has a well-defined vector space structure, allowing affine transformations and other operations that are ill-defined on the manifold itself.

Following this idea, we build upon one of the earliest and most well-studied flows: the RealNVP flow (Dinh et al., 2017). At its core, the RealNVP flow uses a computationally symmetric transformation (affine coupling layer) which has the benefit of being fast to evaluate and invert due to its lower triangular Jacobian, whose determinant is cheap to compute. Operationally, the coupling layer is implemented using a binary mask, and partitions some input $\tilde{x}$ into two sets, where the first set, $\tilde{x}_1 := \tilde{x}_{1:d}$ is transformed elementwise independently of other dimensions. The second set, $\tilde{x}_2 := \tilde{x}_{d+1:n}$, is also transformed elementwise but in a way that depends on the first set (see Appendix B.2 for more details). Since all coupling layer operations occur at $\mathcal{T}_{\mathbf{0}}\mathbb{H}^n_K$ we term this form of coupling as Tangent Coupling ($\mathcal{TC}$).

Thus the overall transformation due to one layer of our $\mathcal{TC}$ flow is a composition of a logarithmic map, affine coupling defined on $\mathcal{T}_{\mathbf{0}}\mathbb{H}^n_k$, and an exponential map:

$$\tilde{f}^{\mathcal{TC}}(\tilde{x}) = \begin{cases} \tilde{z}_1 & = \tilde{x}_1 \\ \tilde{z}_2 & = \tilde{x}_2 \odot \sigma(s(\tilde{x}_1)) + t(\tilde{x}_1) \end{cases}$$
$$f^{\mathcal{TC}}(\mathbf{x}) = \exp_{\mathbf{0}}^K(\tilde{f}^{\mathcal{TC}}(\log_{\mathbf{0}}^K(\mathbf{x}))), \tag{11}$$

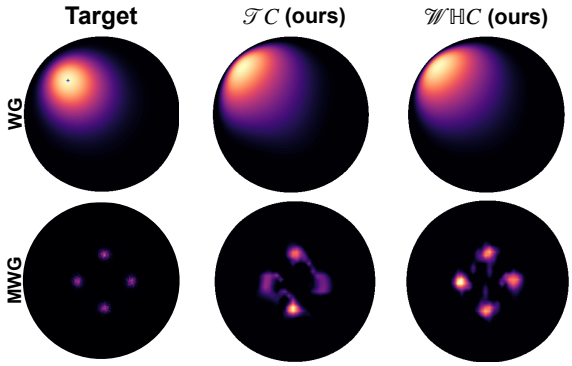

Figure 2. Comparison of density estimation in hyperbolic space for 2D wrapped Gaussian (WG) and mixture of wrapped gaussian (MWG) on $\mathbb{P}_1^2$. Densities are visualized in the Poincaré disk.

where $\tilde{x} = \log_{\mathbf{0}}^K(\mathbf{x})$ is a point on $\mathcal{T}_{\mathbf{0}}\mathbb{H}^n_K$, and σ is a pointwise non-linearity such as the exponential function. Functions s and t are parameterized scale and translation functions implemented as neural nets from $\mathcal{T}_{\mathbf{0}}\mathbb{H}^d_K \to \mathcal{T}_{\mathbf{0}}\mathbb{H}^{n-d}_K$. One important detail is that arbitrary operations on a tangent vector $v \in \mathcal{T}_{\mathbf{0}}\mathbb{H}^n_K$ may transport the resultant vector outside the tangent space, hampering subsequent operations. To avoid this we can keep the first dimension fixed at $v_0 = 0$ to ensure we remain in $\mathcal{T}_{\mathbf{0}}\mathbb{H}^n_K$.

Similar to the Euclidean RealNVP, we need an efficient expression for the Jacobian determinant of $f^{\mathcal{TC}}$.

Proposition 1. *The Jacobian determinant of a single $\mathcal{TC}$ layer in equation 11 is:*

$$\left|\det\Big(\frac{\partial \mathbf{y}}{\partial \boldsymbol{x}}\Big)\right| = \Big(\frac{R\sinh(\frac{||z||_{\mathcal{L}}}{R})}{||z||_{\mathcal{L}}}\Big)^{n-1} \times \prod_{i=d+1}^{n} \sigma(s(\tilde{x}_1))_i$$
$$\times \Big(\frac{R\sinh(\frac{||\log_{\boldsymbol{o}}^K(\boldsymbol{x})||_{\mathcal{L}}}{R})}{||\log_{\boldsymbol{o}}^K(\boldsymbol{x})||_{\mathcal{L}}}\Big)^{1-n} \tag{12}$$

where, $\mathbf{z} = \tilde{f}^{\mathcal{TC}}(\tilde{x})$ and $\tilde{f}^{\mathcal{TC}}$ is as defined above.

Proof Sketch. Here we only provide a sketch of the proof and details can be found in Appendix C. First, observe that the overall transformation is a valid composition of functions: $\mathbf{y} := \exp_{\mathbf{0}}^K \circ \tilde{f}^{\mathcal{TC}} \circ \log_{\mathbf{0}}^K(\mathbf{x})$. Thus, the overall determinant can be computed by chain rule and the identity, $\det\Big(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\Big) = \det\Big(\frac{\partial \exp_{\mathbf{0}}^K(z)}{\partial z}\Big) \cdot \det\Big(\frac{\partial f(\tilde{x})}{\partial \tilde{x}}\Big) \cdot \det\Big(\frac{\partial \log_{\mathbf{0}}^K(\mathbf{x})}{\partial \mathbf{x}}\Big)$. Tackling each function in the composition individually, $\det\Big(\frac{\partial \exp_{\mathbf{0}}^K(z)}{\partial z}\Big) = \Big(\frac{R\sinh(\frac{||z||_{\mathcal{L}}}{R})}{||z||_{\mathcal{L}}}\Big)^{n-1}$ as derived in Skopek et al. (2019). As the logarithmic map is the inverse of the exponential map the Jacobian determinant is simply the inverse of the determinant of the exponential map, which gives the $\det\Big(\frac{\partial \log_{\mathbf{0}}^K(\mathbf{x})}{\partial \mathbf{x}}\Big)$ term. For the middle term, we must calculate the directional derivative of $\tilde{f}^{\mathcal{TC}}$ in an orthonormal basis w.r.t. the Lorentz inner product, of $\mathcal{T}_{\mathbf{0}}\mathbb{H}^n_K$. Since the

standard Euclidean basis vectors $e_1, ..., e_n$ are also a basis for $\mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$, the Jacobian determinant $\det\left(\frac{\partial f(\tilde{x})}{\partial \tilde{x}}\right)$ simplifies to that of the RealNVP flow, which is lower triangluar and is thus efficiently computable in $\mathcal{O}(n)$ time.

□

It is remarkable that the middle term in Proposition 1 is precisely the same change in volume associated with affine coupling in RealNVP. The change in volume due to the hyperbolic space only manifests itself through the exponential and logarithmic maps, each of which can be computed in $\mathcal{O}(n)$ cost. Thus, the overall cost is only slightly larger than the regular Euclidean RealNVP, but still $\mathcal{O}(n)$.

3.2. Wrapped Hyperboloid Coupling

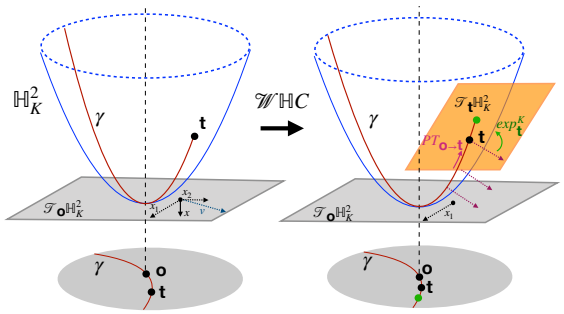

Figure 3. Wrapped Hyperbolic Coupling. The left figure depicts a partitioned input point $\tilde{x}_1 := \tilde{x}_{1:d}$ and $\tilde{x}_2 := \tilde{x}_{d+1:n}$ prior to parallel transport. The right figure depicts the $\tilde{x}_2$ vector after it is transformed, parallel transported, and projected to $\mathbb{H}^n_K$.

The hyperbolic normalizing flow with $\mathcal{T}C$ layers discussed above operates purely in the tangent space at the origin. This simplifies the computation of the Jacobian determinant, but anchoring the flow at the origin may hinder its expressive power and its ability to leverage disparate regions of the manifold. In this section, we remedy this shortcoming with a new hyperbolic flow that performs translations between tangent spaces via parallel transport.

