NEURAL SNOWFLAKES: UNIVERSAL LATENT GRAPH INFERENCE VIA TRAINABLE LATENT GEOMETRIES

Haitz Sáez de Ocáriz Borde[*]
University of Oxford
Oxford, UK
chri6704@ox.ac.uk

Anastasis Kratsios[*]
Department of Mathematics
McMaster University and the Vector Institute
Ontario, Canada
kratsioa@mcmaster.ca

ABSTRACT

The inductive bias of a graph neural network (GNN) is largely encoded in its specified graph. Latent graph inference relies on latent geometric representations to dynamically rewire or infer a GNN's graph to maximize the GNN's predictive downstream performance, but it lacks solid theoretical foundations in terms of embedding-based representation guarantees. This paper addresses this issue by introducing a trainable deep learning architecture, coined *neural snowflake*, that can adaptively implement fractal-like metrics on $\mathbb{R}^d$. We prove that any given finite weighted graph can be isometrically embedded by a standard MLP encoder, together with the metric implemented by the neural snowflake. Furthermore, when the latent graph can be represented in the feature space of a sufficiently regular kernel, we show that the combined neural snowflake and MLP encoder do not succumb to the curse of dimensionality by using only a low-degree polynomial number of parameters in the number of nodes. This implementation enables a low-dimensional isometric embedding of the latent graph. We conduct synthetic experiments to demonstrate the superior metric learning capabilities of neural snowflakes when compared to more familiar spaces like Euclidean space.

Additionally, we carry out latent graph inference experiments on graph benchmarks. Consistently, the neural snowflake model achieves predictive performance that either matches or surpasses that of the state-of-the-art latent graph inference models. Importantly, this performance improvement is achieved without requiring random search for optimal latent geometry. Instead, the neural snowflake model achieves this enhancement in a differentiable manner.

1 INTRODUCTION

Geometric deep learning (Bronstein et al., 2017; 2021) is a rapidly developing field that expands the capabilities of deep learning to encompass structured and geometric data, such as graphs, point-clouds, meshes, and manifolds. Graph neural networks (GNNs) derive their knowledge primarily from the specific graph they operate on, but many real-world problems lack an accessible ground truth graph for computation. Latent graph inference aims to address this by dynamically inferring graphs through geometric representations. Existing models lack a strong theoretical foundation and use arbitrary similarity measures for graph inference, lacking principled guidelines. A key challenge is the absence of a differentiable method to deduce geometric similarity for latent graph inference. Recently the concept of neural latent geometry search has been introduced (Sáez de Ocáriz Borde et al., 2023a), which can be formulated as follows: given a search space $\mathfrak{N}$ denoting the set of all possible latent geometries, and the objective function $L_{T,A}(g)$ which evaluates the performance of a given geometry g on a downstream task T for a machine learning model architecture A, the objective is to optimize the latent geometry: $\inf_{g\in\mathfrak{N}} L_{T,A}(g)$. In the context of latent graph inference $\mathfrak{N}$ would denote the space of possible geometric similarity measures used to construct the latent graphs. Previous studies have utilized random search to find the optimal geometry in $\mathfrak{N}$ (Kazi et al., 2022; Sáez de Ocáriz Borde et al., 2023c). However, these methods have their limitations as they

[*]Equal Contribution.

cannot infer geometry in a differentiable manner, and the representation capabilities of Riemannian manifolds are constrained by certain assumptions inherent in their geometry.

Contributions. We introduce a trainable deep learning architecture which we can adaptively implement metrics on $\mathbb{R}^d$ spaces with a fractal-like geometry, called *neural snowflakes*. We prove that together a neural snowflake and a simple MLP encoder are enough to discover any latent graph geometry. In particular, the neural snowflake implements a fractal geometry on $\mathbb{R}^d$ in which any given finite latent weighted graph can be isometrically embedded and the elementary MLP implements that embedding. We show that in cases where the latent weighted graph has favourable geometry, the neural snowflake and MLP encoder break the curse of dimensionality by only requiring a polynomial number of parameters in the graph nodes to implement the isometric embedding. We note the contrast with universal approximation theorems, e.g. Yarotsky (2017); Lu et al. (2021); Shen et al. (2022); Kratsios & Papon (2022), where the number of parameters required to implement a generic approximation depend exponentially on the dimension of the ambient space. Our embedding results exhibit no such exponential dependence on the dimension of the ambient space. We verify our theoretical guarantees experimentally with synthetic metric learning experiments and graph embedding tasks. Additionally we show that the neural snowflake and MLP encoder combination beat or match the state of the art across several latent graph inference benchmarks from the literature. This is achieved by learning the latent geometry in a differentiable manner, utilizing a single model. Thus, the neural snowflake eliminates the need to conduct costly combinatorial searches across numerous combinations of potential embedding spaces.

2 Background

Related Work. In the field of Geometric Deep Learning, most research has relied on human annotators or simple preprocessing algorithms to generate the graph structure used in GNNs. However, even when the correct graph is provided, it may not be optimal for the specific task, and the GNN could benefit from a rewiring process (Topping et al., 2021). Latent graph inference allows models to dynamically learn the intrinsic graph structure of problems where the true graph is unknown (Wang et al., 2019; Kazi et al., 2022). This is particularly relevant in real-world applications where data might only be available in the form of a pointcloud. There are several works in the literature addressing latent graph inference. In particular, we can think of graph rewiring (Arnaiz-Rodríguez et al., 2022; Bi et al., 2022; Guo et al., 2023; Topping et al., 2021) as a subset of latent graph inference in which an input graph is provided to the network, whereas latent graph inference in its most general form allows GNNs to infer a graph starting from only a pointcloud. When the underlying connectivity structure is unknown, traditional architectures like transformers (Vaswani et al., 2017) and attentional multi-agent predictive models (Hoshen, 2017) use a fully-connected graph. This assumption, however, leads to challenges when training with large graphs. Generating sparse graphs can offer computationally tractable solutions (Fetaya et al., 2018) and prevent over-smoothing (Chen et al., 2020a). Various models have been proposed to tackle this problem, starting from Dynamic Graph Convolutional Neural Networks (DGCNNs) (Wang et al., 2019), to approaches that separate graph inference and information diffusion, such as the Differentiable Graph Modules (DGMs) in Cosmo et al. (2020) and Kazi et al. (2022). Recent approaches have focused on generalizing the DGM leveraging product manifolds (Sáez de Ocáriz Borde et al., 2023c;b). Latent graph inference is also referred to as graph structure learning in the literature. A survey of similar methods can be found in Zhu et al. (2021), and some classical methods include LDS-GNN (Franceschi et al., 2019), IDGL (Chen et al., 2020b), and Pro-GNN (Jin et al., 2020). Moreover, recently generalizing latent graph inference to latent topology inference (Battiloro et al., 2023) has also been proposed.

Graphs. A weighted graph can be defined as an ordered pair $\mathcal{G} = (V, E, W)$, where V represents a set of nodes (or vertices), $E \subseteq \{\{u, v\} : u, v \in V\}$ forms the collection edges (or links) within the graph, and $W : E \to (0, \infty)$ weights the importance of each edge. An (unweighted) graph $\mathcal{G}$ is a weighted graph for which $W(\{u, v\}) = 1$ for every edge $\{u, v\} \in E$. The neighborhood $\mathcal{N}(v)$ of a node $v \in V$ is the set of nodes sharing an edge with u; i.e. $\mathcal{N}(v) \overset{\text{def.}}{=} \{u \in V : \{u, v\} \in E\}$.

Graph Neural Networks. To compute a message passing *Graph Neural Network* (GNN) layer over a graph $\mathcal{G}$ (excluding edge and graph level features for simplicity), the following equation is typically implemented: $\mathbf{x}_i^{(l+1)} = \varphi\Big(\mathbf{x}_i^{(l)}, \bigoplus_{j\in\mathcal{N}(x_i^{(l)})} \psi(\mathbf{x}_i^{(l)}, \mathbf{x}_j^{(l)})\Big)$. In the given equation, $\psi \in \mathbb{R}^d \times \mathbb{R}^d \to$

$\mathbb{R}^h$ represents a message passing function. The symbol $\bigoplus$ denotes an aggregation function, which must be permutation-invariant, e.g. the sum or max operation. Additionally, $\varphi \in \mathbb{R}^d \times \mathbb{R}^h \to \mathbb{R}^m$ represents a readout function. We note that the update equation is local and relies solely on the neighborhood of the node. Both ψ and φ can be Multi-Layer Perceptrons (MLPs). In our manuscript all MLPs will use the $\operatorname{ReLU}(t) \overset{\text{def.}}{=} \max\{0, t\}$ activation function, where $t \in \mathbb{R}$. Several special cases have resulted in the development of a wide range of GNN layers: the most well-known being Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017) and Graph Attention Networks (GATs) (Veličković et al., 2018).

Quasi-Metric Spaces. While Riemannian manifolds have been employed for formalizing non-Euclidean distances between points, their additional structural properties, such as smoothness and infinitesimal angles, impose substantial limitations, rendering the demonstration of Riemannian manifolds with closed-form distance functions challenging. Quasimetric spaces, isolate the relevant properties of Riemannian distance functions without requiring any of their additional structure for graph embedding. A *quasi-metric space* is a set X with a distance function $d : X \times X \to [0, \infty)$ satisfying for every $x, y, z \in X$: i) $d(x, y) = 0$ if and only if $x = y$, ii) $d(x, y) = d(y, x)$, iii) $d(x, y) \le C\big(d(x, z) + d(z, y)\big)$, for some constant $C \ge 1$. When $C = 1$, (X, d) is called a metric space, examples include Banach spaces and also, every (geodesic) distance on a Riemannian manifold satisfies (i)-(iii). Conversely, several statistical divergences are weaker structures than quasi-metrics since they fail (ii), and typically fail (iii); see e.g. (Hawkins et al., 2017, Proposition A.2). Property (iii) is called the *C-relaxed triangle inequality* if $C > 1$; otherwise (iii), is called the *triangle inequality*. Quasi-metric spaces typically share many of the familiar properties of metric spaces, such as similar notions of convergence, uniform-continuity of maps between quasimetric spaces, and compactness results for functions between quasimetric spaces such as Arzela-Ascoli theorems (Xia, 2009). The next example of quasimetric spaces are called metric *snowflakes*.

Example 1 (Xia (2009))**.** *Let $p > 0$ and (X, d) be a metric space. Then, (X, d^p) is a quasimetric space with $C = 2^{p-1}$ if $p > 1$. When $0 < p \le 1$, then (X, d^p) is a metric space; whence, $C = 1$.*

Snowflakes are a simple tool for constructing new (quasi) metric spaces from old ones with the following properties. Unlike products of Riemannian manifolds, a snowflake's geometry can be completely different than the original untransformed space's geometry. Unlike classical methods for constructing new distances from old ones, e.g. as warped products in differential geometry (Chen, 1999; Alexander & Bishop, 1998), snowflakes admit simple *closed-form* distance functions.

Proposition 1 (Snowflakes are Metric Spaces - (Weaver, 2018, Proposition 2.50))**.** *Let $f : [0, \infty) \to [0, \infty)$ be a continuous, concave, monotonically increasing function with $f(0) = 0$, and let (X, d) be a metric space; then, $d_f : X \times X \to \mathbb{R}$ is a metric on X $d_f(x, z) \overset{\text{def.}}{=} f(d(x, z))$, for any $x, z \in X$.*

3 Adaptive Geometries via Neural Snowflakes

We overcome one of the main challenges in contemporary Geometric Deep Learning, namely the problem of discovering a latent graph which maximizes the performance of a downstream GNN by searching a catalogue of combinations of products of elementary geometries (Gu et al., 2019); in an attempt to identify which product geometry the latent graph can be best embedded in (Sáez de Ocáriz Borde et al., 2023b;c). The major computational hurdle with these methods is that they pose a *non-differentiable* combinatorial optimization problem with a non-convex objective, making them computationally challenging to scale. Therefore, by designing a class of metrics which are differentiable in their parameters, we can instead discover which geometry best suits a learning task using backpropagation. Core to this is a trainable metric on $\mathbb{R}^d$, defined for any $x, y \in \mathbb{R}^d$ by

$$\|x - y\|_{\sigma_{\alpha,\beta,\gamma,p,C}} \overset{\text{def.}}{=} \Big(\underbrace{C_1 \left(1 - e^{-\gamma \|x-y\|}\right)}_{\text{Bounded}} + \underbrace{C_2 \, \|x - y\|^{\alpha}}_{\text{Fractal}} + \underbrace{C_3 \, \log(1 + \|x - y\|)^{\beta}}_{\text{Irregular Fractal}} \Big)^{1+|p|} \qquad (1)$$

where $0 < \alpha, \beta \le 1$, $0 \le p$, $0 \le C_1, C_2, C_3, \gamma$ not all of which are 0 and $C = (C_i)_{i=1}^3$. The trainable metric in equation 1, coined the *snowflake activation*, is the combination of three components, a *bounded geometry*, a *fractal geometry*, and an *irregular fractal* geometry part; as labeled therein. The first *bounded geometry* can adapt to latent geometries which are bounded akin to spheres, the second *fractal geometry* component can implement any classical snowflake as in Example 1 (where

$0 < p \leq 1$) and the *irregular fractal* adapts to latent geometries much more irregular where the distance between nearby points grows logarithmically at large scales and exponentially at small scales[1]. By Proposition 1, if $p = 0$, the distance in equation 1 is a metric on $\mathbb{R}^d$. For $p > 0$ $(\mathbb{R}^d, \|\cdot\|_{\sigma_{\alpha,\beta,\gamma,p,C}})$ is a quasi-metric space with 2^{p-1}-relaxed triangle inequality, by Example 1.