We term this transformation *Wrapped Hyperboloid Coupling* ($\mathcal{W}\mathbb{H}C$). As with the $\mathcal{T}C$ layer, it is a fully invertible transformation $f^{\mathcal{W}\mathbb{H}C} : \mathbb{H}^n_k \to \mathbb{H}^n_k$ with a tractable analytic form for the Jacobian determinant. To define a $\mathcal{W}\mathbb{H}C$ layer we first use the logarithmic map at the origin to transport a point to the tangent space. We employ the coupling strategy previously discussed and partition our input vector into two components: $\tilde{x}_1 := \tilde{x}_{1:d}$ and $\tilde{x}_2 := \tilde{x}_{d+1:n}$. Let $\tilde{x} = \log^K_{\mathbf{o}}(\mathbf{x})$ be the point on $\mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$ after the logarithmic map. The remainder of the $\mathcal{W}\mathbb{H}C$ layer can be defined as follows;

$$\tilde{f}^{\mathcal{W}\mathbb{H}C}(\tilde{x}) = \begin{cases} \tilde{z}_1 &= \tilde{x}_1 \\ \tilde{z}_2 &= \log^K_{\mathbf{o}}\left(\exp^K_{t(\tilde{x}_1)}\left(\mathrm{PT}_{\mathbf{o}\to t(\tilde{x}_1)}(v)\right)\right) \end{cases}$$
$$v = \tilde{x}_2 \odot \sigma(s(\tilde{x}_1))$$
$$f^{\mathcal{W}\mathbb{H}C}(\mathbf{x}) = \exp^K_{\mathbf{o}}(\tilde{f}^{\mathcal{W}\mathbb{H}C}(\log^K_{\mathbf{o}}(\mathbf{x}))). \tag{13}$$

Functions $s : \mathcal{T}_{\mathbf{o}}\mathbb{H}^d_k \to \mathcal{T}_{\mathbf{o}}\mathbb{H}^{n-d}_k$ and $t : \mathcal{T}_{\mathbf{o}}\mathbb{H}^d_k \to \mathbb{H}^n_k$ are taken to be arbitrary neural nets, but the role of t when compared to $\mathcal{T}C$ is vastly different. In particular, the generalization of translation on Riemannian manifolds can be viewed as parallel transport to a different tangent space. Consequently, in Eq. 13, the function t predicts a point on the manifold that we wish to parallel transport to. This greatly increases the flexibility as we are no longer confined to the tangent space at the origin. The logarithmic map is then used to ensure that both $\tilde{z}_1$ and $\tilde{z}_2$ are in the same tangent space before the final exponential map that projects the point to the manifold.

One important consideration in the construction of t is that it should only parallel transport functions of $\tilde{x}_2$. However, the output of t is a point on $\mathbb{H}^n_k$ and without care this can involve elements in $\tilde{x}_1$. To prevent such a scenario we construct the output of $t = [t_0, 0, \ldots, 0, t_{d+1}, \ldots, t_n]$ where elements $t_{d+1:n}$ are used to determine the value of t_0 using Eq. 5, such that it is a point on the manifold and every remaining index is set to zero. Such a construction ensures that only components of any function of $\tilde{x}_2$ are parallel transported as desired. Figure 3 illustrates the transformation performed by the $\mathcal{W}HC$ layer.

Inverse of $\mathcal{W}\mathbb{H}C$. To invert the flow it is sufficient to show that argument to the final exponential map at the origin itself is invertible. Furthermore, note that $\tilde{x}_1$ undergoes an identity mapping and is trivially invertible. Thus we need to show that the second partition is invertible, i.e. that the following transformation is invertible:

$$\tilde{z}_2 = \log^K_{\mathbf{o}}\left(\exp^K_{t(\tilde{x}_1)}\left(\mathrm{PT}_{\mathbf{o}\to t(\tilde{x}_1)}(v)\right)\right). \tag{14}$$

As discussed in Section 2, the parallel transport, exponential map, and logarithmic map all have well-defined inverses with closed forms. Thus, the overall transformation is invertible in closed form:

$$\begin{cases} \tilde{x}_1 &= \tilde{z}_1 \\ \tilde{x}_2 &= \left(\mathrm{PT}_{t(\tilde{z}_1)\to\mathbf{o}}(\log^K_{t(\tilde{z}_1)}(\exp^K_{\mathbf{o}}(\tilde{z}_2))\right) \odot \sigma(s(\tilde{z}_1))^{-1} \end{cases}$$

Properties of $\mathcal{W}\mathbb{H}C$. To compute the Jacobian determinant of the full transformation in Eq. 13 we proceed by analyzing the effect of $\mathcal{W}\mathbb{H}C$ on valid orthonormal bases w.r.t. the Lorentz inner product for the tangent space at the origin. We state our main result here and provide a sketch of the proof, while the entire proof can be found in Appendix D.

Proposition 2. *The Jacobian determinant of the function $\tilde{f}^{\mathcal{WHIC}}$ in equation 13 is:*

$$\left|\det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)\right| = \prod_{i=d+1}^{n} \sigma(s(\tilde{x}_1))_i \times \left(\frac{R\sinh(\frac{||q||_{\mathcal{L}}}{R})}{||q||_{\mathcal{L}}}\right)^{l} \times \left(\frac{R\sinh(\frac{||\log_{\boldsymbol{o}}^K(\hat{\boldsymbol{q}})||_{\mathcal{L}}}{R})}{||\log_{\boldsymbol{o}}^K(q)||_{\mathcal{L}}}\right)^{-l} \times \left(\frac{R\sinh(\frac{||\tilde{z}||_{\mathcal{L}}}{R})}{||\tilde{z}||_{\mathcal{L}}}\right)^{n-1} \times \left(\frac{R\sinh(\frac{||\log_{\boldsymbol{o}}^K(\boldsymbol{x})||_{\mathcal{L}}}{R})}{||\log_{\boldsymbol{o}}^K(\boldsymbol{x})||_{\mathcal{L}}}\right)^{1-n}, \quad (15)$$

where $\tilde{z} = \text{concat}(\tilde{z}_1, \tilde{z}_2)$, the constant $l = n - d$, σ is a non-linearity, $q = \text{PT}_{\boldsymbol{o}\to t(\tilde{x}_1)}(v)$ and $\hat{\boldsymbol{q}} = \exp_{\boldsymbol{t}}^K(q)$.

Proof Sketch. We first note that the exponential and logarithmic maps applied at the beginning and end of the $\mathcal{WHIC}$ can be dealt with by appealing to the chain rule and the known Jacobian determinants for these functions as used in Proposition 1. Thus, what remains is the following term: $\left|\det\left(\frac{\partial z}{\partial \tilde{x}}\right)\right|$. To evaluate this term we rely on the following Lemma.

Lemma 1. *Let $h : \mathcal{T}_{\mathbf{o}}\mathbb{H}_k^n \to \mathcal{T}_{\mathbf{o}}\mathbb{H}_k^n$ be a function defined as:*

$$h(\tilde{x}) = z = \text{concat}(\tilde{z}_1, \tilde{z}_2). \quad (16)$$

Now, define a function $h^* : \mathcal{T}_{\mathbf{o}}\mathbb{H}^{n-d} \to \mathcal{T}_{\mathbf{o}}\mathbb{H}^{n-d}$ which acts on the subspace of $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{n-d}$ corresponding to the standard basis elements $e_{d+1}, ..., e_n$ as

$$h^*(\tilde{x}_2) = \log_{\mathbf{o}_2}^K\left(\exp_{\mathbf{t}_2}^K\left(\text{PT}_{\mathbf{o}_2\to\mathbf{t}_2}(v)\right)\right), \quad (17)$$

where $\tilde{x}_2$ denotes the portion of the vector $\tilde{x}$ corresponding to the standard basis elements $e_{d+1}, ..., e_n$ and s and t are constants (which depend on $\tilde{x}_1$). In Equation equation 17, we use $\mathbf{o}_2 \in \mathbb{H}^{n-d}$ to denote the vector corresponding to only the dimensions $d+1, ..., n$ and similarly for $\mathbf{t}_2$. Then we have that

$$\left|\det\left(\frac{\partial z}{\partial \tilde{x}}\right)\right| = \left|\det\left(\frac{\partial h^*(\tilde{x}_{d+1:n})}{\partial \tilde{x}_{d+1:n}}\right)\right|. \quad (18)$$

The proof for Lemma 1 is provided in Appendix D. Using Lemma 1, and the fact that $|\det(\text{PT}_{\mathbf{u}\to\mathbf{t}}(v))| = 1$ (Nagano et al., 2019) we are left with another composition of functions but on the subspace $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{n-d}$. The Jacobian determinant for these functions, are simply that of the logarithmic map, exponential map and the argument to the parallel transport which can be easily computed as $\prod_{i=d+1}^{n} \sigma(s(\tilde{x}_1))$. □

The cost of computing the change in volume for one $\mathcal{WHIC}$ layer is $\mathcal{O}(n)$ which is the same as a $\mathcal{TC}$ layer plus the added cost of the two new maps that operate on the lower subspace of basis elements.

4. Experiments

We evaluate our $\mathcal{TC}$-flow and $\mathcal{WHIC}$-flow on three tasks: structured density estimation, graph reconstruction, and graph generation.[2] Throughout our experiments, we rely on three main baselines. In Euclidean space, we use Gaussian latent variables and affine coupling flows (Dinh et al., 2017), denoted $\mathcal{N}$ and $\mathcal{NC}$, respectively. In the Lorentz model, we use Wrapped Normal latent variables as an analogous baseline (Nagano et al., 2019). Since all model parameters are defined on tangent spaces, models can be trained with conventional optimizers like Adam (Kingma & Ba, 2014). Following previous work, we also consider the curvature K as a learnable parameter and we clamp the max norm of vectors to 40 before any logarithmic or exponential map (Skopek et al., 2019). Appendix E contains details on model architectures and implementation concerns.