3.1 Neural Snowflakes

We leverage the expressiveness of deep learning, by extending the trainable distance function equation 1 to a deep neural network generating distances on $\mathbb{R}^d$, called the *neural snowflake*.

We begin by rewriting equation 1 as a trainable activation function $\sigma_{a,b} : \mathbb{R} \to [0, \infty)$ which sends on any vector $u \in \mathbb{R}^J$, for $J \in \mathbb{N}_+$, to the $J \times 3$ matrix $\sigma_{a,b}(u)$ whose j^{th} row is

$$\sigma_{a,b}(u)_j \overset{\text{def.}}{=} \big(1 - e^{-|u_j|}, |u_j|^a, \log(1 + |u_j|)^b\big). \tag{2}$$

The parameters $0 < a$ and $0 \leq b \leq 1$ are trainable.

We introduce a neural network architecture leveraging the "tensorized" snowflake activation function in equation 2, which can adaptively perturb any metric. To ensure that the neural network model always preserves the metric structure of its input metric, typically the Euclidean metric on $\mathbb{R}^d$, we must constrain the weighs of the hidden layers to ensure that the model satisfies the conditions of Proposition 1. Building on the insights of monotone (Daniels & Velikova, 2010), "input convex" (Amos et al., 2017) neural network architectures, and monotone-value (Weissteiner et al., 2022) neural networks, we simply require that all hidden weights are non-negative and do not all vanish. Lastly, the final layer of our neural snowflake model raises the generated metric to the $(1 + |p|)^{th}$ power as in equation 1. This allows the neural snowflake to leverage the flexibility of quasi-metrics, whenever suitable. They key point here is that by only doing so on the final layer, we can explicitly track the relaxation of the triangle inequality discovered while training. That is, as in Example 1, $C = 2^{p-1}$ if $p > 1$ and $C = 1$ otherwise. Putting it all together, a *neural snowflake* is a map $f : [0, \infty) \to [0, \infty)$, with iterative representation

$$\begin{aligned} f(t) &= t_I^{1+|p|} \\ t_i &= B^{(i)}\, \sigma_{a_i,b_i}(A^{(i)} t_{i-1})\, C^{(i)} \qquad \text{for } i = 1, \dots, I \\ t_0 &= t \end{aligned} \tag{3}$$

where for $i = 1, \dots, I$, $A^{(i)}$ is a $\tilde{d}_i \times d_{i-1}$ matrix, $B^{(i)}$ is a $d_i \times \tilde{d}_i$-matrix, and $C^{(i)}$ is a 3×1 matrix all of which have non-negative weights and at-least one non-zero weight, $p \in \mathbb{R}$; furthermore, for $i = 1, \dots, I$, $0 < a_i \leq 1$, $0 \leq b_i \leq 1$, $d_1, \dots, d_I \in \mathbb{N}_+$, and $d_0 = 1 = d_I$. We will always treat the neural snowflake as synonymous with the trainable distance function $\|x - y\|_f \overset{\text{def.}}{=} f(\|x - y\|)$, where $x, y \in \mathbb{R}^d$ for some contextually fixed d and f is as in equation 3.

4 Inferability Guarantees

This section contains the theoretical guarantees underpinning the neural snowflake graph inference model. We first show that it is universal, in the sense of graph representation learning, which we formalize. We then derive a series of qualitative guarantees showing that the neural snowflake graph inference model requires very few parameters to infer any latent weighted graph. In particular, neural snowflakes require a computationally feasible number of parameters to be guaranteed to work.

4.1 Universal Graph Embedding

Many graph inference pipelines depend on preserving geometry representations or encodings within latent geometries when inferring the existence of an edge between any two points (nodes) in a point cloud. Therefore, the effectiveness of any algorithm in this family of encoders hinges on its capacity to accurately or approximately represent the geometry of the latent graph. In this work, we demonstrate that the neural snowflake can infer any latent graph in this way. Thus, we formalize what

[1]Note that $\log(1 + \|x - y\|) \approx 1 - e^{-\|x-y\|}$ when $0 \approx \|x - y\|$.

it means for a *graph inference model* to be able to represent any latent (weighted) graph structure in $\mathbb{R}^D$ based on a class of geometries $\mathfrak{R}$. For any $D \in \mathbb{N}_+$, we call a pair $(\mathfrak{E}, \mathfrak{R})$ a *graph inference model*, on $\mathbb{R}^D$, if $\mathfrak{R}$ is a family of quasi-metric spaces, and $\mathfrak{E}$ is a family of maps with domain $\mathbb{R}^D$ and codomain in some member $(\mathcal{R}, d_{\mathcal{R}})$ of $\mathfrak{R}$. Whenever $\mathbb{R}^D$ is clear from the context, we do not explicitly mention it.

4.1.1 Universal Riemannian Representation Is Impossible

Our primary qualitative guarantee asserts the universality of the graph inference model $(\mathfrak{E}, \mathfrak{R})$, where $\mathfrak{E}$ represents the set of MLPs into $\mathbb{R}^d$ with ReLU activation functions, and $\mathfrak{R}$ comprises all $(\mathbb{R}^d, |\cdot|_f)$, where f is a neural snowflake; for integers $d \in \mathbb{N}_+$. We now formalized what is meant by a *universal* graph embedding model.

Definition 1 (Universal Graph Embedding)**.** *A graph inference model* $(\mathfrak{E}, \mathfrak{R})$ *is universal if: for every non-empty finite subset* $V \subseteq \mathbb{R}^D$ *and every connected weighted graph* $\mathcal{G} = (V, E, W)$ *there is a (quasi-metric) representation space* $(\mathcal{R}, d_{\mathcal{R}}) \in \mathfrak{R}$ *and an encoder* $\mathcal{E} : \mathbb{R}^D \to \mathcal{R}$ *in* $\mathfrak{E}$ *satisfying*

$$d_{\mathcal{G}}(u, v) = d_{\mathcal{R}}(\mathcal{E}(u), \mathcal{E}(v)) \qquad \forall\, u, v \in V.$$

Our interest in universal graph inference models lies in their ability to infer graph edges. This is done by first learning an embedding $\mathcal{E} \in \mathfrak{E}$ into some representation space $(\mathcal{R}, d_{\mathcal{R}}) \in \mathfrak{R}$ and subsequently sampling edges based on nearest neighbors within the aforementioned embedding.

One technical point worth noting is that, when forming sets of nearest neighbors, ties between equidistant points are broken arbitrarily. This is accomplished by indexing (possibly randomly) the graph's vertices and selecting the first few nearest points based on the ordering of that index, similar to the approach in Fakcharoenphol et al. (2004).

The formalization of this reconstruction procedure, in Theorem 1, uses the following notation. For every positive integer N, we denote the first N positive integers by $[N] \overset{\text{def.}}{=} \{1, \dots, N\}$. For every quasi-metric (representation) space $(\mathcal{R}, d_{\mathcal{R}})$ each point $x \in \mathcal{R}$, and each radius $r \geq 0$ the closed unit ball about x of radius r is $\bar{B}_{\mathcal{R}}(x, r) \overset{\text{def.}}{=} \{u \in \mathcal{R} :\, d_{\mathcal{R}}(x, y) \leq r\}$.

Theorem 1 (Generic Graph Reconstruction via Universal Graph Inference Models)**.** *Fix* $D \in \mathbb{N}_+$ *and a latent graph inference model* $(\mathfrak{E}, \mathfrak{R})$ *on* $\mathbb{R}^D$. *For every non-empty finite subset* $V \subseteq \mathbb{R}^D$*, every graph* $G = (V, E)$*, and each index* $V = \{v_i\}_{i=1}^N$ *there exists: a quasi-metric (representation) space* $(\mathcal{R}, d_{\mathcal{R}}) \in \mathfrak{R}$ *and an encoder* $\mathcal{E} : \mathbb{R}^D \to \mathcal{R}$ *in* $\mathfrak{E}$ *such that: for each* $i \in [N]$ *there is a (number of nearest neighbours)* $k_i \in [N]$ *satisfying*

$$\{u_i, u_j\} \in E \Leftrightarrow j \leq i^\star \text{ and } d_{\mathcal{R}}\big(\mathcal{E}(u_i), \mathcal{E}(u_j)\big) \leq r(k_i)$$

where $r(k_i) \overset{\text{def.}}{=} \inf\big\{r \geq 0 :\, \#\{v \in V :\, d_{\mathcal{R}}\big(\mathcal{E}(u_i), \mathcal{E}(v)\big) \leq r\} \geq k_i\big\}$ *and where* $i^\star \overset{\text{def.}}{=} \{j \in [N] : \#(\bar{B}_{\mathcal{R}}(u_i, r(k_i)) \cap \{u_s\}_{s=1}^j) \leq k_i\}$.

Theorem 1 shows that if a latent graph inference model is universal, then it can be used to reconstruct the edge set of any latent graph structure by first embedding the vertices/point-cloud into a latent representation space and then joining nearest neighbours. The next natural question is: *"How does one build a universal latent graph inference model which is differentiable?"*

It is known that the family of Euclidean spaces $\mathfrak{R} = \{(\mathbb{R}^d, \|\cdot\|)\}_{d \in \mathbb{N}_+}$ are not flexible enough to isometrically accommodate all weighted graphs; even if $\mathfrak{E}$ is the family of *all functions* from any $\mathbb{R}^D$ into any Euclidean space $(\mathbb{R}^d, \|\cdot\|)$. This is because, some weighted graphs do not admit isometric embeddings into any Euclidean space (Bourgain et al., 1986; Linial et al., 1995; Matoušek, 1997). Even infinite-dimensions need not be enough, since for every $n \in \mathbb{N}_+$, there is an n-point weighted graph which cannot be embedded in the Hilbert space ℓ^2 with distortion less than $\Omega(\log(n)/\log(\log(n))$ (Bourgain, 1985, Proposition 2). In particular, it cannot be isometrically embedded therein. For example, any l-leaf tree embeds in d-dimensional Euclidean space with distortion at-least $\Omega(l^{1/d})$. In contrast, any finite tree can embed with arbitrary low-distortion into the hyperbolic plane (Kratsios et al., 2023b), which is a particular two-dimensional non-flat Riemannian geometry. Several authors (Sarkar, 2012) have shown that cycle graphs can be embedded isometrically in spheres of appropriate dimension and radius (Schoenberg, 1942), or in the product of spheres (Guella et al., 2016), but they cannot be embedded isometrically in Euclidean space (Enflo, 1976). These observations motivate geodesic deep learners (Liu et al., 2019; Chamberlain et al., 2017; Chami et al., 2019; Sáez de Ocáriz Borde et al., 2023c;b) and network scientists (Verbeek & Suri, 2014) to use families of Riemannian representation spaces $\mathfrak{R}$ in which it is hoped that general graphs can faithfully be embedded, facilitating embedding-based latent graph inference. Unfortunately, 5 nodes and 5 edges are enough to construct a graph which cannot be isometrically embedded into *any* non-pathological Riemannian manifold.

Proposition 2 (Riemannian Representation Spaces are Too Rigid to be Universal)**.** *For any $D \in \mathbb{N}_+$ and any 5-point subset V of $\mathbb{R}^D$, there exists a set of edges E on V such that:*

(i) the graph $\mathcal{G} \overset{\text{def.}}{=} (V, E)$ is connected

(ii) for every complete[2] and connected smooth Riemannian manifold $(\mathcal{R}, g)$ there does not exist an isometric embedding $\varphi : (V, d_\mathcal{G}) \to (\mathcal{R}, d_\mathcal{R})$

where $d_\mathcal{G}$ and $d_\mathcal{R}$ respectively denote the shortest path (geodesic) distances on $\mathcal{G}$ and on $(\mathcal{R}, g)$.

In other words, Proposition 2 shows that if $\mathfrak{R}$ is any set of non-pathological Riemannian manifolds and $\mathfrak{E}$ any set of functions from $\mathbb{R}^D$ into any Riemannian manifold $(\mathcal{R}, g)$ in $\mathfrak{R}$ the graph inference model $(\mathfrak{E}, \mathfrak{R})$ is not universal. Furthermore, the graph "breaking its universality" is nothing obscure but a simple 5 node graph. Note that, no edge weights (not equal to 1) are needed in Proposition 2.

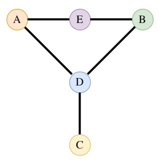

Figure 1: **Explanation of Proposition 2**: The Graph of Proposition 2 cannot be isometrically embedded into any complete and connected (smooth) Riemannian manifold. Briefly, the issue is that any isometric embedding into such a Riemannian manifold must exhibit a pair of geodesics one of which travels from the embeddings of node C to node A, while passing through the embedding of node D; and likewise, the other of which travels from the embedding of node C to node B and again passes through the embedding of node D. However, this would violate the local uniqueness of geodesics in such a Riemannian manifold, around the embedding of node D (implied by the Picard-Lindelöf theorem for ODEs); thus no such embedding can exist.

Proposition 2 improves on (Kratsios et al., 2023a, Propositions 13 and 15) since the latter only show that no compact connected Riemannian manifolds (e.g. products of spheres) and no connected Riemannian manifold with bounded non-positive sectional curvatures (e.g. products of hyperbolic spaces) can accommodate certain sequences of expander graphs (see (Kratsios et al., 2023a, Remark 14)). However, those results do not rule out more complicated Riemannian representation spaces; e.g. the products of spheres, hyperbolic, and Euclidean spaces recently explored by Gu et al. (2019); Tabaghi et al. (2021); Di Giovanni et al. (2022); Sáez de Ocáriz Borde et al. (2023c).

4.1.2 Universal Representation Is Possible With Neural Snowflake

Juxtaposed against Proposition 2, our first main result shows that together, neural snowflake and MLPs, are universal graph embedding models.

Theorem 2 (Universal Graph Embedding)**.** *Let $D \in \mathbb{N}_+$ and $\mathcal{G}$ be a weighted graph with $V \subseteq \mathbb{R}^D$ with $I \in \mathbb{N}_+$ vertices. There exists an embedding dimension $d \in \mathbb{N}_+$, an MLP $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^d$ with* ReLU *activation function, and a neural snowflake f such that*

$$d_\mathcal{G}(u, v) = \|\mathcal{E}(u) - \mathcal{E}(v)\|_f,$$

for each $u, v \in V$. The $(\mathbb{R}^d, \|\cdot\|_f)$ supports a $2^{O(\log(1+\frac{1}{I-1})^{-1})}$-relaxed triangle inequality. If $\mathcal{G}$ is a tree then $(\mathbb{R}^d, \|\cdot\|_f)$ instead supports an 8-triangle inequality.

Comparison: State-of-the-Art Deep Embedding Guarantees. Recently, Kratsios et al. (2023a) built on Andoni et al. (2018) and proposed a universal graph embedding model which uses the

[2]Here, complete is meant in the sense of metric spaces; i.e. all Cauchy sequences in a complete metric space have a limit therein.

transformer architecture of Kratsios (2021) to represent graphs in the order 2-Wasserstein space on $\mathbb{R}^3$. The drawbacks of this approach are that the metric is not available in closed-form, it is computationally infeasible to evaluate exactly for large graph embeddings, and it is still challenging to evaluate approximately (Cuturi, 2013). In contrast, Theorem 2 guarantees that a simple MLP can isometrically embedding any weighted graph into a finite-dimensional representation space with closed-form distance function explicitly implemented by a neural snowflake.

Comparison: MLP without Neural Snowflake. We examine the necessity of the neural snowflake in Theorem 2, by showing that the MLP alone cannot isometrically represent any weighted graph into its natural output space; namely some Euclidean space.

Theorem 3 (Neural Snowflakes & MLPs Are More Powerful For Representation Learning Than MLPs)**.** *Let $d, D \in \mathbb{N}_+$. The following hold:*

- *(i)* ***No Less Expressive Than MLP:*** *For any weighted graph $\mathcal{G} = (V, E, W)$ with $V \subseteq \mathbb{R}^D$, if there is an MLP $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^d$ which isometrically embeds $\mathcal{G}$ then there is a neural snowflake f and an MLP $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^d$ which isometrically embeds $\mathcal{G}$ into $(\mathbb{R}^d, \| \cdot \|_f)$.*
- *(ii)* ***Strictly More Expressive Than MLP:*** *There exists a complete weighted graph $G = (V, E, W)$ with $V \subset \mathbb{R}^D$ by any MLP $\tilde{\mathcal{E}} : \mathbb{R}^D \to \mathbb{R}^d$ but for which there exists a neural snowflake f and an MLP $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^d$ that isometrically embeds $\mathcal{G}$ into $(\mathbb{R}^D, \| \cdot \|_f)$.*

4.2 Isometric Representation Guarantees - By Small Neural Snowflakes

Theorem 2 offers a qualitative assurance that the neural snowflake can represent any finite weighted graph. We now show that any weighted graph which can be isometrically represented in the latent geometry induced by kernel regressors, can be implemented by a neural snowflake. We assume that the latent geometry of the weighted graph is encoded in a low-dimensional space and the distances in that low-dimensional space are given by a *radially symmetric* and positive-definite kernel.

Assumption 1 (Latent Radially-Symmetric Kernel)**.** *There are $d, D \in \mathbb{N}_+$, a feature map $\Phi : \mathbb{R}^D \to \mathbb{R}^d$, and a non-constant positive-definite function*[3] *$f : [0, \infty) \to [0, \infty)$ satisfying*

$$d_{\mathcal{G}}(x, y) = \bar{f}\big(\|\Phi(x) - \Phi(y)\|\big),$$

for every $x, y \in \mathbb{R}^d$; where $\bar{f}(t) \overset{\text{def.}}{=} f(0) - f(t)$.

Several satisfy Assumption 1, as we emphasize using two exotic examples (Appendix D).

Example 2. *The map $f(t) \overset{\text{def.}}{=} (1 + \sum_{k=1}^K |t|^{r_k})^{-\beta}$ satisfies Assumption 1 for any $K \in \mathbb{N}_+$ and $0 \leq r_1, \ldots, r_K, \beta \leq 1$.*

Example 3. *The map $f(t) \overset{\text{def.}}{=} \exp\left(\frac{-a(t-1)}{\log(t)}\right)$ satisfies Assumption 1, for all $a > 0$.*

We find that the neural snowflake can implement an isometric representation of the latent geometry, as in Theorem 2, using a small number of parameters comparable to Theorem 5.

Theorem 4 (Quantitative Embedding Guarantees for Bounded Metric Geometries)**.** *Let $D, d \in \mathbb{N}_+$, G be a finite weighted graph with $V \subset \mathbb{R}^D$ and suppose that Assumption 1 (or equation 2) holds. Then, there is a neural snowflake $(\mathcal{E}, f)$ satisfying*

$$\|\Phi(v) - \Phi(u)\|_f = \|\mathcal{E}(v) - \mathcal{E}(u)\|_{\bar{f}}$$

for every $u, v \in V$. Furthermore, the depth and width of $\mathcal{E}$ and f are recorded in Table 1.

4.3 Representation Guarantees Leveraging Distortion

Theorem 2 shows the neural snowflake and an simple MLP are *universal graph embedding* models. By allowing the neural snowflake and MLP some possible slack to distort the latent graph's geometry, either by stretching or contracting pairwise distances ever so slightly, we are able to derive explicit bounds on the embedding dimension d and on the complexity of the neural snowflake and MLP.

[3] A positive-definite function, is map $f : \mathbb{R} \to \mathbb{C}$ for which each $\big(f(x_i - x_j)\big)_{i,j=1}^N$ is a positive-definite matrix, for every $N \in \mathbb{N}_+$ and each $x_1, \ldots, x_N \in \mathbb{R}$.

Theorem 5 (Quantitative Approximately Isometric Embeddings for Weighted Graphs)**.** *Let $G = (V, E, W)$ be a weighted graph $N \in \mathbb{N}_+$ nodes, a non-negative weighting function W, and $V \subset \mathbb{R}^d$. For every $1 < p < 2$, there exists an MLP with ReLU activation function $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}^{O(\log(\dim(G))}$ and a snowflaking network $f : \mathbb{R} \to \mathbb{R}$ such that: for every $x, u \in V$*

$$d_G(x, u) \leq \|\mathcal{E}(x) - \mathcal{E}(u)\|_f \leq D^p \, d_G(x, u),$$

where $D \in \mathcal{O}\big(\frac{\log(\dim(G))^2}{p^2}\big)$ and $(\mathbb{R}^n, \| \cdot \|_f)$ is a quasi-metric space satisfying the 2^{p-1}-triangle inequality. The depth and width of $\mathcal{E}$ and f are recorded in Table 1.

Table 1: Complexity of the Neural Snowflake and MLPs. Here, $\mathcal{O}$ suppresses a constant depending only on D and $C_{\mathcal{G}} > 0$ depends only on $\mathcal{G}$ (explicit constants are given in Appendix B).

Geometry	**Net.**	**Hidden Layers (Depth -1)**	**Width**	**Theorem**
General	f	$\Theta(1)$	$\Theta(1)$	5
Bounded	f	$\Theta(1)$	$\mathcal{O}(I^2)$	4
All	$\mathcal{E}$	$\mathcal{O}\big(I\sqrt{I \log(I)} \log\big(I^2 \, C_{\mathcal{G}}\big)\big)$	$\mathcal{O}\big(DI + d\big)$	4 and 5

Discussion - Comparison with the Best Proven Deep Representation. The complexity of the neural snowflake and MLP are reported in Table 1. Theorem 5 shows that the neural snowflake can match the best known embedding guarantees by a deep learning model with values in a curved infinite-dimensional space (Kratsios et al., 2023a, Theorem 4), both in terms of distortion and the number of parameters. The neural snowflake's adaptive geometry allows for the representations to be implemented in $\mathcal{O}(\log(\dim(G))$ dimensions and the implemented representation space $(\mathbb{R}^d, \| \cdot \|_f)$ has an explicit closed-form distance function making it trivial to evaluate; unlike the distance function in the representation space of Kratsios et al. (2023a). Our guarantees show that neural snowflakes can theoretically represent any weighted graph; either isometrically or nearly isometrically with provable few parameters.

5 Experimental Results

Next, we validate our embedding results through synthetic graph experiments, and we demonstrate how the combined representation capabilities of neural snowflakes and MLPs can enhance existing state-of-the-art latent graph inference pipelines.

Synthetic Embedding Experiments: Neural Snowflakes vs Euclidean Graph Embedding Spaces. To assess the effectiveness of neural snowflakes as well as to compare their performance with that of MLPs in approximating metrics using Euclidean space, we conduct synthetic graph embedding experiments. Specifically, we focus on fully connected graphs, where the node coordinates are randomly sampled from a multivariate Gaussian distribution within a 100-dimensional hypercube in Euclidean space, denoted as $\mathbb{R}^{100}$. The weights of the graphs are computed according to the metrics in Table 2. In the leftmost column, the MLP model projects the node features in $\mathbb{R}^{100}$ to $\mathbb{R}^2$ and aims at approximating the edge weights of the graph using the euclidean distance $\|\text{MLP}(\mathbf{x}) - \text{MLP}(\mathbf{y})\|$ between the projected features. In the central column, the neural snowflake f learns a quasi-metric based on the input Euclidean metric previously mentioned in $\mathbb{R}^2$, $f(\|\text{MLP}(\mathbf{x}) - \text{MLP}(\mathbf{y})\|)$. Finally, in the rightmost column the neural snowflake learns based on the Euclidean distance between features in $\mathbb{R}^{100}$: $f(\|\mathbf{x} - \mathbf{y}\|)$. In order to demonstrate the superior embedding capabilities of neural snowflakes compared to Euclidean spaces, we intentionally equip the MLP in the first column with a significantly larger number of model parameters than the other models. This is done to highlight the fact that, despite having fewer learnable parameters, neural snowflakes outperform Euclidean spaces by orders of magnitude. Furthermore, the results of our experiments reveal that neural snowflakes exhibit remarkable flexibility in learning metrics even in lower-dimensional spaces such as $\mathbb{R}^2$. This is evident from the similarity of results obtained in the rightmost column (representing $\mathbb{R}^{100}$) compared to those achieved in the case of learning the metric in $\mathbb{R}^2$. Additional information regarding these embedding experiments can be found in Appendix I.

Graph Benchmarks. In this section, we present our findings on using Neural Snowflakes compared to other latent graph inference models. Our objective is to evaluate the representation power offered

Table 2: Results for synthetic graph embedding experiments for the test set. The Neural Snowflake models are able to learn the metric better with substantially lesser number of model parameters.

	MLP	**Neural Snowflake (+ MLP)**	**Neural Snowflake**
No. Parameters	5422	4169	847
Embedding space, $\mathbb{R}^n$	2	2	100
Metric	Mean Square Embedding Error		
$\|\mathbf{x}-\mathbf{y}\|^{0.5}\log(1+\|\mathbf{x}-\mathbf{y}\|)^{0.5}$	1.3196	0.0034	**0.0029**
$\|\mathbf{x}-\mathbf{y}\|^{0.1}\log(1+\|\mathbf{x}-\mathbf{y}\|)^{0.9}$	1.2555	0.0032	**0.0032**
$1-\frac{1}{(1+\|\mathbf{x}-\mathbf{y}\|^{0.5})}$	0.1738	**0.00004**	**0.00004**
$1-\exp\frac{-(\|\mathbf{x}-\mathbf{y}\|-1)}{\log(\|\mathbf{x}-\mathbf{y}\|)}$	0.3249	0.00008	**0.00008**
$1-\frac{1}{(1+\|\mathbf{x}-\mathbf{y}\|)^{0.2}}$	0.0315	**0.00005**	**0.00005**
$1-\frac{1}{1+\|\mathbf{x}-\mathbf{y}\|^{0.2}+\|\mathbf{x}-\mathbf{y}\|^{0.5}}$	0.2484	**0.00002**	**0.00002**

by various latent space metric spaces, and thus, we contrast our results with the original DGM (Kazi et al., 2022) and its non-Euclidean variants (Sáez de Ocáriz Borde et al., 2023b;c). We also introduce a new variant of the DGM which uses a snowflake activation on top of the classical DGM, to equip the module with a snowflake quasi-metric space. We ensure a fair evaluation, by conducting all experiments using the same GCN model and only modify the latent geometries used for latent graph inference. We take care to maintain consistency in the number of model parameters, as well as other training specifications such as learning rates, training and testing splits, number of GNN layers, and so on. This approach guarantees that all comparisons are solely based on the metric (or quasi-metric) space utilized for representations. By eliminating the influence of other factors, we can obtain reliable and trustworthy experimental results. A detailed and systematic analysis of the results is provided in Appendix I.

We first present results from latent graph inference on the well-known Cora and CiteSeer homophilic graph benchmarks. We use a consistent latent space dimensionality of 8 and perform the Gumbel top-k trick for edge sampling with a k value of 7. The models all share the same latent space dimensionality, differing solely in their geometric characteristics. In scenarios where a product manifold is used, the overall manifold is constructed by amalgamating two 4-dimensional manifolds through a Cartesian product. This yields a total latent space dimensionality of 8. This methodology ensures an equitable comparison based exclusively on geometric factors. All other parameters, comprising network settings and training hyperparameters, remain unaltered. For all DGM modules, GCNs are employed as the underlying GNN diffusion layers. Table 3 displays the results for Cora and CiteSeer, leveraging the original dataset graphs as inductive biases.