4.1. Structured Density Estimation

We first consider structured density estimation in a canonical VAE setting (Kingma & Welling, 2013), where we seek to learn rich approximate posteriors using normalizing flows and evaluate the marginal log-likelihood of test data. Following work on hyperbolic VAEs, we test the approaches on a branching diffusion process (BDP) and dynamically binarized MNIST (Mathieu et al., 2019; Skopek et al., 2019).

Our results are shown in Tables 1 and 2. On both datasets we observe our hyperbolic flows provide significant improvements when using latent spaces of low dimension. This result matches theoretical expectations—*e.g.*, that trees can be perfectly embedded in $\mathbb{H}_K^2$—and dovetails with previous work on graph embedding (Nickel & Kiela, 2017). This highlights that the benefit of leveraging hyperbolic space is most prominent in small dimensions. However, as we increase the latent dimension, the Euclidean approaches can compensate for this intrinsic geometric limitation.

Model	BDP-2	BDP-4	BDP-6
$\mathcal{N}$-VAE	$-55.4_{\pm 0.2}$	$-55.2_{\pm 0.3}$	$-56.1_{\pm 0.2}$
$\mathbb{H}$-VAE	$-54.9_{\pm 0.3}$	$-55.4_{\pm 0.2}$	$-58.0_{\pm 0.2}$
$\mathcal{NC}$	$-55.4_{\pm 0.4}$	**-54.7**$_{\pm 0.1}$	**-55.2**$_{\pm 0.3}$
$\mathcal{TC}$	$-54.9_{\pm 0.1}$	$-55.6_{\pm 0.2}$	$-57.5_{\pm 0.2}$
$\mathcal{WHIC}$	**-52.8**$_{\pm 0.3}$	$-55.2_{\pm 0.2}$	$-57.4_{\pm 0.3}$

Table 1. Test Log Likelihood on Binary Diffusion Process versus latent dimension. All normalizing flows use 2-coupling layers.

4.2. Graph Reconstruction

We evaluate the practical utility of our hyperbolic flows by conducting experiments on the task of link prediction using graph neural networks (GNNs) (Scarselli et al., 2008) as an

[2] Code is included with the submission and will be released.

Model	MNIST 2	MNIST 4	MNIST 6
$\mathcal{N}$-VAE	$-139.5_{\pm 1.0}$	$-115.6_{\pm 0.2}$	$-100.0_{\pm 0.02}$
$\mathbb{H}$-VAE	*	$-113.68_{\pm 0.9}$	$-99.8_{\pm 0.2}$
$\mathcal{NC}$	$-139.2_{\pm 0.4}$	$-115.2_{\pm 0.6}$	$\mathbf{-98.7}_{0.3}$
$\mathcal{TC}$	*	$\mathbf{-112.5}_{\pm 0.2}$	$-99.3_{\pm 0.2}$
$\mathcal{WHC}$	$\mathbf{-136.5}_{\pm 2.1}$	$-112.8_{\pm 0.5}$	$-99.4_{\pm 0.2}$

Table 2. Test Log Likelihood on MNIST averaged over 5 runs verus latent dimension. * indicates numerically unstable settings.

inference model. Given a simple graph $\mathcal{G} = (\mathcal{V}, A, X)$, defined by a set of nodes $\mathcal{V}$, an adjacency matrix $A \in \mathbb{Z}^{|\mathcal{V}|\times|\mathcal{V}|}$ and node feature matrix $X \in \mathbb{R}^{|\mathcal{V}|\times n}$, we learn a VGAE (Kipf & Welling, 2016) model whose inference network, q_ϕ, defines a distribution over node embeddings $q_\phi(Z|A, X)$. To score the likelihood of an edge existing between pairs of nodes we use an inner product decoder: $p(A_{u,v} = 1|z_u, z_v) = \sigma(z_u^T z_v)$ (with dot products computed in $\mathcal{T}_{\mathbf{0}}\mathbb{H}^n_K$ when necessary). Given these components, the inference GNNs are trained to maximize the variational lower bound on a training set of edges.

We use two different disease datasets taken from (Chami et al., 2019) and (Mathieu et al., 2019)[3] for evaluation purposes. The first dataset Diseases-I is composed of a network of disorders and disease genes linked by the known disordergene associations (Goh et al., 2007). In the second dataset Diseases-II, we build tree networks of a SIR disease spreading model (Anderson et al., 1992), where node features determine the susceptibility to the disease. In Table 3 we report the AUC and average precision (AP) on the test set. We observe consistent improvements when using hyperbolic $\mathcal{WHC}$ flow. Similar to the structured density estimation setting, the performance gains of $\mathcal{WHC}$ are best observed in low-dimensional latent spaces.

Model	Dis-I AUC	Dis-I AP	Dis-II AUC	Dis-II AP
$\mathcal{N}$-VAE	$0.90_{\pm 0.01}$	$0.92_{\pm 0.01}$	$0.92_{\pm 0.01}$	$0.91_{\pm 0.01}$
$\mathbb{H}$-VAE	$0.91_{\pm 5e\text{-}3}$	$0.92_{\pm 5e\text{-}3}$	$0.92_{\pm 4e\text{-}3}$	$0.91_{\pm 0.01}$
$\mathcal{NC}$	$0.92_{\pm 0.01}$	$0.93_{\pm 0.01}$	$0.95_{\pm 4e\text{-}3}$	$0.93_{\pm 0.01}$
$\mathcal{TC}$	$\mathbf{0.93}_{\pm 0.01}$	$0.93_{\pm 0.01}$	$\mathbf{0.96}_{\pm 0.01}$	$0.95_{\pm 0.01}$
$\mathcal{WHC}$	$\mathbf{0.93}_{\pm 0.01}$	$\mathbf{0.94}_{\pm 0.01}$	$\mathbf{0.96}_{\pm 0.01}$	$\mathbf{0.96}_{\pm 0.01}$

Table 3. Test AUC and Test AP on Graph Embeddings where Dis-I has latent dimesion 6 and Dis-II has latent dimension 2.

4.3. Graph Generation

Finally, we explore the utility of our hyperbolic flows for generating hierarchical structures. As a synthetic testbed, we construct datasets containing uniformly random trees as well as uniformly random lobster graphs (Golomb, 1996), where each graph contains between 20 to 100 nodes. We

[3]We uncovered issues with the two remaining datasets in (Mathieu et al., 2019) and thus omit them (Appendix F).

Model	Accuracy	Avg. Clust.	Avg. GC.
$\mathcal{NC}$	$56.6_{\pm 5.5}$	$40.9_{\pm 42.7}$	$0.34_{\pm 0.10}$
$\mathcal{TC}$	$32.1_{\pm 1.9}$	$98.3_{\pm 89.5}$	$0.25_{\pm 0.12}$
$\mathcal{WHC}$	$\mathbf{62.1}_{\pm 10.9}$	$\mathbf{21.1}_{\pm 13.4}$	$\mathbf{0.13}_{\pm 0.07}$

Table 4. Generation statistics on random trees over 5 runs.

then train a generative model to learn the distribution of these graphs. We expect the hyperbolic flows to provide a significant benefit for generating valid random trees, as well as learning the distribution of lobster graphs, which are a special subset of trees.

We follow the two-stage training procedure outlined in Graph Normalizing Flows (Liu et al., 2019a) in that we first train an autoencoder to give node-level latents on which we train an NF for density estimation. Empirically, we find that using GRevNets (Liu et al., 2019a) and defining edge probabilities using a distance-based decoder consistently leads to better generation performance. Thus we define edge probabilities as $p(A_{u,v} = 1|z_u, z_v) = \sigma((-d_{\mathcal{G}}(u, v) - b)/\tau)$ where b and τ are learned edge specific bias and temperature parameters. At inference time, we first sample the number of nodes to generate from the empirical distribution of the dataset. We then independently sample node latents from our prior, beginning with a fully connected graph, and then push these samples through our learned flow to give refined edge probabilities.

To evaluate the various approaches, we construct 100 training graphs for each dataset to train our model. Figure 4 shows representative samples generated by the various approaches. We see that hyperbolic normalizing flows learn to generate tree-like graphs and also match the specific properties of the lobster graph distribution, whereas the Euclidean flow model tends to generate densely connected graphs with many cycles (or else disconnected graphs). To quantify these intuitions, Table 4 contains statistics on how often the different models generate valid trees (denoted by "accuracy"), as well as the average number of triangles and the average global clustering coefficients for the generated graphs. Since the target data is random trees, a perfect model would achieve 100% accuracy, with no triangles, and a global clustering of 0 for all graphs. We see that the hyperbolic models generate valid trees more often, and they generate graphs with fewer triangles and lower clustering on average. Finally, to evaluate how well the models match the specific properties of the lobster graphs, we follow Liao et al. (2019) and report the MMD distance between the generated graphs and a test set for various graph statistics (Figure 5). Again, we see that the hyperbolic approaches significantly outperform the Euclidean normalizing flow.