Table 3: Results for Cora and CiteSeer leveraging the original input graph as an inductive bias.

		Cora	**CiteSeer**
Model	Metric Space	Accuracy (%) ± Standard Deviation	
Neural Snowflake	Snowflake	**87.07**$\pm$**3.45**	**74.76**$\pm$**1.74**
DGM	Snowflake	85.41$\pm$3.70	**74.19**$\pm$**2.08**
DGM	Euclidean	**85.77**$\pm$**3.64**	73.67$\pm$2.30
DGM	Hyperboloid	85.25$\pm$3.34	73.46$\pm$1.79
DGM	Poincare	**86.07**$\pm$**3.53**	71.23$\pm$5.53
DGM	Spherical	76.14$\pm$2.84	73.13$\pm$2.93
DGM	Euclidean $\times$ Hyperboloid	84.33$\pm$2.56	73.29$\pm$2.18
DGM	Hyperboloid $\times$ Hyperboloid	84.59$\pm$5.40	**74.42**$\pm$**1.83**
GCN	Euclidean	83.50$\pm$2.00	70.03$\pm$2.04

Next we perform experiments on Cora and CiteSeer without considering their respective graphs, that is, the latent graph inference models only take pointclouds as inputs in this case. We also include results for the Tadpole and Aerothermodynamics datasets used in Sáez de Ocáriz Borde et al. (2023c). Note that in these experiments all models used GCNs and a fixed latent space dimensionality of 8, unlike in the original paper which uses a larger latent spaces and GAT layers. The effect of changing the latent space dimensionality is further explored in Appendix I. For both Tadpole and Aerothemodynamics, the Gumbel Top-k algorithm samples 3 edges per node. See Table 4.

Table 4: Results for Cora and CiteSeer, and the real-world Tadpole and Aerothermodynamics datasets, without leveraging the original input graph as an inductive bias.

		Cora	**CiteSeer**	**Tadpole**	**Aerothermodynamics**
Model	Metric Space	Accuracy (%) ± Standard Deviation			
Neural Snowflake	Snowflake	**71.22**±4.27	**67.80**±2.44	**90.02**±3.51	**88.65**±3.10
DGM	Snowflake	**69.51**±4.42	66.86±2.82	**91.61**±4.38	88.55±2.35
DGM	Euclidean	68.37±5.39	**68.10**±2.80	89.29±4.66	88.28±2.61
DGM	Hyperboloid	**70.00**±4.08	**68.34**±1.59	88.75±5.11	88.29±2.85
DGM	Poincare	65.74±4.02	64.63±2.98	86.96±5.30	**89.00**±2.34
DGM	Spherical	37.03±14.33	20.00±4.13	82.32±11.74	88.37±3.00
DGM	Euclidean × Hyperboloid	62.18±6.61	65.72±2.48	**90.71**±3.17	88.20±3.23
DGM	Hyperboloid × Hyperboloid	67.14±4.19	64.33±9.44	89.64±5.57	**89.55**±2.52
MLP	Euclidean	58.92±3.28	59.48±2.14	87.70±3.46	80.99±8.34

From the results, we can see that models using snowflake metric spaces are consistently amongst the top performers for both Cora and CiteSeer. On the other hand, employing non-learnable metric spaces necessitates conducting an exploration of various latent space geometries to achieve optimal outcomes, since there is no metric space that consistently outperforms the rest regardless of the dataset. In our synthetic experiments, we have clearly demonstrated the remarkable advantage of neural snowflakes in learning metric spaces with enhanced flexibility, when compared to Euclidean space. In the context of latent graph inference, this distinction is not as pronounced as observed in the synthetic experiments. This can be attributed to the suboptimal nature of the Gumbel Top-k edge sampling algorithm (Appendix E), a topic discussed in other research works (Kazi et al., 2022; Battiloro et al., 2023), which essentially introduces a form of "distortion" to the learned metric. Yet, it is worth noting that the development of improved edge sampling algorithms to foster better synergy between metric space learning and graph construction is not the primary focus of this paper. Instead, our emphasis is on introducing a more comprehensive and trainable metric space and integrating it with existing edge sampling techniques.

6 Conclusion

Our theoretical analysis showed that a small neural snowflake, denoted as f, can adaptively implement fractal-like geometries $|\cdot|_f$ on $\mathbb{R}^d$, which are flexible enough to grant a small MLP the capacity to isometrically embed any finite graph. We showed that the non-smooth geometry implemented by the neural snowflake is key by showing that there are simple graphs that cannot be isometrically embedded into any smooth Riemannian representation space. We then explored several cases in which the combination of neural snowflakes and MLPs requires a small number of total parameters, independent of ambient dimensions, to represent certain classes of regular weighted graphs. We complemented our theoretical analysis by extensively exploring the best approaches to implement neural snowflakes, ensuring stability during training, in both synthetic and graph benchmark experiments. We also introduced a snowflake activation that can easily be integrated into the DGM module using a differentiable distance function, enabling the DGM to leverage a snowflake quasi-metric. We conducted tests on various graph benchmarks, systematically comparing the effectiveness of snowflake quasi-metric spaces for latent graph inference with other Riemannian metrics such as Euclidean, hyperbolic variants, spherical spaces, and product manifolds of model spaces. Our proposed model is consistently able to match or outperform the baselines while learning the latent geometry in a differentiable manner and without having to perform random search to find the optimal embedding space.

Note that our experiments were conducted in accordance with the differentiable graph module framework for discrete edge sampling as proposed by Kazi et al. (2022). Our primary objective was to compare the representation capabilities of various (quasi-)metric spaces, while keeping all other model architecture choices constant. Recently, the NodeFormer (Wu et al., 2023) architecture was introduced, enabling the scalability of latent graph inference to large graphs. However, this development is slightly tangential to the research presented in this work, which focuses on analyzing the geometric characteristics of different embedding spaces. We propose considering the incorporation of the geometric notions discussed in this work into new scalable architectures, such as the NodeFormer, as part of future research.

7 Acknowledgment and Funding

AK acknowledges financial support from the NSERC Discovery Grant No. RGPIN-2023-04482 and their McMaster Startup Funds.

References

Ben Adcock, Simone Brugiapaglia, Nick Dexter, and Sebastian Moraga. Deep neural networks are effective at learning high-dimensional hilbert-valued functions from limited data. *arXiv preprint arXiv:2012.06081*, 2020.

Stephanie B Alexander and Richard L Bishop. Warped products of hadamard spaces. *manuscripta mathematica*, 96:487–505, 1998.

Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International Conference on Machine Learning*, pp. 146–155. PMLR, 2017.

Alexandr Andoni, Assaf Naor, and Ofer Neiman. Snowflake universality of Wasserstein spaces. *Ann. Sci. Éc. Norm. Supér. (4)*, 51(3):657–700, 2018. ISSN 0012-9593,1873-2151. doi: 10.24033/asens.2363. URL https://doi.org/10.24033/asens.2363.

Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver. Diffwire: Inductive graph rewiring via the lovász bound. In *LOG IN*, 2022.

Claudio Battiloro, Indro Spinelli, Lev Telyatnikov, Michael Bronstein, Simone Scardapane, and Paolo Di Lorenzo. From latent graph to latent topology inference: Differentiable cell complex module, 2023.

Wendong Bi, Lun Du, Qiang Fu, Yanlin Wang, Shi Han, and Dongmei Zhang. Make heterophily graphs better fit gnn: A graph rewiring approach. *ArXiv*, abs/2209.08264, 2022.

Helmut Bolcskei, Philipp Grohs, Gitta Kutyniok, and Philipp Petersen. Optimal approximation with sparsely connected deep neural networks. *SIAM Journal on Mathematics of Data Science*, 1(1): 8–45, 2019.

J. Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel J. Math.*, 52 (1-2):46–52, 1985. ISSN 0021-2172. doi: 10.1007/BF02776078. URL https://doi.org/10.1007/BF02776078.

J. Bourgain, V. Milman, and H. Wolfson. On type of metric spaces. *Trans. Amer. Math. Soc.*, 294(1):295–317, 1986. ISSN 0002-9947,1088-6850. doi: 10.2307/2000132. URL https://doi.org/10.2307/2000132.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, Cambridge, 2004. ISBN 0-521-83378-7. doi: 10.1017/CBO9780511804441. URL https://doi.org/10.1017/CBO9780511804441.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017. doi: 10.1109/MSP.2017.2693418.

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

Benjamin P. Chamberlain, James R. Clough, and Marc Peter Deisenroth. Neural embeddings of graphs in hyperbolic space. *ArXiv*, 2017.

Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks, 2019.

Benjamin Charlier, Jean Feydy, Joan Alexis Glaunès, François-David Collin, and Ghislain Durif. Kernel operations on the gpu, with autodiff, without memory overflows. *Journal of Machine Learning Research*, 22(74):1–6, 2021.

Chien-Hsiung Chen. Warped products of metric spaces of curvature bounded from above. *Transactions of the American Mathematical Society*, 351(12):4727–4740, 1999.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, 2020a.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *ArXiv*, abs/2006.13009, 2020b.

Luca Di Cosmo, Anees Kazi, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M. Bronstein. Latent-graph learning for disease prediction. In *MICCAI*, 2020.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Hennie Daniels and Marina Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

Ingrid Daubechies, Ronald DeVore, Simon Foucart, Boris Hanin, and Guergana Petrova. Nonlinear approximation and (deep) relu networks. *Constructive Approximation*, 55(1):127–172, 2022.

M Deza and Hiroshi Maehara. Metric transforms and euclidean embeddings. *Transactions of the American Mathematical Society*, 317(2):661–671, 1990.

Francesco Di Giovanni, Giulia Luise, and Michael Bronstein. Heterogeneous manifolds for curvature-aware graph embedding, 2022.

Dennis Elbrächter, Dmytro Perekrestenko, Philipp Grohs, and Helmut Bölcskei. Deep neural network approximation theory. *IEEE Transactions on Information Theory*, 67(5):2581–2623, 2021.

P. Enflo. Uniform homeomorphisms between Banach spaces. In *Séminaire Maurey-Schwartz (1975–1976), Espaces L^p, applications radonifiantes et géométrie des espaces de Banach,*, pp. Exp. No. 18, 7. „ 1976.

Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. Approximating metrics by tree metrics. *ACM SIGACT News*, 35(2):60–70, 2004.

T. Fetaya, Elias Wang, K. C. Welling, Michelle Zemel, Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. *arXiv: Machine Learning*, 2018.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *ICML*, 2019.

Luca Galimberti, Giulia Livieri, and Anastasis Kratsios. Designing universal causal deep learning models: The case of infinite-dimensional dynamical systems from stochastic analysis. *arXiv preprint arXiv:2210.13300*, 2022.

Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *ICLR*, 2019.

J. C. Guella, V. A. Menegatto, and A. P. Peron. An extension of a theorem of Schoenberg to products of spheres. *Banach J. Math. Anal.*, 10(4):671–685, 2016. ISSN 2662-2033. doi: 10.1215/17358787-3649260. URL https://doi.org/10.1215/17358787-3649260.

Jiayan Guo, Lun Du, Wendong Bi, Qiang Fu, Xiaojun Ma, Xu Chen, Shi Han, Dongmei Zhang, and Yan Zhang. Homophily-oriented heterogeneous graph rewiring. *Proceedings of the ACM Web Conference 2023*, 2023.

Jeff Hawkins, Subutai Ahmad, and Yuwei Cui. A theory of how columns in the neocortex enable learning the structure of the world. *Frontiers in Neural Circuits*, 11, 2017.

Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. In *NIPS*, 2017.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

Jürgen Jost. *Riemannian geometry and geometric analysis*. Universitext. Springer, Cham, seventh edition, 2017. ISBN 978-3-319-61859-3; 978-3-319-61860-9. doi: 10.1007/978-3-319-61860-9. URL https://doi.org/10.1007/978-3-319-61860-9.

Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael Bronstein. Differentiable graph module (DGM) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2022.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, 2017.

Wouter Kool, Herke van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement, 2019.

Anastasis Kratsios. Universal regular conditional distributions. *arXiv preprint arXiv:2105.07743*, 2021.

Anastasis Kratsios and Leonie Papon. Universal approximation theorems for differentiable geometric deep learning. *Journal of Machine Learning Research*, 23(196):1–73, 2022.

Anastasis Kratsios, Valentin Debarnot, and Ivan Dokmanić. Small transformers compute universal metric embeddings. *Journal of Machine Learning Research*, 24(170):1–48, 2023a. URL http://jmlr.org/papers/v24/22-1246.html.

Anastasis Kratsios, Ruiyang Hong, and Haitz Sáez de Ocáriz Borde. Capacity bounds for hyperbolic neural network representations of latent tree structures, 2023b.

R Krauthgamer, JR Lee, M Mendel, and A Naor. Measured descent: a new embedding method for finite metrics. *Geometric & Functional Analysis GAFA*, 15(4):839–858, 2005.

Enrico Le Donne and Tapio Rajala. Assouad dimension, Nagata dimension, and uniformly close metric tangents. *Indiana Univ. Math. J.*, 64(1):21–54, 2015. ISSN 0022-2518. doi: 10.1512/iumj.2015.64.5469. URL https://doi.org/10.1512/iumj.2015.64.5469.

Enrico Le Donne, Tapio Rajala, and Erik Walsberg. Isometric embeddings of snowflakes into finite-dimensional Banach spaces. *Proc. Amer. Math. Soc.*, 146(2):685–693, 2018. ISSN 0002-9939. doi: 10.1090/proc/13778. URL https://doi.org/10.1090/proc/13778.

Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995. ISSN 0209-9683. doi: 10.1007/BF01200757. URL https://doi.org/10.1007/BF01200757.

Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. In *NeurIPS*, 2019.

Jianfeng Lu, Zuowei Shen, Haizhao Yang, and Shijun Zhang. Deep network approximation for smooth functions. *SIAM J. Math. Anal.*, 53(5):5465–5506, 2021. ISSN 0036-1410. doi: 10.1137/20M134695X. URL https://doi.org/10.1137/20M134695X.

Hiroshi Maehara. Metric transforms of finite spaces and connected graphs. *Discrete mathematics*, 61(2-3):235–246, 1986.

Carlo Marcati, Joost AA Opschoor, Philipp C Petersen, and Christoph Schwab. Exponential relu neural network approximation rates for point and edge singularities. *Foundations of Computational Mathematics*, pp. 1–85, 2022.

Jiří Matoušek. On embedding expanders into l_p spaces. *Israel J. Math.*, 102:189–197, 1997. ISSN 0021-2172,1565-8511. doi: 10.1007/BF02773799. URL https://doi.org/10.1007/BF02773799.

Robert J. McGlinn. Uniform approximation of completely monotone functions by exponential sums. *J. Math. Anal. Appl.*, 65(1):211–218, 1978. ISSN 0022-247X. doi: 10.1016/0022-247X(78)90210-X. URL https://doi.org/10.1016/0022-247X(78)90210-X.

Hrushikesh N Mhaskar and Tomaso Poggio. Deep vs. shallow networks: An approximation theory perspective. *Analysis and Applications*, 14(06):829–848, 2016.

Assaf Naor and Ofer Neiman. Assouad's theorem with dimension independent of the snowflaking. *Rev. Mat. Iberoam.*, 28(4):1123–1142, 2012. ISSN 0213-2230. doi: 10.4171/RMI/706. URL https://doi-org.libaccess.lib.mcmaster.ca/10.4171/RMI/706.

Philipp Petersen and Felix Voigtlaender. Optimal approximation of piecewise smooth functions using deep relu neural networks. *Neural Networks*, 108:296–330, 2018.

T. R. L. Phillips, K. M. Schmidt, and A. Zhigljavsky. Extension of the Schoenberg theorem to integrally conditionally positive definite functions. *J. Math. Anal. Appl.*, 470(1):659–678, 2019. ISSN 0022-247X. doi: 10.1016/j.jmaa.2018.10.032. URL https://doi.org/10.1016/j.jmaa.2018.10.032.

Iosif Pinelis. Representation of continuous, monotone, concave functions. URL https://mathoverflow.net/q/448012.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Haitz Sáez de Ocáriz Borde, Alvaro Arroyo, Ismael Morales, Ingmar Posner, and Xiaowen Dong. Neural latent geometry search: Product manifold inference via gromov-hausdorff-informed bayesian optimization, 2023a.

Haitz Sáez de Ocáriz Borde, Alvaro Arroyo, and Ingmar Posner. Projections of model spaces for latent graph inference. In *ICLR 2023 Workshop on Physics for Machine Learning*, 2023b.

Haitz Sáez de Ocáriz Borde, Anees Kazi, Federico Barbero, and Pietro Lio. Latent graph inference using product manifolds. *The Eleventh International Conference on Learning Representations*, 2023c.

Rik Sarkar. Low distortion Delaunay embedding of trees in hyperbolic plane. In *Graph drawing*, volume 7034 of *Lecture Notes in Comput. Sci.*, pp. 355–366. Springer, Heidelberg, 2012. doi: 10.1007/978-3-642-25878-7_34. URL https://doi.org/10.1007/978-3-642-25878-7_34.

René L. Schilling, Renming Song, and Zoran Vondraček. *Bernstein functions*, volume 37 of *De Gruyter Studies in Mathematics*. Walter de Gruyter & Co., Berlin, second edition, 2012. ISBN 978-3-11-025229-3; 978-3-11-026933-8. doi: 10.1515/9783110269338. URL https://doi.org/10.1515/9783110269338. Theory and applications.

I. J. Schoenberg. Metric spaces and completely monotone functions. *Ann. of Math. (2)*, 39(4):811–841, 1938. ISSN 0003-486X. doi: 10.2307/1968466. URL https://doi.org/10.2307/1968466.

I. J. Schoenberg. Positive definite functions on spheres. *Duke Math. J.*, 9:96–108, 1942. ISSN 0012-7094. URL http://projecteuclid.org/euclid.dmj/1077493072.

Isaac J Schoenberg. On certain metric spaces arising from euclidean spaces by a change of metric and their imbedding in hilbert space. *Annals of mathematics*, pp. 787–793, 1937.

Zuowei Shen, Haizhao Yang, and Shijun Zhang. Optimal approximation rate of relu networks in terms of width and depth. *Journal de Mathématiques Pures et Appliquées*, 157:101–135, 2022.

Puoya Tabaghi, Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenković. Linear classifiers in product space forms, 2021.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *ArXiv*, abs/2111.14522, 2021.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *ArXiv*, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *ArXiv*, 2018.

Kevin Verbeek and Subhash Suri. Metric embedding, hyperbolic space, and social networks. *Proceedings of the thirtieth annual symposium on Computational geometry*, 2014.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38:1 – 12, 2019.

Nik Weaver. *Lipschitz algebras*. World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2018. ISBN 978-981-4740-63-0. Second edition of [MR1832645].

Jakob Weissteiner, Jakob Heiss, Julien Siems, and Sven Seuken. Monotone-value neural networks: Exploiting preference monotonicity in combinatorial assignment. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI-22)*, 2022.

David Vernon Widder. *The Laplace Transform*. Princeton Mathematical Series, vol. 6. Princeton University Press, Princeton, N. J., 1941.

Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification, 2023.

Qinglan Xia. The geodesic problem in quasimetric spaces. *J. Geom. Anal.*, 19(2):452–479, 2009. ISSN 1050-6926. doi: 10.1007/s12220-008-9065-4. URL https://doi-org.libaccess.lib.mcmaster.ca/10.1007/s12220-008-9065-4.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94: 103–114, 2017.

Ding-Xuan Zhou. Theory of deep convolutional neural networks: Downsampling. *Neural Networks*, 124:319–327, 2020a.

Ding-Xuan Zhou. Universality of deep convolutional neural networks. *Applied and computational harmonic analysis*, 48(2):787–794, 2020b.

Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. Deep graph structure learning for robust representations: A survey. *ArXiv*, abs/2103.03036, 2021.

A Additional Background

Many of the proofs or our paper's results, and generalizations thereof contained only in the manuscript's appendix, rest on some additional technology from the theory of metric spaces. This brief appendix overviews those tools. We also include formal definitions of the involved MLPs with ReLU activation function which we routinely use.

A.1 Metric Spaces

Suppose that (X, d) is a metric space; i.e. $C = 1$ in (iii) above. Topologically, (X, d) and its snowflakes (X, d^p) are identical but geometrically they are quite different. Geometrically, the latter much more complex than the former. In this paper we quantify complexity, when required, using the *doubling dimension* and *aspect ratio* of a metric space.

Denote a ball of radius $r \geq 0$ in (X, d) about a point $x \in X$ is the set $B(x, r) \overset{\text{def.}}{=} \{u \in X : d(x, u) < r\}$. The *doubling dimension* of (X, d), denoted by $\dim(X, d)$, is the smallest integer k for which: for every ball $B(x, r)$ about any point $x \in X$ and radius $r > 0$ there are $x_1, \dots, x_{2^k} \in X$ covering it by balls of half its radius; i.e. the metric ball $B(x, r) \subseteq \bigcup_{i=1}^{2^k} B(x_i, r/2)$. We emphasize that metric tools are finer then topological tools, for instance the dimension of many metric spaces[4] Le Donne & Rajala (2015) is often by much greater than their topological dimension.

We make use of the *aspect ratio* of a weighted graph $\mathcal{G} = (V, E, W)$, as the ratio of the largest distance $d_{\mathcal{G}}$ between any two nodes over the smallest possible distance between any two nodes, with respect to the geodesic distance on $\mathcal{G}$. This aspect ratio coincides with the aspect ratio used in Kratsios et al. (2023a) and a metric variant of the aspect ratio of the measure theoretic aspect ratio of Krauthgamer et al. (2005). Briefly, when $\#V > 1$ we define

$$\operatorname{aspect}(\mathcal{G}) \overset{\text{def.}}{=} \frac{\max_{v,u \in V} d_{\mathcal{G}}(u,v)}{\min_{v,u\in V;\, u\neq v} d_{\mathcal{G}}(u,v)},$$

otherwise, we set $\operatorname{aspect}(\mathcal{G}) = 1$. In the case where $V \subset \mathbb{R}^D$, for some $D \in \mathbb{N}_+$, and $W(u,v) = \|u - v\|$ we write $\operatorname{aspect}_2(V) \overset{\text{def.}}{=} \operatorname{aspect}(\mathcal{G})$.

We often reply on the notion of a *bi-Lipschitz* embedding, which we now recall. Given any $0 < s \leq L < \infty$, metric spaces (X, d) and (Y, ρ), and a map $f : \mathcal{X} \to \mathcal{Y}$ is called (s, L)-*bi-Lipschitz* if

$$s\, d(u,v) \leq \rho(f(u), f(v)) \leq L\, d(u,v) \tag{4}$$

for each $u, v \in \mathcal{X}$. If, $s = L = 1$, then the map f is said to be an isometric embedding[5].

A.2 MLPs with ReLU Activation Function

We will often be using **M**ulti-**l**ayer **P**erceptron (MLPs) based on the perceptron model of Rosenblatt (1958), and typically called feedforward neural networks in the contemporary approximation theory literature Mhaskar & Poggio (2016); Yarotsky (2017); Petersen & Voigtlaender (2018); Bolcskei et al. (2019); Elbrächter et al. (2021); Galimberti et al. (2022); Daubechies et al. (2022); Marcati et al. (2022); Adcock et al. (2020); Shen et al. (2022). Our MLPs will always use the $\operatorname{ReLU}(t) \overset{\text{def.}}{=} \max\{0, t\}$ activation function, where $t \in \mathbb{R}$. We will routinely apply the ReLU function *component-wise*, any vector $u \in \mathbb{R}^d$ for any positive integer N, denoted by $\operatorname{ReLU} \bullet u$ and defined by

$$\sigma \bullet u \overset{\text{def.}}{=} (\sigma(u_i))_{i=1}^N. \tag{5}$$

For any pair of positive integers $D, d \in \mathbb{N}_+$, a map $f : \mathbb{R}^D \to \mathbb{R}^d$ is called an *MLP* if it admits the recursive representation

$$\begin{aligned} f(x) &\overset{\text{def.}}{=} A^{(J)} x^{(J)} + b^{(J)}, \\ x^{(j+1)} &\overset{\text{def.}}{=} \sigma \bullet (A^{(j)} x^{(j)} + b^{(j)}) \quad \text{for } j = 0, \dots, J-1, \\ x^{(0)} &\overset{\text{def.}}{=} A^{(0)} x. \end{aligned} \tag{6}$$

for positive integers $d_0, \dots, d_J, d_{J+1}, J$ with $d_0 = d$ and $d_{J+1} = D$, $d_j \times d_{j+1}$-matrices $A^{(j)}$, and vectors $b^{(j)} \in \mathbb{R}^{d_{j+1}}$. The depth of (the representation equation 6 of) f is $D + 1$, the number of *hidden layers* of (the representation equation 6 of) f is D, the width $W(f)$ of (the representation equation 6 of) f is $\max_{j=0,\dots,J+1} d_j$, and the number of trainable/non-zero parameters $P(f)$ of (the representation equation 6 of) f is

$$P(f) \overset{\text{def.}}{=} \sum_{j=0}^{J+1} \|A^{(j)}\|_0 + \|b^{(j)}\|_0 \leq W(f)^2\,(D+1)$$

where $\|A^{(j)}\|_0$ (resp. $\|b^{(j)}\|_0$) counts the number of non-zero entries of $A^{(j)}$ (resp. of $b^{(j)}$). We remark that the estimate $P(f) \leq W(f)^2\,(D+1)$ is often larger for several types neural networks architectures, e.g. convolutional neural networks with downsampling layers Zhou (2020a;b).

B Detailed Model Complexities

This appendix contains a detailed version of Table 1 with fully explicit constants.

[4]This is true for several sub-Riemannian manifolds, for instance

[5]Some authors call the special case where $s = \frac{1}{L}$ an L-quasi-isometry.

Table 5: Details Complexity of the Neural Snowflakes and MLPs.

Geometry	**Net.**	**Hidden Layers (Depth -1)**	**Width**	**Theorem**
General	f	1	3	5
Bounded	f	1	$\lceil \frac{I(I-1)+2}{4} \rceil$	4
All	$\mathcal{E}$	$\mathcal{O}\left(I\left\{1+\sqrt{I\log(I)}\left[1+\frac{\log(2)}{\log(I)}\left(C_D+\frac{\log\left(I^2\,\operatorname{aspect}_2(V)\right)}{\log(2)}\right)_+\right]\right\}\right)$	$D(I-1)+\max\{d,12\}$	4 and 5

$I = \#V$ and the "dimensional constant" is $C_D \overset{\text{def.}}{=} \frac{2\log(5\sqrt{2\pi})+\frac{3}{2}\log(D)-\frac{1}{2}\log(D+1)}{2\log(2)} > 0$.