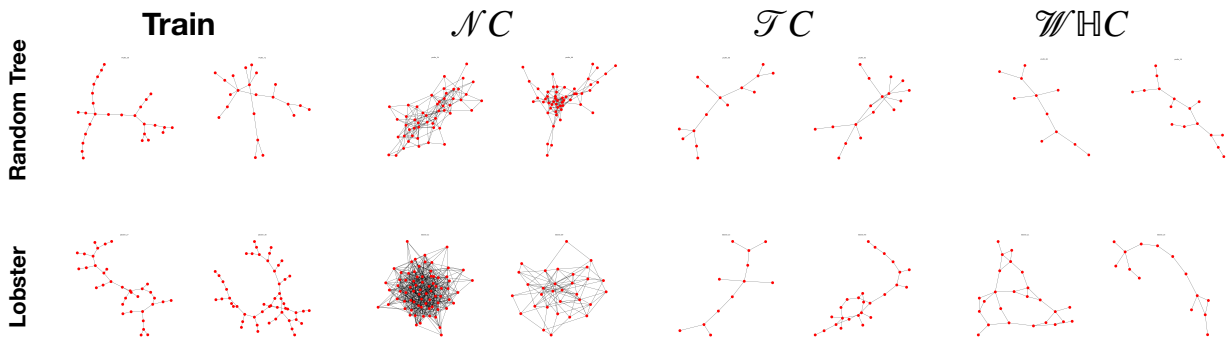

Figure 4. Selected qualitative results on graph generation for lobster and random tree graph.

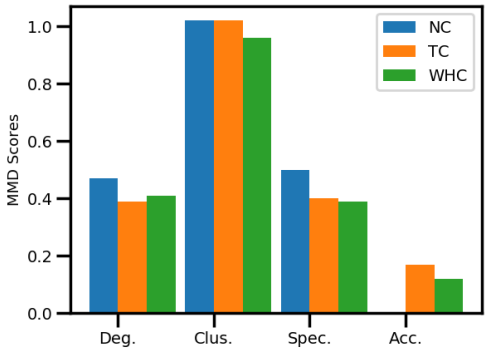

Figure 5. MMD scores for graph generation on Lobster graphs. Note, that $\mathcal{NC}$ achieves 0% accuracy.

5. Related Work

Hyperbolic Geometry in Machine Learning:. The intersection of hyperbolic geometry and machine learning has recently risen to prominence (Dhingra et al., 2018; Tay et al., 2018; Law et al., 2019; Khrulkov et al., 2019; Ovinnikov, 2019). Early prior work proposed to embed data into the Poincaré ball model (Nickel & Kiela, 2017; Chamberlain et al., 2017). The equivalent Lorentz model was later shown to have better numerical stability properties (Nickel & Kiela, 2018), and recent work has leveraged even more stable tiling approaches (Yu & De Sa, 2019). Recent works have extended several conventional deep learning modules (e.g., dense neural network layers and GNN architectures) to hyperbolic space (Gulcehre et al., 2018; Ganea et al., 2018; Chami et al., 2019). Latent variable models on hyperbolic space have also been investigated in the context of VAEs, using generalizations of the normal distribution (Nagano et al., 2019; Mathieu et al., 2019). In contrast, our work learns a flexible approximate posterior using a novel normalizing flow designed to use the geometric structure of hyperbolic spaces. In addition to work on hyperbolic VAEs, there are also several works that explore other non-Euclidean spaces (e.g., spherical VAEs) (Davidson et al., 2018; Falorsi et al., 2019; Grattarola et al., 2019).

Normalizing Flows:. Normalizing flows (NFs) (Rezende & Mohamed, 2015; Dinh et al., 2017) are a class of probabilistic models which use invertible transformations to map samples from a simple base distribution to samples from a more complex learned distribution.While there are many classes of normalizing flows, see survey (Papamakarios et al., 2019; Kobyzev et al., 2019), our work largely follows flows designed with partial ordered dependency as found in affine coupling transformations (Dinh et al., 2017). Recently, normalizing flows have also been extended to Riemannian manifolds such as spherical spaces in Gemici et al. (2016). In a different line of work authors in Wang & Wang (2019) construct a planar flow (Rezende & Mohamed, 2015) on Riemannian manifolds parameterized by the inverse multiquadratics kernel function.

6. Conclusion

In this paper, we introduce two novel normalizing flows on hyperbolic spaces. We show that our flows are efficient to sample from, easy to invert and require only $\mathcal{O}(n)$ cost to compute the change in volume. We demonstrate the effectiveness of constructing hyperbolic normalizing flows for latent variable modeling of hierarchical data. We empirically observe improvements in structured density estimation, graph reconstruction and also generative modeling of tree-structured data, with large qualitative improvements in generated sample quality compared to Euclidean methods. One important limitation is in the numerical error introduced by clamping operations which prevent the creation of deep flow architectures. We hypothesize that this is an inherent limitation of the Lorentz model, which may be alleviated with newer models of hyperbolic geometry that use integer-based tiling (Yu & De Sa, 2019). In addition, while we considered hyperbolic generalizations of the coupling transforms to define our normalizing flows, designing new classes of invertible transformations like autoregressive and residual flows on non-Euclidean spaces is an interesting direction for future work.

Acknowledgements

Funding: AJB is supported by an IVADO Excellence Fellowship. RL was supported by Connaught International Scholarship and RBC Fellowship. WLH is supported by a Canada CIFAR AI Chair. This work was also supported by NSERC Discovery Grants held by WLH and PP. In addition the authors would like to thank Chinwei Huang, Maxime Wabartha, Andre Cianflone and Andrea Madotto for helpful feedback on earlier drafts of this work and Kevin Luk, Laurent Dinh and Niky Kamran for helpful technical discussions.

References

Anderson, R. M., Anderson, B., and May, R. M. *Infectious diseases of humans: dynamics and control*. Oxford university press, 1992.

Chamberlain, B. P., Clough, J., and Deisenroth, M. P. Neural embeddings of graphs in hyperbolic space. *arXiv preprint arXiv:1705.10359*, 2017.

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.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*, 2018.

Dhingra, B., Shallue, C. J., Norouzi, M., Dai, A. M., and Dahl, G. E. Embedding text in hyperbolic spaces. *arXiv preprint arXiv:1806.04313*, 2018.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. In *The 5th International Conference on Learning Representations (ICLR), Vancouver*, 2017.

Falorsi, L., de Haan, P., Davidson, T. R., and Forré, P. Reparameterizing distributions on lie groups. *arXiv preprint arXiv:1903.02958*, 2019.

Ganea, O., Bécigneul, G., and Hofmann, T. Hyperbolic neural networks. In *Advances in neural information processing systems*, pp. 5345–5355, 2018.

Gemici, M. C., Rezende, D., and Mohamed, S. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

Goh, K.-I., Cusick, M. E., Valle, D., Childs, B., Vidal, M., and Barabási, A.-L. The human disease network. *Proceedings of the National Academy of Sciences*, 104 (21):8685–8690, 2007.

Golomb, S. W. *Polyominoes: puzzles, patterns, problems, and packings*, volume 16. Princeton University Press, 1996.

Grathwohl, W., Chen, R. T., Bettencourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Grattarola, D., Livi, L., and Alippi, C. Adversarial autoencoders with constant-curvature latent manifolds. *Applied Soft Computing*, 81:105511, 2019.

Gulcehre, C., Denil, M., Malinowski, M., Razavi, A., Pascanu, R., Hermann, K. M., Battaglia, P., Bapst, V., Raposo, D., Santoro, A., et al. Hyperbolic attention networks. *arXiv preprint arXiv:1805.09786*, 2018.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. In *Proceedings of the 35th international conference on Machine learning*, 2018.

Khrulkov, V., Mirvakhabova, L., Ustinova, E., Oseledets, I., and Lempitsky, V. Hyperbolic image embeddings. *arXiv preprint arXiv:1904.02239*, 2019.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

Kobyzev, I., Prince, S., and Brubaker, M. A. Normalizing flows: Introduction and ideas. *arXiv preprint arXiv:1908.09257*, 2019.

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.

Liao, R., Li, Y., Song, Y., Wang, S., Hamilton, W., Duvenaud, D. K., Urtasun, R., and Zemel, R. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems*, pp. 4257–4267, 2019.

Liu, J., Kumar, A., Ba, J., Kiros, J., and Swersky, K. Graph normalizing flows. In *Advances in Neural Information Processing Systems*, pp. 13556–13566, 2019a.

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

Mathieu, E., Le Lan, C., Maddison, C. J., Tomioka, R., and Teh, Y. W. Continuous hierarchical representations with poincaré variational auto-encoders. In *Advances in neural information processing systems*, pp. 12544–12555, 2019.

Nagano, Y., Yamaguchi, S., Fujita, Y., and Koyama, M. A wrapped normal distribution on hyperbolic space for gradient-based learning. In *International Conference on Machine Learning*, pp. 4693–4702, 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. *arXiv preprint arXiv:1806.03417*, 2018.

Ovinnikov, I. Poincar\'e wasserstein autoencoder. *arXiv preprint arXiv:1901.01427*, 2019.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *arXiv preprint arXiv:1912.02762*, 2019.

Pennec, X. Intrinsic statistics on riemannian manifolds: Basic tools for geometric measurements. *Journal of Mathematical Imaging and Vision*, 25(1):127, 2006.

Ratcliffe, J. G. *Foundations of Hyperbolic Manifolds*. Number 149 in Graduate Texts in Mathematics. Springer-Verlag, 1994.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32nd international conference on Machine learning*. ACM, 2015.