C PROOFS

This section contains derivations of our paper's main results.

C.1 LEMMATA

We can generate functions satisfying the conditions of Proposition 1 using the following lemma.

Lemma 1 (Pinelis). *A function $f : [0,\infty) \to [0,\infty)$ is continuous, concave, and monotonically increasing if and only if there is a non-negative decreasing function $g : \mathbb{R} \to \mathbb{R}$ satisfying*

$$f(t) = \int_0^t g(s)\,ds$$

for each $t \in [0,\infty)$.

Example 4. *For example, for any $0 < a \leq 1$ and $0 \leq b \leq 1-a$, the map $f : [0,\infty) \to [0,\infty)$ given by $f(t) \overset{\text{def.}}{=} t^a \ln(1+t)^b$ is continuous, concave, and monotonically increasing since*

$$f(t) = \int_0^t f(s)\left(\frac{a}{s} + \frac{b}{(1+s)\log(1+s)}\right) ds$$

and $t \mapsto f(t)\left(\frac{a}{t} + \frac{b}{(1+t)\log(1+s)}\right)$ is a non-negative decreasing function on $\mathbb{R}$.

The following helpful lemma in constructing functions satisfying the conditions of Proposition 1 from more elementary ones.

Lemma 2 ((Boyd & Vandenberghe, 2004, page 102)). *Let $C \subseteq \mathbb{R}^d$ be a non-empty convex set and $d \in \mathbb{N}_+$. If $f : C \to [0,\infty)$ is concave, and $g : [0,\infty) \to [0,\infty)$ is non-decreasing and concave, then $g \circ f : C \to \mathbb{R}$ is concave. If $C = [0,\infty)$ and if f and g are increasing, then so is $g \circ f$.*

Lemma 3 (Exponential Transformations). *Let $a > 0$. The real-valued map f on $[0,\infty)$ given for each $t \geq 0$ by $f(t) = 1 - e^{-a\,t}$ satisfies the conditions of Proposition 1.*

Proof of Lemma 3. Since $\partial_t\, f(t) = a\,e^{-at} \geq a > 0$ and $\partial_t^2\, f(t) = -a^2\,e^{-at}$, for every $t \in [0,\infty)$, then f is strictly increasing and concave on $[0,\infty)$. Noting that $f(0) = 1 - e^0 = 0$ completes the proof. □

Lemma 3 directly implies that conical combinations of functions satisfying the conditions of Proposition 1 also satisfy the condition of Proposition 1.

Lemma 4. *If $N \in \mathbb{N}_+$, $A \in (0,\infty)^N$, and $f^{(1)}, \dots, f^{(N)}$ satisfy the conditions of Proposition 1, then $f(t) \overset{\text{def.}}{=} \sum_{n=1}^N A_n\, f^{(n)}(t)$ satisfies the conditions of Proposition 1.*

Lemma 5 (Classical Snowflaking Functions). *Let $0 < \alpha \leq 1$. The real-valued map f on $[0,\infty)$ given for each $t \geq 0$ by $f(t) = t^\alpha$ satisfies the conditions of Proposition 1.*

Proof of Lemma 5. The case where α is clear; therefore, suppose that $\alpha < 1$. For every $t \in (0,\infty)$, we have that $\partial_t\, f(t) = \alpha t^{\alpha-1} > 0$ and $\partial_t^2\, f(t) = (1-\alpha)\alpha t^{\alpha-2} = (1-\alpha^2)t^{\alpha-2} < (1-\alpha)t^{\alpha-2} < 0$. Therefore, f is strictly increasing and concave on $[0,\infty)$. Noting that $f(0) = 0^\alpha = 0$ completes the proof. □

Lemma 6 (Logarithmic Snowflaking Functions)**.** *Let $0 < \beta \leq 1$. The real-valued map f on $[0, \infty)$ given for each $t \geq 0$ by $f(t) = \log(1 + |t|)^\beta$ satisfies the conditions of Proposition 1.*

Proof of Lemma 6. Suppose that $\beta = 1$. Since $\log(1 + |t|) = \log(1) = 0$ then $f(0) = 0$. For every $t \in (0, \infty)$, we have that $\partial_t f(t) = \frac{1}{1+t}0$ and $\partial_t^2 f(t) = -\frac{1}{1+t^2} < 0$. Therefore, f is strictly increasing and concave on $[0, \infty)$. By Lemma 3 $t \mapsto t^\beta$ is increasing and concave on $(0, \infty)$ and since f was strictly increasing and concave, then Lemma 2 yields the conclusion. □

C.2 Proof of Theorem 1

Proof of Theorem 1.
Step 1 - Characterization of Edges by Cardinality of Geodesic Unit Balls
Let $n \overset{\text{def.}}{=} \#V$. Fix any enumeration $V = \{v_i\}_{i=1}^n$. For each $i \in \{1, \ldots, n\}$ set

$$k_i \overset{\text{def.}}{=} \#\{w \in V : \{u_i, w\} \in E\}.$$

Note that, by definition, $k_i \in \{1, \ldots, n\}$. Since G is unweighted, then $W(\{u_i, u_j\}) = 1$ for each $i, j \in [n]$. Consequentially, for each $i, j \in [n]$ if $\{u_i, u_j\} \in E$ then $(u_i = x_1, x_2 = u_j)$ is a minimal path of length one, from u_i to u_j. Therefore, for each $i, j \in [n]$ we have that

$$\begin{aligned} d_G(u_i, u_j) &= \inf_{(x_1,\ldots,x_k)} \sum_{i=1}^{k-1} W(\{x_i, x_{i+1}\}) & (7) \\ &= \inf_{(x_1,\ldots,x_k)} \sum_{i=1}^{k-1} 1 \\ &= \inf_{(x_1,\ldots,x_k)} (k-1) \\ &= 1 & (8) \end{aligned}$$

where the infimum is taken over all paths $(u_i = x_1, \ldots, x_k = u_j)$ on G from u_i to u_j. Thus, equation 7-equation 8 imply that: for each $i \in [n]$

$$\#\{w \in V : d_{\mathcal{G}}(u_i, w) = 1\} = \#\{w \in V : \{u_i, w\} \in E\} = k_i. \tag{9}$$

By construction, for each $i \in [n]$, $r(k_i) = 1$ and there exists exactly k_i elements in $\overline{B}_{(V,d_G)}(u_i, r(k_i)) \overset{\text{def.}}{=} \{v \in V : d_G(u_i, v) \leq r(k_i)\}$. Consequentially,

$$\{u_i, u_j\} \in E \qquad \Leftrightarrow \qquad j \leq i^\star \text{ and } d_G\big(u_i, u_j\big) \leq 1 = r(k_i). \tag{10}$$

In particular, $i^\star = \#(\overline{B}_{(V,d_G)}(u_i, r(k_i)) \cap V\}$; meaning that the condition $j \leq i^\star$ can be dropped. Thus, equation 10 simplifies to: for each $i \in [n]$

$$j \leq i^\star \text{ and } d_G\big(u_i, u_j\big) \leq r(k_i) \qquad \Leftrightarrow \qquad d_G\big(u_i, u_j\big) \leq r(k_i) \tag{11}$$

Incorporating equation 11 into equation 10 allows us to further simplify to: for all $i, j \in [n]$

$$\{u_i, u_j\} \in E \qquad \Leftrightarrow \qquad d_G\big(u_i, u_j\big) \leq r(k_i) \tag{12}$$

Step 2 - Reformulation via Embeddings
Since the graph inference model $(\mathfrak{E}, \mathfrak{R})$ is universal, in the sense of Definition 1, then there exists a quasi-metric space $(\mathcal{R}, d_{\mathcal{R}}) \in \mathfrak{R}$ and an encoder $\mathcal{E} : \mathbb{R}^D \to \mathcal{R}$ in $\mathfrak{E}$ such that for each $i, j \in [n]$

$$d_{\mathcal{R}}\big(\mathcal{E}(u_i), \mathcal{E}(u_j)\big) = d_G\big(u_i, u_j\big). \tag{13}$$

Combining equation 13 and equation 12 implies that: for each $i, j \in [n]$ the pair $\{u_i, u_j\}$ belongs to E if and only if

$$d_{\mathcal{R}}\big(\mathcal{E}(u_i), \mathcal{E}(u_j)\big) = d_G\big(u_i, u_j\big) \leq 1 = r(k_i). \tag{14}$$

Combining equation 14 with equation 11 yields: for each $i, j \in [n]$

$$\{u_i, u_j\} \in E \qquad \Leftrightarrow \qquad j \leq i^\star \text{ and } d_{\mathcal{R}}\big(\mathcal{E}(u_i), \mathcal{E}(u_j)\big) \leq r(k_i).$$

This concludes the proof. □

Proof of Theorem 2. **Constructing The Isometric Embedding into a Euclidean Snowflake** Since $(V, d_{\mathcal{G}})$ is an I-point metric space. If $I = 1$, there is nothing to show; suppose, therefore, that $I > 1$. Then, (Deza & Maehara, 1990, Corollary 3) implies that for $\varepsilon^\star \overset{\text{def.}}{=} \log_2(1 + \frac{1}{I-1})/2$ there exists some $d \in \mathbb{N}_+$ and a map $\tilde{\varphi}_{\varepsilon^\star} : V \to \mathbb{R}^{d_{\varepsilon^\star}}$ satisfying

$$d_{\mathcal{G}}(u, v)^{\varepsilon^\star} = \|\varphi(v) - \varphi(u)\|. \tag{15}$$

As shown in Schoenberg (1937), equation 15 implies that the statement holds (mutatis mondanis) for any other $\varepsilon \in (0, \varepsilon^\star]$ for some other map $\tilde{\varphi}_\varepsilon : V \to \mathbb{R}^{d_\varepsilon}$ into some Euclidean space $\mathbb{R}^{d_\varepsilon}$. Fix any such ε, set $p \overset{\text{def.}}{=} 1/\varepsilon$ and let φ by any extension of $\tilde{\varphi}_\varepsilon$ defined on all of $\mathbb{R}^D$.

(Kratsios et al., 2023a, Lemma 20) implies that is an MLP with ReLU activation function $E : \mathbb{R}^D \to \mathbb{R}^d$ satisfying $\mathcal{E}(u) = \varphi(u)$ for all $u \in V$. Whence, equation 15 implies that

$$d_{\mathcal{G}}(u, v) = \|\mathcal{E}(v) - \mathcal{E}(u)\|^p.$$

NB, in the special case where $\mathcal{G}$ is a tree (Maehara, 1986, Theorem 6) we may instead set $\varepsilon^\star = 1/2$ and therefore p may be instead taken to be $p = 4$ in the above argument. **Implementing The Snowflaking Function** We now implement the snowflaking map $t \mapsto t^p$ using a neural snowflake f; i.e. with representation equation 3. Set $I = 1$, let $p = 2/\big(\log_2(1 + \frac{1}{I-1})\big)$, and consider the parameters

$$A^{(1)} = (1),\, B^{(1)} = (1),\, C^{(1)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \alpha = 1,\, \beta = 0.$$

Then, f defined by equation 3 satisfies $f(t) = t^p$. Furthermore, f has 1 hidden layer, width 3, and 4 trainable parameters and equation 15 implies that

$$d_{\mathcal{G}}(u, v) = f\big(\|\mathcal{E}(v) - \mathcal{E}(u)\|\big).$$

Since $p = 2/\big(\log_2(1 + \frac{1}{I-1})\big)$ then Example 1 show that $f(\|\cdot - \cdot\|)$ is a quasi-metric space with $2^{2/\big(\log_2(1+\frac{1}{I-1})\big)-1}$-relaxed triangle inequality.

In the special case where $\mathcal{G}$ is a tree, we have $p = 4$. Thus, the neural snowflake supports an 8-relaxed triangle inequality. □

Proof of Theorem 5. **Existence and Memorization of the Snowflake Embedding** Set $\varepsilon \overset{\text{def.}}{=} 1 - p^{-1}$ and note that $\varepsilon \in (1/2, 1)$. By (Naor & Neiman, 2012, Theorem 2), there exists $D, N > 0$ and a map $\varphi : (V, d_G^{1-\varepsilon}) \to \mathbb{R}^N$ satisfying: for every $x, u \in V$

$$d_G(x, u)^{1-\varepsilon} \le f\big(\|\varphi(x) - \varphi(u)\|\big) \le D\, d_G(x, u)^{1-\varepsilon}, \tag{16}$$

where $N = c_1 \log(\dim(G))$ and $D = c_2\, \frac{\log(\dim(G))^2}{\varepsilon^2}$, for absolute constants $c_1, c_2 > 0$ independent of G and of p. Set $f(t) \overset{\text{def.}}{=} t^p = t^{1/(1-\varepsilon)}$. Since f is monotone increasing, then applying it through the inequalities in equation 16 yields

$$d_G(x, u) \le \|\varphi(x) - \varphi(u)\| \le D^p\, d_G(x, u), \tag{17}$$

for every $x, u \in V$.

By (Kratsios et al., 2023a, Lemma 20), there exists an MLP $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}^D$ with ReLU activation function such that, for every $x \in V$ we have $\varphi(x) = \mathcal{E}(x)$. Therefore, equation 17 implies that

$$d_G(x, u) \le \|\varphi(x) - \varphi(u)\| \le D^p\, d_G(x, u), \tag{18}$$

for each $u, u \in V$. Furthermore, the depth, width, and number of trainable parameters defining $\mathcal{E}$ are as in (Kratsios et al., 2023a, Lemma 20) and a recorded in Table 5 (with abbreviated versions recorded in Table 1).