Said, S., Bombrun, L., and Berthoumieu, Y. New riemannian priors on the univariate normal model. *Entropy*, 16 (7):4015–4031, 2014.

Sarkar, R. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pp. 355–366. Springer, 2011.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Skopek, O., Ganea, O.-E., and Bécigneul, G. Mixed-curvature variational autoencoders. *arXiv preprint arXiv:1911.08411*, 2019.

Tay, Y., Tuan, L. A., and Hui, S. C. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 583–591, 2018.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wang, P. Z. and Wang, W. Y. Riemannian normalizing flow on variational Wasserstein autoencoder for text modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 284–294, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1025. URL https://www.aclweb.org/anthology/N19-1025.

Xu, B., Wang, N., Chen, T., and Li, M. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

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.

A. Background on Riemannian Geometry

An n-dimensional manifold is a topological space that is equipped with a family of open sets U_i which cover the space and a family of functions ψ_i that are homeomorphisms between the U_i and open subsets of $\mathbb{R}$. The pairs (U_i, ψ_i) are called *charts*. A crucial requirement is that if two open sets U_i and U_j intersect in a region, call it U_{ij}, then the composite map $\psi_i \circ \psi_j^{-1}$ restricted to U_{ij} is infinitely differentiable. If $\mathcal{M}$ is an $n-$dimensional manifold then a chart, $\psi : U \to V$, on $\mathcal{M}$ maps an open subset U to an open subset $V \subset \mathbb{R}^n$. Furthermore, the image of the point $p \in U$, denoted $\psi(p) : \mathbb{R}^n$ is termed the local coordinates of p on the chart ψ. Examples of manifolds include $\mathbb{R}^n$, the Hypersphere $\mathbb{S}^n$, the Hyperboloid $\mathbb{H}^n$, a torus. In this paper we take an extrinsic view of the geometry, that is to say a manifold can be thought of as being embedded in a higher dimensional Euclidean space, —i.e. $\mathcal{M}^n \subset \mathbb{R}^{n+1}$, and inherits the coordinate system of the ambient space. This is not how the subject is usually developed but for spaces of constant curvature one gets convenient formulas.

Tangent Spaces. Let $p \in \mathcal{M}$ be a point on an $n-$dimensional smooth manifold and let $\gamma(t) \to \mathcal{M}$ be a differentiable parametric curve with parameter $t \in [-\epsilon, \epsilon]$ passing through the point such that $\gamma(0) = p$. Since $\mathcal{M}$ is a smooth manifold we can trace the curve in local coordinates via a chart ψ and the entire curve is given in local coordinates by $x = \psi \circ \gamma(t)$. The tangent vector to this curve at p is then simply $v = (\psi \circ \gamma)'(0)$. Another interpretation of the tangent vector of γ is by interpreting the point p as a position vector and the tangent vector is then interpreted as the velocity vector at that point. Using this definition the set of all tangent vectors at p is denoted as $\mathcal{T}_p\mathcal{M}$, and is called the tangent space at p.

Riemannian Manifold. A Riemannian metric tensor g on a smooth manifold $\mathcal{M}$ is defined as a family of inner products such that at each point $p \in \mathcal{M}$ the inner product takes vectors from the tangent space at p, $g_p = \langle \cdot, \cdot \rangle_p : \mathcal{T}_p\mathcal{M} \times \mathcal{T}_p\mathcal{M} \to \mathbb{R}$. This means g is defined for every point on $\mathcal{M}$ and varies smoothly. Locally, g can be defined using the basis vectors of the tangent space $g_{ij}(p) = g(\frac{\partial}{\partial p_i}, \frac{\partial}{\partial p_j})$. In matrix form the Riemannian metric, $G(p)$, can be expressed as, $\forall u, v \in \mathcal{T}_p\mathcal{M} \times \mathcal{T}_p\mathcal{M}, \langle u, v \rangle_p = g(p)(u, v) = u^T G(p) v$. A smooth manifold manifold $\mathcal{M}$ which is equipped with a Riemannian metric at every point $p \in \mathcal{M}$ is called a Riemannian manifold. Thus every Riemannian manifold is specified as the tuple $(\mathcal{M}, g)$ which define the smooth manifold and its associated Riemannian metric tensor.

Armed with a Riemannian manifold we can now recover some conventional geometric insights such as the length of a parametric curve γ, the distance between two points on the manifold, local notion of angle, surface area and volume. We define the length of a curve, $L[\gamma] = \int_a^b g_{\gamma(t)} ||\gamma'(t)|| dt$. This definition is very similar to the length of a curve on Euclidean spaces if we just observe that the Riemannian metric is I_n. Now turning to the distance between points p and q we can reason that it must be the smallest or distance minimizing parametric curve between the points which in the literature are known as *geodesics*[4]. Stated another way: $d(p, q) = \inf \left\{ L[\gamma] \,|\gamma : [a, b] \to \mathcal{M} \right\}$ with , $\gamma(a) = p$ and $\gamma(b) = q$. A norm is induced on every tangent space by g_p and is defined as $\mathcal{T}_p\mathcal{M} : ||\cdot||_p : \sqrt{\langle \cdot, \cdot \rangle_p}$. Finally, we can also define an infitisimal volume element on each tangent space and as a result measure $d\mathcal{M}(p) = \sqrt{|G(p)|} dp$, with dp being the Lebesgue measure.

B. Background Normalizing Flows

Given a parametrized density on $\mathbb{R}^n$ a *normalizing flow* defines a sequence of invertible transformations to a more complex density over the same space via the change of variable formula for probability distributions (Rezende & Mohamed, 2015). Starting from a sample from a base distribution, $z_0 \sim p(z)$, a mapping $f : \mathbb{R}^d \to \mathbb{R}^d$, with parameters θ that is both invertible and smooth, the log density of $z' = f(z_0)$ is defined as $\log p_\theta(z') = \log p(z_0) - \log \det \left| \frac{\partial f}{\partial z} \right|$. Where, $p_\theta(z')$ is the probability of the transformed sample and $\partial f / \partial x$ is the Jacobian of f. To construct arbitrarily complex densities a chain of functions of the same form as f can be defined and through successive application of change of density for each invertible transformation in the flow. Thus the final sample from a flow is then given by $z_j = f_j \circ f_{j-1} ... \circ f_1(z_0)$ and it's corresponding density can be determined simply by $\ln p_\theta(z_j) = \ln p(z_0) - \sum_{i=1}^{j} \ln \det \left| \frac{\partial f_i}{\partial z_{i-1}} \right|$. Of practical importance when designing normalizing flows is the cost associated with computed the log determinant of the Jacobian which is computationally expensive and can range anywhere from $O(n!) - O(n^3)$ for an arbitrary matrix and a chosen algorithm. However, through an appropriate choice of f this computation cost can be brought down significantly. While there are many different choices for the transformation function, f, in this work we consider only RealNVP based flows as presented in (Dinh et al., 2017) and (Rezende & Mohamed, 2015) due to their simplicity and expressive power in capturing complex data distributions.

[4] Actually a geodesic is usually defined as a curve such that the tangent vector is parallel transported along it. It is then a theorem that it gives the shortest path.

B.1. Variational Inference with Normalizing Flows

One obvious use case for Normalizing Flows is in learning a more expressive often multi-modal posterior distribution needed in Variational Inference. Recall that a variational approximation is a lower bound to the data log-likelihood. Take for example amortized variational inference in a VAE like setting whereby the posterior q_θ is parameterized and is amenable to gradient based optimization. The overall objective with both encoder and decoder networks:

$$\log p(x) = \log \int p(x|z)p(z)dz \tag{19}$$

$$\geq \mathbb{E}_{q_\theta(z|x)}[\log \frac{p(x,z)}{q_\theta(z|x)}] \quad \text{(Jensen's Inequality)} \tag{20}$$

$$= \mathbb{E}_{q_\theta(z|x)}[\log p(x|z)] + \mathbb{E}_{q_\theta(z|x)}\Big[\log \frac{p(z)}{q_\theta(z|x)}\Big] \tag{21}$$

$$= \mathbb{E}_{q_\theta(z|x)}[\log p(x|z)] - D_{KL}(q_\theta(z|x)||p(z)) \tag{22}$$

The tightness of the Evidence Lower Bound (ELBO) also known as the negative free energy of the system, $-\mathcal{F}(x)$, is determined by the quality of the posterior approximation to the true posterior. Thus, one way to enrich the posterior approximation is by letting q_θ be a normalizing flow itself and the resultant latent code be the output of the transformation. If we denote $q_0(z_0)$ the probability of the latent code z_0 under the base distribution and z_k as the latent code after K flow layers we may rewrite the Free Energy as follows:

$$\mathcal{F}(x) = \mathbb{E}_{q_0(z_0)}[\log q_k(z_j) - \log p(x, z_j)] \tag{23}$$

$$= \mathbb{E}_{q_0(z_0)}\Big[\log q_0(z_0) - \sum_{i=1}^{j} \ln \det \Big|\frac{\partial f_i}{\partial z_{i-1}}\Big| - \log p(x, z_i)\Big] \tag{24}$$

$$= D_{KL}(q_0(z_0)||p(z_j)) - \mathbb{E}_{q_0(z_0)}\Big[\sum_{i=1}^{j} \ln \det \Big|\frac{\partial f_i}{\partial z_{i-1}}\Big| - \log p(x|z_i)\Big] \tag{25}$$

For convenience we may take $q_0 = \mathcal{N}(\mu, \sigma^2)$ which is a reparametrized gaussian density and $p(z) = \mathcal{N}(0, I)$ a standard normal.

B.2. Euclidean RealNVP

Computing the Jacobian of functions with high-dimensional domain and codomain and computing the determinants of large matrices are in general computationally very expensive. Further complications can arise with the restriction to bijective functions make for difficult modelling of arbitrary distributions. A simple way to significantly reduce the computational burden is to design transformations such that the Jacobian matrix is triangular resulting in a determinant which is simply the product of the diagonal elements. In (Dinh et al., 2017), real valued non-volume preserving (RealNVP) transformations are introduced as simple bijections that can be stacked but yet retain the property of having the composition of transformations having a triangular determinant. To achieve this each bijection updates a part of the input vector using a function that is simple to invert, but which depends on the remainder of the input vector in a complex way. Such transformations are denoted as affine coupling layers. Formally, given a D dimensional input x and $d < D$, the output y of an affine coupling layer follows the equations:

$$y_{1:d} = x_{1:d} \tag{26}$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d}). \tag{27}$$

Where, s and t are parameterized scale and translation functions. As the second part of the input depends on the first, it is easy to see that the Jacobian given by this transformation is lower triangular. Similarly, the inverse of this transformation is given by:

$$x_{1:d} = y_{1:d} \tag{28}$$

$$x_{d+1:D} = (y_{d+1:D} - t(y_{1:d}) \odot \exp(-s(y_{1:d})). \tag{29}$$

Note that the form of the inverse does not depend on calculating the inverses of either s or t allowing them to be complex functions themselves. Further note that with this simple bijection part of the input vector is never touched which can limit the expressiveness of the model. A simple remedy to this is to simply reverse the elements that undergo scale and translation transformations prior to the next coupling layer. Such an alternating pattern ensures that each dimension of the input vector depends in a complex way given a stack of couplings allowing for more expressive models. Finally, the Jacobian of this transformation is a lower triangular matrix,

$$\frac{\partial y}{\partial x} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial y_{d+1:D}}{x_{1:d}^T} & \text{diag}(\exp s(x_{1:d})) \end{bmatrix}. \tag{30}$$

C. Change of Variable for Tangent Coupling

We now derive the change in volume formula associated with one $\mathcal{T}C$ layer. Without loss of generality we first define a binary mask which we use to partition the elements of a vector at $\mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$ into two sets. Thus b is defined as

$$b_j = \begin{cases} 1 & \text{if } j \leq d \\ 0 & \text{otherwise,} \end{cases}$$

Note that all $\mathcal{T}C$ layer operations exclude the first dimension which is always copied over by setting $b_0 = 1$ and ensures that the resulting sample always remains on $\mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$. Utilizing b we may rewrite Equation 11 as,

$$\mathbf{y} = \exp_{\mathbf{o}}^K \big(b \odot \tilde{x} + (1-b) \odot (\tilde{x} \odot \sigma(s(b \odot \tilde{x})) + t(b \odot \tilde{x}))\big), \tag{31}$$

where $\tilde{x} = \log_{\mathbf{o}}^K(x)$ is a point on the tangent space at $\mathbf{o}$. Similar to the Euclidean RealNVP, we wish to calculate the jacobian determinant of this overall transformation. We do so by first observing that the overall transformation is a valid composition of functions: $y := \exp_{\mathbf{o}}^K \circ f \circ \log_{\mathbf{o}}^K(\mathbf{x})$, where $z = f(\tilde{x})$ is the flow in tangent space. Utilizing the chain rule and the identity that the determinant of a product is the product of the determinants of its constituents we may decompose the jacobian determinant as,

$$\det\Big(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\Big) = \det\Big(\frac{\partial \exp_{\mathbf{o}}^K(z)}{\partial z}\Big) \cdot \det\Big(\frac{\partial f(\tilde{x})}{\partial \tilde{x}}\Big) \cdot \det\Big(\frac{\partial \log_{\mathbf{o}}^K(\mathbf{x})}{\partial \mathbf{x}}\Big). \tag{32}$$

Tackling each term on RHS of Eq. 32 individually, $\det\Big(\frac{\partial \exp_{\mathbf{o}}^K(z)}{\partial z}\Big) = \Big(\frac{R \sinh(\frac{||z||_{\mathcal{L}}}{R})}{||z||_{\mathcal{L}}}\Big)^{n-1}$ as derived in (Nagano et al., 2019). As the logarithmic map is the inverse of the exponential map the jacobian determinant is also the inverse —i.e. $\det\Big(\frac{\partial \log_{\mathbf{o}}^K(\mathbf{x})}{\partial \mathbf{x}}\Big) = \Big(\frac{\sinh(||\log_{\mathbf{o}}^K(\mathbf{x})||_{\mathcal{L}})}{||\log_{\mathbf{o}}^K(\mathbf{x})||_{\mathcal{L}}}\Big)^{1-n}$. For the middle term in Eq. 32 we proceed by selecting the standard basis $\{e_1, e_2, ...e_n\}$ which is an orthonormal basis with respect to the Lorentz inner product. The directional derivative with respect to a basis element e_j is computed as follows:

$$\begin{aligned} \mathrm{d}f(\tilde{x}) &= \frac{\partial}{\partial \epsilon}\Big|_{\epsilon=0} f(\tilde{x} + \epsilon e_j) \\ &= \frac{\partial}{\partial \epsilon}\Big|_{\epsilon=0} \{b \odot (\tilde{x} + \epsilon e_j) + (1-b) \odot ((\tilde{x} + \epsilon e_j) \odot \sigma(s(b \odot (\tilde{x} + \epsilon e_j))) + t(b \odot (\tilde{x} + \epsilon e_j)))\} \\ &= b \odot e_j + \frac{\partial}{\partial \epsilon}\Big|_{\epsilon=0} \{(1-b) \odot ((\tilde{x} + \epsilon e_j) \odot \sigma(s(b \odot (\tilde{x} + \epsilon e_j))) + t(b \odot (\tilde{x} + \epsilon e_j)))\} \end{aligned}$$

As $b \in [0,1]^n$ is a binary mask, it is easy to see that if $b_j = 1$ then only the first term on the RHS remains and the directional derivative with respect to e_j is simply the basis vector itself. Conversely, if $b_j = 0$ then the first term goes to zero and we are left with the second term,

$$\begin{aligned}\mathrm{d}f(\tilde{x}) &= \frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0}\{(1-b)\odot((\tilde{x}+\epsilon e_j)\odot\sigma(s(b\odot(\tilde{x}+\epsilon e_j)))+t(b\odot(\tilde{x}+\epsilon e_j)))\}\\ &= \frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0}\{(1-b)\odot((\tilde{x}+\epsilon e_j)\odot\sigma(s(b\odot\tilde{x}))+t(b\odot\tilde{x}))\}\\ &= e_j\odot\sigma(s(b\odot\tilde{x})).\end{aligned}$$

Where in the second line we've used the fact $b\odot\epsilon e_j = 0$. All together, the directional derivatives computed using our chosen basis elements are,

$$\mathrm{d}f(\tilde{x}) = (e_1, e_2, \dots e_d, e_{d+1}\odot\sigma(s(b\odot\tilde{x})), \dots e_D\odot\sigma(s(b\odot\tilde{x}))).$$

The volume factor given by this linear map is $\det(\mathrm{d}f(\tilde{x})) = \sqrt{G^TG}$, where G is the matrix of all directional derivatives. As the basis elements are orthogonal all non-diagonal entries of G^TG go to zero and the determinant is the product of the Lorentz norms of each component. As $||e_j||_{\mathcal{L}} = 1$ and $||e_j\odot\sigma(s(b\odot\tilde{x}))||_{\mathcal{L}} = ||e_j\odot\sigma(s(b\odot\tilde{x}))||_2$ for $\mathcal{T}_{\mathbf{o}}H^n_K$ the overall determinant is then $\mathrm{d}f(\tilde{x}) = \mathrm{diag}\,\sigma(s(b\odot\tilde{x}))$. Finally, the full log jacobian determinant of a $\mathcal{T}C$ layer is given by,

$$\log\det\Big(\frac{\partial\mathbf{y}}{\partial\mathbf{x}}\Big) = \Big(\frac{R\sinh(\frac{||z||_{\mathcal{L}}}{R})}{||z||_{\mathcal{L}}}\Big)^{n-1} + \sum_{i=d+1}^{n}\sigma(s(\tilde{x}_1))_i + \Big(\frac{R\sinh(\frac{||\log_{\mathbf{o}}^K(\mathbf{x})||_{\mathcal{L}}}{R})}{||\log_{\mathbf{o}}^K(\mathbf{x})||_{\mathcal{L}}}\Big)^{1-n} \tag{33}$$

Thus the overall computational cost is only slightly larger than the regular Euclidean RealNVP, $\mathcal{O}(n)$.