Implementing The Snowflaking Function As in the proof of Theorem 2, we now implement the snowflaking map $t \mapsto t^p$ using a neural snowflake f; i.e. with representation equation 3. Set $I = 1$, let p in equation 3 be the as in equation 17, and consider the parameters

$$A^{(1)} = (1),\, B^{(1)} = (1),\, C^{(1)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \alpha = 1,\, \beta = 0.$$

Then, f defined by equation 3 satisfies $f(t) = t^p$. Furthermore, f has 1 hidden layer, width 3, and 4 trainable parameters. □

C.3 Bounded Component

For the proof of the next result, we recall that a function $f : [0, \infty) \to [0, \infty)$ is said to be *completely monotone* if f is continuous on $[0, \infty)$, smooth on $(0, \infty)$, and its derivatives satisfy the following alternating sum property

$$(-1)^n \partial_t^n f(t) \geq 0 \tag{19}$$

for every $t > 0$ and every $n \in \mathbb{N}_+$. See (Widder, 1941, Chapter IV) for a detailed study of completely monotone functions and several examples thereof.

In particular, this and Proposition 1 imply that $-f$ is monotonically increasing and concave, therefore the map, $\bar{f}$, defined for $t \geq 0$ by $\bar{f}(t) \overset{\text{def.}}{=} f(0) - f(t)$ produces a well-defined snowflake metric $d_{\bar{f}} \overset{\text{def.}}{=} \bar{f}(\| \cdot - \cdot \|)$. Since f is completely monotone then it is monotonically decreasign on $[0, \infty)$ and bounded below by 0; whence, $\bar{f}(t)$ is monotonically increasing and contained in $[0, f(0)]$. Therefore, $d_{\bar{f}}$ is *bounded* between $[0, f(0)]$. We will show that the neural snowflake can generate a snowflake metric which interpolates any bi-Lipschitz embedding into such a space.

Before proving Theorem 4 we note that it holds under the following alternative assumption to Assumption 1. Intuitively, this assumptions state that the map f, in Assumption 1 can be taken to be the moment-generating function (MGF) of some probability measure on $[0, \infty)$.

Assumption 2 (Alternative to Assumption 1: Latent MGD Geometry)**.** *There are $d, D \in \mathbb{N}_+$, a latent feature map $\Phi : \mathbb{R}^D \to \mathbb{R}^d$, and a Borel probability measure $\mathbb{P}$ on $[0, \infty)$ whose MGF $f(t) = \mathbb{E}_{X \sim \mathbb{P}}[e^{-t\,X}]$ exists for all $t \geq 0$, is non-constant, and satisfies*

$$d_{\mathcal{G}}(x, y) = \bar{f}\big(\|\Phi(x) - \Phi(y)\|\big),$$

where $\bar{f}(t) \overset{\text{def.}}{=} f(0) - f(t)$.

Both Assumptions 1 and 2 are special cases of the following more general assumption.

Assumption 3 (Kernel or Moment-Generating Priors)**.** *Suppose that $f : [0, \infty) \to [0, \infty)$ is non-constant and either:*

(i) $f(t) = \int_0^\infty e^{-t\,u}\, \mu(du)$ for some finite Borel measure μ on $[0, \infty)$,

(ii) For each $k \in \mathbb{N}_+$, the map $K_f : (x, y) \in \mathbb{R}^k \times \mathbb{R}^k \mapsto f(x^\top y) \in \mathbb{R}$ is a positive-definite kernel on $\mathbb{R}^k$.

Define $\bar{f}(t) \overset{\text{def.}}{=} f(0) - f(t)$.

The following result implies, and generalizes, Theorem 4, to bi-Lipschitz embeddings.

Proposition 3 (Embeddings into Bounded Metric Spaces - Bi-Lipschitz Version)**.** *Let $D, d \in \mathbb{N}_+$, f satisfy Assumption 3, and G be a finite weighted graph with $V \subset \mathbb{R}^d$. Then, $\| \cdot - \cdot \|_{\bar{f}}$ defines a bounded metric on $\mathbb{R}^d$ and for every (s, L)-bi-Lipschitz embedding $\Phi : (V, d_G) \to (\mathbb{R}^d, \| \cdot \|_{\bar{f}})$ there is a neural snowflake $(\mathcal{E}, f)$ satisfying*

$$\|\Phi(v) - \Phi(u)\|_f = \|\mathcal{E}(v) - \mathcal{E}(u)\|_{\bar{f}}$$

for every $u, v \in V$. In particular, for each $u, v \in V$ we have

$$s\, d_G(v, u) \leq \|\mathcal{E}(v) - \mathcal{E}(u)\|_f \leq s\, L\, d_G(v, u).$$

Furthermore, the depth, width, and number of trainable parameters of $\mathcal{E}$ and of f are as in Table 1.

Proof of Theorem 3. **Rephrasing as completely monotone functions** Suppose that Assumption 3 holds. The Hausdorff-Bernstein-Widder theorem, see (Widder, 1941, Theorem IV.12a), implies that f is completely monotone. Alternatively, suppose that Assumption 3 (ii) holds then Schoenberg's theorem, see[6] (Schoenberg, 1938, Theorem 3), implies that f is completely monotone.

Since f is defined on all of $[0, \infty)$ then equation 19 implies that $f(t) \geq 0$ for all t whence takes non-negative valued. Likewise equation 19 implies that $\partial_t\, f(t) \leq 0$ therefore f is non-increasing;

[6]See Phillips et al. (2019) for more general version.

whence $\bar{f}$ is bounded in $[0, f(0)]$. Finally, equation 19 implies that $\partial_t^2 f(t) \geq 0$ therefore f is convex; thus $\bar{f}$ is concave since $\partial_t^2 \bar{f} \leq 0$. Whence Proposition 1 implies that $\| \cdot - \cdot \|_{\bar{f}}$ is a metric on $\mathbb{R}^D$.

Interpolating $\mathcal{E}$ and $\bar{f}$

Enumerate $V = \{x_i\}_{i=1}^I$ where $I \overset{\text{def.}}{=} \# I$. Consider an (s, L)-bi-Lipschitz embedding $\Phi : (V, d_G) \to (\mathbb{R}^d, \| \cdot \|_f)$. By (Kratsios et al., 2023a, Lemma 20), there exists an MLP $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^d$ with ReLU activation function satisfying

$$\Phi(v) = \mathcal{E}(v) \tag{20}$$

for each $v \in V$. Moreover, the depth, width, and number of trainable parameters determining E are as in (Kratsios et al., 2023a, Lemma 20) and are recorded in Table 5 (and abbreviated in Table 1).

Since Φ is bi-Lipschitz then it is injective; whence $\{\|\Phi(x_i) - \Phi(x_j)\|\}_{i,j=1}^I$ has exactly as many points as $\{\|x_i - x_j\|\}_{i,j=1}^I$. By symmetry of the Euclidean metric, observe that the number $\tilde{I}$ of elements in set $\{\|\Phi(x_i) - \Phi(x_j)\|\}_{i,j=1}^I \cup \{0\}$ is at-most $I(I-1)/2 + 1$ elements; which we sort and enumerate $\{\|x_i - x_j\|\}_{i,j=1}^I \cup \{0\}$ by $\{t_i\}_{i=1}^{\tilde{I}}$. By the (McGlinn, 1978, Corollary on page 215) there exists a unique exponential sum $Y(t) = \sum_{i=1}^{\lceil \tilde{I}/2 \rceil} \beta_i \, e^{-\alpha_i \, t}$ satisfying

$$Y(t_i) = f(t_i) \tag{21}$$

for every $i = 0, \dots, \tilde{I}$, and in particular $Y(0) = f(0)$ since $0 \in \{t_i\}_{i=1}^{\tilde{I}}$. Since $\bar{f}(t) = f(0) - f(t)$ for all $t \geq 0$, then equation 21 implies that

$$\begin{aligned}
\bar{f}(t_i) =& f(0) - f(t_i) \qquad (22) \\
=& Y(0) - Y(t_i) \\
=& \left(\sum_{i=1}^{\lceil \tilde{I}/2 \rceil} \beta_i \, e^{-\alpha_i \, 0} \right) - \left(\sum_{i=1}^{\lceil \tilde{I}/2 \rceil} \beta_i \, e^{-\alpha_i \, t_i} \right) \\
=& \sum_{i=1}^{\lceil \tilde{I}/2 \rceil} \left(\beta_i \, e^{-\alpha_i \, 0} - \beta_i \, e^{-\alpha_i \, t_i} \right) \\
=& \sum_{i=1}^{\lceil \tilde{I}/2 \rceil} \left(\beta_i \, 1 - \beta_i \, e^{-\alpha_i \, t_i} \right) \\
=& \sum_{i=1}^{\lceil \tilde{I}/2 \rceil} \beta_i \left(1 - e^{-\alpha_i \, t_i} \right) \overset{\text{def.}}{=} \bar{Y}(t_i), \qquad (23)
\end{aligned}$$

for all $i = 1, \dots, \tilde{I}$. In particular, equation 22-equation 23, the definition of the t_i, and the memorization/interpolation guarantee in equation 20 implies that

$$f\big(\|\Phi(x_i) - \Phi(x_j)\|\big) = \bar{Y}\big(\|\mathcal{E}(x_i) - \mathcal{E}(x_j)\|\big) = \bar{Y}\big(\|\mathcal{E}(x_i) - \mathcal{E}(x_j)\|\big) \tag{24}$$

for $i, j = 1, \dots, I$ (note $0 = \|\mathcal{E}(x_1) - \mathcal{E}(x_1)\|$ so $f(0) = Y(0)$ is implied by equation 24). Therefore, for each $i, j = 1, \dots, I$ we have

$$\begin{aligned}
\|\mathcal{E}(x_i) - \mathcal{E}(x_j)\|_{\bar{f}} =& \|\Phi(x_i) - \Phi(x_j)\|_{\bar{f}} \\
\overset{\text{def.}}{=}& \bar{f}\big(\|\Phi(x_i) - \Phi(x_j)\|\big) \\
=& \bar{Y}\big(\|\Phi(x_i) - \Phi(x_j)\|\big) \\
=& \|\Phi(x_i) - \Phi(x_j)\|_{\bar{Y}}.
\end{aligned}$$

It remains to show that $\bar{Y}$ can be implemented by a map, which we denote by $\tilde{f}$, with representation equation 3.

Implementing The Exponential Sum In the notation of equation 3, set $I = 1$ and consider

$$A^{(1)} = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_M \end{pmatrix}, \; B^{(1)} = (\beta_1 \dots \beta_M), \; C^{(1)} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \; d = (0), \; \alpha_1 = 0 = \beta_0. \tag{25}$$

Then, the map $\tilde{f}$ defined as in equation 3 with parameters equation 25 is

$$\tilde{f}(t) = \left(B^{(1)}\, \sigma_{0,0}(A^{(1)}t)C^{(1)}\right)^{1+|0|} = \sum_{i=1}^{M} \beta_i \left(1 - e^{-\alpha_i\, t}\right) = \bar{Y}(t)$$

for every $t \geq 0$. Tallying parameters in $\tilde{f}$, shows that $\tilde{f}$ has $3M + 1$ non-zero parameters, 1 hidden layer, and width M; as recorded in Table 5 (and abbreviated in Table 1). □

C.4 Detailed Euclidean Comparisons

Together, the following Propositions imply Theorem 3.

Proposition 4 (All Embeddings Implementable MLPs are Implementable by a Neural Snowflakes)**.** *For any $d, D, P \in \mathbb{N}_+$, $0 \leq s \leq L$. For any weighted finite graph $G = (E, V, W)$ with $V \subset \mathbb{R}^D$: if there is an MLP $\tilde{\mathcal{E}} : \mathbb{R}^D \to \mathbb{R}^d$ with P non-zero parameters satisfying*

$$s\, d_G(v, u) \leq \|\tilde{\mathcal{E}}(v) - \tilde{\mathcal{E}}(u)\| \leq L\, d_G(v, u)$$

for each $v, u \in V$. There is a pair of an MLP $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^d$ and a neural snowflake f satisfying

$$s\, d_G(v, u) \leq \|\tilde{\mathcal{E}}(v) - \tilde{\mathcal{E}}(u)\|_f \leq L\, d_G(v, u)$$

for each $v, u \in V$. Furthermore, the total number of non-zero parameters in $\mathcal{E}$ and f is $P + 4$.

Proof of Proposition 4. Set $\mathcal{E} \overset{\text{def.}}{=} \tilde{\mathcal{E}}$; in particular, $\mathcal{E}$ is defined by P parameters.