D. Change of Variable for Wrapped Hyperbolic Coupling

We consider the following function $f : \mathbb{H}^n_K \to \mathbb{H}^n_K$, which we use to define a normalizing flow in n-dimensional hyperbolic space (represented via the Lorentz model):

$$f(\mathbf{x}) = \exp_{\mathbf{o}}^K\left(b\odot\tilde{x} + (1-b)\odot\log_{\mathbf{o}}^K\left(\exp_{t(b\odot\tilde{x})}^K\big(\mathrm{PT}^K_{\mathbf{o}\to t(b\odot\tilde{x})}((1-b)\odot\tilde{x}\odot\sigma(s(b\odot\tilde{x})))\big)\right)\right), \tag{34}$$

where $\tilde{x} = \log_{\mathbf{o}}(\mathbf{x}) \in \mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$ is the projection of $\mathbf{x}\in\mathbb{H}^n_K$ to the tangent space at the origin, i.e, $\mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$. As in $\mathcal{T}C$ we again utilize a binary mask b so that

$$b_j = \begin{cases}1 & \text{if } j\le d\\ 0 & \text{otherwise,}\end{cases}$$

where $0 < d < n$. In Equation equation 34 the function $s : \mathcal{T}_o\mathbb{H}^d_K \to \mathcal{T}_o\mathbb{H}^{n-d}_K$ is an arbitrary function on the tangent space at the origin and σ denotes the logistic function. The function $t : \mathcal{T}_o\mathbb{H}^d_K \to \mathbb{H}^*_K \subset H^n_K$ is a map from the tangent space at the origin to a subset of hyperbolic space defined by the set of points satisfying the condition that $\mathbf{v}_i = 0, \forall i = 2...d, \mathbf{v}_i \in \mathbb{H}^n_K$ (under their representation in the Lorentz model).

Our goal is to derive the Jacobian determinant of f, i.e.,

$$\left|\det\left(\frac{\partial f(\mathbf{x})}{\partial\mathbf{x}}\right)\right|, \tag{35}$$

To do so, we will use the following facts without proof or justification:

- **Fact 1:** The chain rule for determinants, i.e., the fact that

$$\left|\det\left(\frac{\partial f(\mathbf{x})}{\partial\mathbf{x}}\right)\right| = \left|\det\left(\frac{\partial f(\mathbf{x})}{\partial\mathbf{v}}\right)\right|\left|\det\left(\frac{\partial\mathbf{v}}{\partial\mathbf{x}}\right)\right|, \tag{36}$$

  where $\mathbf{v}$ is introduced via a valid change of variables.

- **Fact 2:** The Jacobian determinant for the exponential map $\exp^K_{\mathbf{u}}(z) = \mathcal{T}_u\mathbb{H}^n_K \to \mathbb{H}^n_K$ is given by

$$\left|\det(\exp^K_{\mathbf{u}}(z))\right| = \left(\frac{R\sinh(\frac{||z||_{\mathcal{L}}}{R})}{||z||_{\mathcal{L}}}\right)^{n-1} \tag{37}$$

- **Fact 3:** The Jacobian determinant for the logarithmic map $\log^K_{\mathbf{u}}(\mathbf{v}) = \mathbb{H}^n_K \to \mathcal{T}_u\mathbb{H}^n_K$ is given by

$$\left|\det(\log^K_{\mathbf{u}}(\mathbf{v})\right| = \left(\frac{R\sinh(\frac{||\log^K_{\mathbf{o}}(\mathbf{v})||_{\mathcal{L}}}{R})}{||\log^K_{\mathbf{o}}(\mathbf{v})||_{\mathcal{L}}}\right)^{1-n} \tag{38}$$

- **Fact 4:** The Jacobian determinant for parallel transport $\mathrm{PT}^K_{\mathbf{u}\to\mathbf{t}}(v) = \mathcal{T}_{\mathbf{u}}\mathbb{H}^n_K \to \mathcal{T}_t\mathbb{H}^n_K$ is given by

$$\left|\det\left(\mathrm{PT}^K_{\mathbf{u}\to\mathbf{t}}(v)\right)\right| = 1. \tag{39}$$

Fact 2 and Fact 4 are proven in Nagano et al. (2019) "A Wrapped Normal Distribution on Hyperbolic Space for Gradient-Based Learning" for $K = -1$ and rederived for general K in Skopek et al. (2019). Fact 3 follows from the fact that the determinant of the inverse of a function is the inverse of that function's determinant. We will use similar arguments to obtain our determinant as were used in Nagano et al. (2019), and we refer the reader to Appendix A.3 in their work for background.

Our main claim is as follows

Proposition 3. *The Jacobian determinant of the function $\tilde{f}^{\mathcal{W}\mathbb{H}C}$ in equation 13 is:*

$$\left|\det\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)\right| = \prod_{i=d+1}^{n} \sigma(s(\tilde{x}_1))_i \times \left(\frac{R\sinh(\frac{||q||_{\mathcal{L}}}{R})}{||q||_{\mathcal{L}}}\right)^{l} \times \left(\frac{R\sinh(\frac{||\log^K_{\boldsymbol{o}}(\hat{\boldsymbol{q}})||_{\mathcal{L}}}{R})}{||\log^K_{\boldsymbol{o}}(q)||_{\mathcal{L}}}\right)^{-l} \times \left(\frac{R\sinh(\frac{||\tilde{z}||_{\mathcal{L}}}{R})}{||\tilde{z}||_{\mathcal{L}}}\right)^{n-1} \times \left(\frac{R\sinh(\frac{||\log^K_{\boldsymbol{o}}(\boldsymbol{x})||_{\mathcal{L}}}{R})}{||\log^K_{\boldsymbol{o}}(\boldsymbol{x})||_{\mathcal{L}}}\right)^{1-n}, \tag{40}$$

where

$$z = b \odot \tilde{x} + \log^K_{\boldsymbol{o}}\left(\exp^K_{t(b\odot\tilde{x})}\left(\mathrm{PT}^K_{\boldsymbol{o}\to t(b\odot\tilde{x})}((1-b)\odot\tilde{x}\odot\sigma(s(b\odot\tilde{x})))\right)\right)$$

the argument to the parallel transport q is,

$$q = \mathrm{PT}^K_{\boldsymbol{o}\to t(b\odot\tilde{x})}((1-b)\odot\tilde{x}\odot\sigma(s(b\odot\tilde{x}))).$$

and

$$\hat{\boldsymbol{q}} = \exp^K_{\boldsymbol{t}}(q)$$

Proof. We first note that

$$\left|\det\left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right)\right| = \left|\det\left(\frac{\partial f(\mathbf{x})}{\partial z}\right)\right| \times \left|\det\left(\frac{\partial z}{\partial \tilde{x}}\right)\right| \times \left|\det\left(\frac{\partial \tilde{x}}{\partial \mathbf{x}}\right)\right| \tag{41}$$

by the chain rule (recalling that $\tilde{x} = \log_{\mathbf{o}}(\mathbf{x})$). Now, we have that

$$\left|\det\left(\frac{\partial f(\mathbf{x})}{\partial z}\right)\right| = \left(\frac{R\sinh(\frac{||z||_{\mathcal{L}}}{R})}{||z||_{\mathcal{L}}}\right)^{n-1} \tag{42}$$

by Fact 2. And,

$$\left|\det\left(\frac{\partial \tilde{x}}{\partial \mathbf{x}}\right)\right| = \left(\frac{R\sinh(\frac{||\log^K_{\mathbf{o}}(\mathbf{x})||_{\mathcal{L}}}{R})}{||\log^K_{\mathbf{o}}(\mathbf{x})||_{\mathcal{L}}}\right)^{1-n} \tag{43}$$

by Fact 3. Thus, we are left with the term

$$\left|\det\left(\frac{\partial z}{\partial \tilde{x}}\right)\right|.$$

To evaluate this term, we rely on the following Lemma:

Lemma 2. *Let $h : \mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K \to \mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$ be a function from the tangent space at the origin to the tangent space at the origin defined as:*

$$h(\tilde{x}) = z = b \odot \tilde{x} + \log^K_{\boldsymbol{o}}\Big(\exp^K_{t(b\odot\tilde{x})}\big(\mathrm{PT}^K_{\boldsymbol{o}\to t(b\odot\tilde{x})}((1-b)\odot\tilde{x}\odot\sigma(s(b\odot\tilde{x})))\big)\Big). \tag{44}$$

Now, define a function $h^* : \mathcal{T}_{\mathbf{o}}\mathbb{H}^{n-d}_K \to \mathcal{T}_{\mathbf{o}}\mathbb{H}^{n-d}_K$ which acts on the subspace of $\mathcal{T}_{\mathbf{o}}\mathbb{H}^{n-d}_K$ corresponding to the standard basis elements $e_{d+1}, ..., e_n$ as

$$h^*(\tilde{x}_{d+1:n}) = \log^K_{\mathbf{o}_{d+1:n}}\Big(\exp^K_{t_{d+1:n}}\big(\mathrm{PT}^K_{\mathbf{o}_{d+1:n}\to t_{d+1:n}}(\tilde{x}_{d+1:n}\odot\sigma(s))\big)\Big), \tag{45}$$

where $\tilde{x}_{d+1:n}$ denotes the portion of the vector $\tilde{x}$ corresponding to the standard basis elements $e_{d+1}, ..., e_n$ and s and t are constants (which depend on $\tilde{x}_{2:d}$). In equation 45, we use $\mathbf{o}_{d+1:n} \in \mathbb{H}^{n-d}_K$ to denote the vector corresponding to only the dimensions $d+1, ..., n$ and similarily for $t_{d+1:n}$. Then we have that

$$\left|\det\left(\frac{\partial z}{\partial \tilde{x}}\right)\right| = \left|\det\left(\frac{\partial h^*(\tilde{x}_{d+1:n})}{\partial \tilde{x}_{d+1:n})}\right)\right|. \tag{46}$$

Proof. First note that by design we have that

$$[0, 0.., 0] \oplus h^*(\tilde{x}_{d+1:n}) = \log^K_{\mathbf{o}}\Big(\exp^K_{t(b\odot\tilde{x})}\big(\mathrm{PT}^K_{\mathbf{o}\to t(b\odot\tilde{x})}((1-b)\odot\tilde{x}\odot\sigma(s(b\odot\tilde{x})))\big)\Big), \tag{47}$$

i.e., the output of h^* is equal to right hand side of Equation equation 44 after prepending/concatenating 0s to the output of h^*.

Now, we can evaluate

$$\left|\det\left(\frac{\partial z}{\partial \tilde{x}}\right)\right|$$

by examining the directional derivative with respect to a set of basis elements of $\mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$. Now, given that this is the tangent space at the origin, we know that the standard (i.e., Euclidean) basis elements $e_2, ..., e_n$ form a valid basis for this subspace, since they are orthogonal under the Lorentz normal and orthogonal to the origin itself. Now, we can note first that

$$D_{e_i}h(\tilde{x}) = e_i, \forall i = 2...d. \tag{48}$$

In other words, the directional derivative for the first d basis elements is the simply the basis elements themselves. This can be verified by taking the definition of the directional derivative:

$$D_{e_i}h(\tilde{x}) = \frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0} h(\tilde{x} + \epsilon e_i) \tag{49}$$

and noting that the

$$\log^K_{\mathbf{o}}\Big(\exp^K_{t(b\odot\tilde{x})}\big(\mathrm{PT}^K_{\mathbf{o}\to t(b\odot\tilde{x})}((1-b)\odot\tilde{x}\odot\sigma(s(b\odot\tilde{x})))\big)\Big)$$

term must equal zero since $(1-b)\odot e_i = 0, \forall i = 2, ..., d$ by design. Now, for the basis elements e_i with $i > d$ we have that

$$D_{e_i}h(\tilde{x}) \perp e_j, \forall i = d+1, ..., n, j = 2..., d. \tag{50}$$

This holds because

$$\begin{aligned} D_{e_i}h(\tilde{x}) &= \frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0} h(\tilde{x} + \epsilon e_i) && (51)\\ &= \frac{\partial}{\partial\epsilon}\Big|_{\epsilon=0} \log^K_{\mathbf{o}}\Big(\exp^K_{t(b\odot\tilde{x})}\big(\mathrm{PT}^K_{\mathbf{o}\to t(b\odot\tilde{x})}((1-b)\odot\tilde{x}\odot\sigma(s(b\odot\tilde{x})))\big)\Big) && (52)\end{aligned}$$

since $b \odot e_i = 0, \forall i = d+1, ..., n$ by design and because

$$\log^K_{\mathbf{o}}\Big(\exp^K_{t(b\odot\tilde{x})}\big(\mathrm{PT}^K_{\mathbf{o}\to t(b\odot\tilde{x})}((1-b)\odot\tilde{x}\odot\sigma(s(b\odot\tilde{x})))\big)\Big) \perp e_j, \forall\tilde{x} \in \mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K, \forall j = 2..., d. \tag{53}$$

due to the $(1-b)$ term inside the parallel transport and by our design of the function t. Together, these facts give that the Jacobian matrix for h (under the basis $e_2, ..., e_n$) has the following block form:

$$\left(\frac{\partial z}{\partial \tilde{x}}\right) = \begin{bmatrix} I & \mathbf{0} \\ A & \dfrac{\partial h^*(\tilde{x}_{d+1:n})}{\partial \tilde{x}_{d+1:n})} \end{bmatrix} \tag{54}$$

and by the properties of determinants of block matrices we have that

$$\left|\det\left(\frac{\partial z}{\partial \tilde{x}}\right)\right| = \left|\det\left(\frac{\partial h^*(\tilde{x}_{d+1:n})}{\partial \tilde{x}_{d+1:n})}\right)\right| \tag{55}$$

□

Given Lemma 1, all that remains is to evaluate

$$\left|\det\left(\frac{\partial h^*(\tilde{x}_{d+1:n})}{\partial \tilde{x}_{d+1:n})}\right)\right|. \tag{56}$$

This can again be done by the chain rule, where we use Facts 2-4 to compute the determinant for exponential map, logarithmic map, and parallel transport. Finally, the Jacobian determinant for the term

$$\tilde{x} \odot \sigma(s(b \odot \tilde{x}))) \tag{57}$$

can easily be computed as $\prod_{j=d+1}^{n} \sigma(s(b \odot \tilde{x}))_j$ since the standard Euclidean basis is a basis for the tangent space at the origin as shown in Appendix B.2. □

E. Model Architectures and Hyperparameters

In this section we provide more details regarding model architectures and hyperparameters for each experiment in 4. For all hyperbolic models we used a curvature warmup for 10 epochs which aids in numerical stability Skopek et al. (2019). Specifically, we set $R = 11$ and linearly decrease to $R = 2$ every epoch after which it is treated as a learnable parameter.

Structured Density Estimation. For all VAE models our encoder consists of three linear layers. The first layer maps the input to a hidden space and the other two layers are used to paramaterize the mean and variance of the prior distribution and map samples to the latent space. The decoder for these models is simply a small MLP that consists of two linear layers that map the latent space to the hidden space and then finally back to the observation space. One important distinction between Euclidean models and hyperbolic models is that we use aFor BDP the hidden dim size is 200 while for MNIST we use 600 and the latent space is varied as shown in Tables 1 and 2. All flow models used in this setting consist of 2 linear layers each of size 128. Between each layer in either the encoder and decoder we use the LeakyRelu (Xu et al., 2015) activation function while tanh is used between flow layers. Lastly, we train all models for 80 epochs with the Adam optimizer with default setting (Kingma & Ba, 2014).

Graph Reconstruction. For graph reconstruction task we use the VGAE model as a base (Kipf & Welling, 2016) which also uses three linear layers of size 16 as the encoder in the VAE model. The decoder however is parameter less and is simply an inner product either in Euclidean space or in $\mathcal{T}_{\mathbf{o}}\mathbb{H}^n_K$ for Hyperbolic models. As the reconstruction objective contains N^2 terms we rescale the $\mathbb{D}_{KL}$ penalty by a factor of $1/N$ such that each of the losses are on the same scale. This can be understood as a $\beta-$VAE like model where $\beta = 1/N$. Like the structured density estimation setting all our flow models consist of two linear layers of size 128 with a tanh nonlinearity. Finally, we train the each model for 3000 epochs using the Adam optimizer (Kingma & Ba, 2014).

Graph Generation. For the graph generation task we adapt the training setup from (Liu et al., 2019a) in that we pretrain a graph autoencoder for 100 epochs to generate node latents. Empirically, we found that using a VGAE model for hyperbolic space worked better than a vanilla a GAE model. Furthermore, instead of using simple linear layers for the encoder we use GAT (Veličković et al., 2017) layer of size 32, which has access to the adjacency matrix. We use LeakyReLU for our encoder non-linearity while tanh is used for all flow models. Unlike GRevNets that use node features sampled from $\mathcal{N}(0, I)$ we find that it is necessary to provide the actual adjacency matrix otherwise training did not succeed. Our decoder defines edge probabilities as $p(A_{u,v} = 1|z_u, z_v) = \sigma((-d_{\mathcal{G}}(u, v) - b)/\tau)$ where b and τ are learned edge specific bias and temperature parameters implemented as one GAT layer followed by a linear layer both of size 32. Thus both the encoder and decoder are both parameterized and optimized using the Adam optimizer (Kingma & Ba, 2014).

F. Dataset Issues

Upon inspecting the code and data kindly provided by Mathieu et al. (2019) we uncovered some issues that led to us omitting their CS-PhD and Phylogenetics datasets in our comparisons. In particular, Mathieu et al. (2019) use a decoder in their cross-entropy loss that does not define a proper probability. This appears to have caused optimization issues that artificially deflated the reported performance of all the models investigated in that work. When substituting in the dot product decoder employed in this work, the accuracy of all models increases dramatically. After this change, there is no longer any benefit from employing hyperbolic spaces on these datasets. In particular, after applying this fix, the performance of the hyperbolic VAE used by Mathieu et al. (2019) falls substantially below a Euclidean VAE. Since we expect our hyperbolic flows to only give gains in cases where hyperbolic spaces provide a benefit over Euclidean spaces, these datasets do not provide a meaningful testbed for our proposed approach. Lastly, upon inspecting the code and data in Mathieu et al. (2019), we also found that the columns 1 and 2 in Table 4 of their paper appear to be swapped compared to the results generated by their code.