It remains to show that f can implement the identity function. In the notation of equation 3, set $I = 1$, and consider the parameters

$$A^{(1)} = (1),\ B^{(1)} = (1),\ C^{(1)} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\ \alpha = 1,\ \beta = 0.$$

Then, f defined by equation 3 satisfies $f(t) = t$. Furthermore, f has 1 hidden layer, width 3, and 4 trainable parameters. The conclusion now follows since $\tilde{\mathcal{E}}$ was assumed to implement an (s, L)-bi-Lipschitz embedding of (V, d_G) into $(\mathbb{R}^d, \|\cdot\|)$. □

Proposition 5 (Neural Snowflakes can Implement Isometric Embeddings which MLPs Cannot)**.** *For any $d, D \in \mathbb{N}_+$ there exists a fully-connected weighted graph $G = (V, E, W)$ with $V \subset \mathbb{R}^D$ which:*

(i) Cannot be isometrically embedded into $(\mathbb{R}^n, \|\cdot\|)$ for any $n \leq d$,

(ii) There exists a neural snowflake $(\mathcal{E}, f)$ with $\mathcal{E} : \mathbb{R}^D \to \mathbb{R}^d$ satisfying: for each $x, u \in V$

$$\|\mathcal{E}(x) - \mathcal{E}(u)\|_f = d_G(x, u).$$

Proof of Proposition 5. Fix $d, D \in \mathbb{N}_+$, $\alpha = 1/2$, and $p \overset{\text{def.}}{=} \alpha$. Then, (Le Donne et al., 2018, Theorem 1.1) implies that there exists some $N \in \mathbb{N}_+$ such that for any metric space (V, d_G) with at-least N points, the $1/2$-snowflake $(V, d_G^{1/2})$ cannot be isometrically embedded in $(\mathbb{R}^d, \|\cdot\|)$. If $d > 1$, suppose that for some $n < d$, $(V, d_G^{1/2})$ admitted an isometric embedding $\varphi_1 : (V, d^{1/2}) \to (\mathbb{R}^n, \|\cdot\|)$ then, since the map $\varphi_2 : \mathbb{R}^n \to \mathbb{R}^d$ given by $z \mapsto (z_1, \ldots, z_n, 0, \ldots, 0)$, and since the composition of isometries is again an isometry, then $\varphi \overset{\text{def.}}{=} \varphi_2 \circ \varphi_1$ would define an isometry from $(V, d_G^{1/2})$ to $(\mathbb{R}^d, \|\cdot\|)$; which is a contradiction. This, yields (i).

Let us now show (ii). Set $p = 2$ and let V be any finite subset of $\mathbb{R}^D$ with N points. Consider the fully-connected graph $G = (V, E, W)$ with edge weights given by

$$W(x, u) \overset{\text{def.}}{=} \|x - u\|^{1/2}.$$

By construction the graph geodesic distance d_G on G satisfies $d_G(x, u) = W(x, u)$ for every $x, u \in V$. Furthermore, again by construction, for every $x, u \in V$ we have

$$d_G(x, u) = W(x, u) = \|x - u\|^{1/2} = \|\mathcal{E}(x) - \mathcal{E}(u)\|_f$$

where $f(t) = |t|^{1/2}$ and $\mathcal{E}(x) = x$. Applying (Kratsios et al., 2023a, Lemma 20), we find that there exists an MLP with ReLU activation function $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ with, depth, width, and number of non-zero parameters specified therein, satisfying $\Phi(x) = (x)$ for every $x \in V$. □

C.5 Proof of Impossibility Results - Proposition 2

We now prove Proposition 2 by showing that the graph depicted in Figure 1 cannot be embedded isometrically into any Riemannian manifold, as we now show.

Proof of Proposition 2. Fix $D \in \mathbb{N}_+$ and let $V \subseteq \mathbb{R}^D$ be any 5-point subset; whose points we list by $V = \{A, B, C, D, E\}$. Define the set of edges

$$E \overset{\text{def.}}{=} \{\{A, E\}, \{A, D\}, \{E, B\}, \{B, D\}, \{D, C\}\}$$

and consider the graph $\mathcal{G} \overset{\text{def.}}{=} (V, E)$. It is easy to see that $\mathcal{G}$ is connected; thus, the shortest path (geodesic) distance $d_{\mathcal{G}}$ on V is well-defined. Furthermore, there are unique shortest paths joining C to A and C to B; which we denote by $[C : A]$ and $[C : B]$ respectively; these are given by the ordered tuples (ordered pairs in this case)

$$[C : A] \overset{\text{def.}}{=} \big(\{C, D\}, \{D, A\}\big) \text{ and } [C : B] \overset{\text{def.}}{=} \big(\{C, D\}, \{D, B\}\big). \tag{26}$$

We now argue by contradiction.

Suppose that there exists a complete and connected smooth Riemannian manifold $(\mathcal{R}, g)$ and an isometric embedding $\varphi : (V, d_{\mathcal{G}}) \to (\mathcal{R}, d_g)$; where, d_g denotes the shortest path (geodesic) distance on $(\mathcal{R}, g)$. Since $(\mathcal{R}, g)$ is complete (as a metric space) and connected, then the Hopf-Rinow Theorem (Jost, 2017, Theorem 1.7.1) implies that each pair of points in $\mathcal{R}$ can be joined by a distance minimizing geodesic, i.e. it is geodesically complete (Jost, 2017, Definition 1.7.1). Therefore, there exists a pair of geodesics $\gamma_{[C:A]} : [0, 1] \to \mathcal{R}$ and $\gamma_{[C:B]} : [0, 1] \to \mathcal{R}$ satisfying

$$\gamma_{[C:i]}(0) = \varphi(C) \text{ and } \gamma_{[C:i]}(1) = \varphi(i) \tag{27}$$

for $i \in \{A, B\}$. Since φ is an isometric embedding and $\{C, D\} \in [C : A] \cap [C : B]$ then equation 26 implies that there is some $0 < t_2 < 1$ for which

$$\gamma_{[C:A]}(t_2) = \gamma_{[C:B]}(t_2) = \varphi(D). \tag{28}$$

Now, equation 27 and the local uniqueness of geodesics in $(\mathcal{R}, g)$ about any point, in particular about $\varphi(D)$ (see (Jost, 2017, Theorem 1.4.2)) imply that there is some $\varepsilon > 0$ such that

$$\gamma_{[C:A]}(s) = \gamma_{[C:B]}(s), \tag{29}$$

for all $t_2 - \varepsilon \leq s \leq t_2 + \varepsilon$. Now since $A \neq B$ and since φ is injective then $\varphi(A) \neq \varphi(B)$. However, equation 27 and equation 29 cannot simultaneously hold for $t_2 < s < t_2 + \varepsilon$; thus we have a contradiction. Consequentially, a pair $\big(\varphi, (\mathcal{R}, d_{\mathcal{R}})\big)$ cannot exist. $\square$

D Details on Examples

This appendix contains further details, explaining derivations of particular examples within the paper's main text.

Example - Details 1 (Details for Example 2)**.** *This follows from (Phillips et al., 2019, Proposition 3.2) and the Hausdorff-Bernstein-Widder theorem, see (Widder, 1941, Theorem IV.12a), together states that $f \circ u$ satisfies Assumption 2 if f does and if $u : t \mapsto \frac{t}{a+t}$ is a Bernstein function, i.e. a continuous function $f : [0, \infty) \to [0, \infty)$ which is smooth on $(0, \infty)$ and whose derivatives satisfy $(-1)^k \partial_t^k f(t) \leq 0$ for all $t \geq 0$ and all $k \in \mathbb{N}_+$. A long list of Bernstein functions which can be found in the several tables in (Schilling et al., 2012, Section 16.2). For instance, the map $t \mapsto (1 + \sum_{k=1}^K |t|^{r_k})^{-1}$ is one a Bernstein function, see (Schilling et al., 2012, Corollary 6.3), and $f(t) = |t|^\beta$ satisfies Assumption 2.*

Example - Details 2 (Details for Example 3)**.** *This holds, by arguing as in the previous example, since $t \mapsto e^{-b\,t}$ satisfies Assumption 1 and since $t \mapsto \frac{-(t-1)}{log(t)}$ is a Bernstein function, by (Schilling et al., 2012, Corollary 6.3).*

E Discrete Edge Sampling and the Discrete Differentiable Graph Module

Most latent graph inference models require generating a discrete graph based on a similarity measure between latent node representations. The discrete Differentiable Graph Module (dDGM) (Kazi et al., 2022) has served as a source of inspiration for numerous studies in the field of latent graph inference (Sáez de Ocáriz Borde et al., 2023c;b; Battiloro et al., 2023). It generates a sparse k-degree graph using the Gumbel Top-k trick (Kool et al., 2019), a stochastic relaxation of the kNN rule, to sample edges from the probability matrix $\mathbf{P}^{(l)}(\mathbf{X}^{(l)};\mathbf{\Theta}^{(l)},T)$, where each entry corresponds to

$$p_{ij}^{(l)}=\exp(-\varphi(\hat{\mathbf{x}}_i,\hat{\mathbf{x}}_j,T)). \tag{30}$$

where T is a learnable temperature parameter, $\hat{x}$ are latent node feature representation, and φ is some similarity measure. In practice, the main similarity measure used in Kazi et al. (2022) was to compute the distance based on the features of two nodes in the graph embedding space, which was assumed to be Euclidean. Based on

$$\operatorname{argsort}(\log(\mathbf{p}_i^{(l)})-\log(-\log(\mathbf{q}))) \tag{31}$$

where $\mathbf{q}\in\mathbb{R}^N$ is uniform i.i.d in the interval $[0,1]$, we can sample the edges

$$\mathcal{E}^{(l)}(\mathbf{X}^{(l)};\mathbf{\Theta}^{(l)},T,k)=\{(i,j_{i,1}),(i,j_{i,2}),...,(i,j_{i,k}):i=1,...,N\}, \tag{32}$$

where k is the number of sampled connections using the Gumbel Top-k trick. This sampling approach follows the categorical distribution $\frac{p_{ij}^{(l)}}{\Sigma_r p_{ir}^{(l)}}$ and $\mathcal{E}(\mathbf{X}^{(l)};\mathbf{\Theta}^{(l)},T,k)$ is represented by the unweighted adjacency matrix $\mathbf{A}^{(l)}(\mathbf{X}^{(l)};\mathbf{\Theta}^{(l)},T,k)$. Note that including noise in the edge sampling approach will result in the generation of some random edges in the latent graphs which can be understood as a form of regularization.

F Graph Learning Algorithms: Training and Backpropagation

The optimization of the baseline node feature learning component in the architecture, that is, the standard GNN part relies on the loss of the downstream task. In particular, for classification, the cross-entropy loss is utilized. However, it is also necessary to update the parameters of graph learning modules such as the DGM (Kazi et al., 2022), the DCM (Battiloro et al., 2023), and the neural snowflake. To accomplish this, we implement a compound loss that provides incentives for edges contributing to accurate classification while penalizing edges that lead to misclassification. We introduce a reward function

$$\delta\left(y_i,\hat{y}_i\right)=\mathbb{E}(ac_i)-ac_i.$$

The aforementioned disparity is calculated as the difference between the mean accuracy of the ith sample and the present accuracy of the prediction. Here, y_i and $\hat{y}_i$ represent the true and predicted labels, respectively, while ac_i is assigned a value of 1 if $y_i=\hat{y}_i$, and 0 otherwise. The loss function for graph learning is formulated in terms of the reward function:

$$L_{GL}=\sum_{i=1}^{N}\left(\delta\left(y_i,\hat{y}_i\right)\sum_{l=1}^{l=L}\sum_{j:(i,j)\in\hat{\varphi}^{(l)}}\log p_{ij}^{(l)}\right), \tag{33}$$

and it approximates the gradient of the expectation $\mathbb{E}_{(\mathcal{G}^{(1)},...,\mathcal{G}^{(L)})\sim(\mathbf{P}^{(1)},..,\mathbf{P}^{(L)})}\sum_{i=1}^{N}\delta\left(y_i,\hat{y}_i\right)$. The expectation $\mathbb{E}(ac_i)^{(t)}$ is calculated based on

$$\mathbb{E}(ac_i)^{(t)} = \beta\mathbb{E}(ac_i)^{(t-1)} + (1-\beta)ac_i, \tag{34}$$

with $\beta = 0.9$ and $\mathbb{E}(ac_i)^{(t=0)} = 0.5$. For further details refer to Kazi et al. (2022) and Sáez de Ocáriz Borde et al. (2023c).

G Neural Snowflake for Latent Graph Inference Algorithms

This appendix includes Algorithm 1, 2, and 3. They summarize how learned latent graphs are incorporated into a standard GNN pipeline, how the dDGM samples graph edges, and how the neural snowflake architecture computes similarities between latent node representations, respectively. Superscripts are employed to denote layer-specific quantities, while subscripts are utilized for indices.

Algorithm 1: Node Level Prediction leveraging Inferred Latent Graph (Forward Pass)

```
Require: $\mathbf{X}, \mathbf{A}$                                      ▷ Node Features and Adjacency Matrix
  return $\mathbf{Y}$                                                   ▷ Predicted Node Labels
  $\mathbf{X}^{(0)} \leftarrow \mathbf{X}$
  $\hat{\mathbf{A}} \leftarrow \texttt{DGM}(\mathbf{X}^{(0)}, \mathbf{A})$                               ▷ Refer to Algorithm 2
  For $l = 1$ to $L$
      $\mathbf{X}^{(l)} \leftarrow \texttt{GNN}^{(l)}(\mathbf{X}^{(l-1)}, \hat{\mathbf{A}})$                     ▷ GNN diffusion layers
  end
  $\mathbf{Y} \leftarrow \texttt{MLP}(\mathbf{X}^{(L)})$                                 ▷ Node level prediction
```

Algorithm 2, is a mild modification of the differentiable graph model of Kazi et al. (2022). Briefly, it allows the GNN to update its graph in a differentiable manner.

Algorithm 2: Discrete Differentiable Graph Module (modified based on Kazi et al. (2022))

```
Require: $\mathbf{X}, \mathbf{A}$                                      ▷ Node Features and Adjacency Matrix
  return $\hat{\mathbf{A}}$                                                   ▷ Latent Graph
  $\hat{\mathbf{X}} \leftarrow f(\mathbf{X}, \mathbf{A})$                                        ▷ Transform node features
  $s_{ij} \leftarrow \texttt{Neural Snowflake}(\hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}_\mathbf{j})$      ▷ Compute similarity measures. Refer to Algorithm 3
  $p_{ij} \leftarrow g(s_{ij})$                  ▷ Compute edge sampling probabilities based on similarities
  For $i = 1$ to $N$
      $\mathbf{q} \sim U(0,1)$                                              ▷ Uniform i.i.d.
      $\mathbf{j}_{\{k\}} = \texttt{argtopk}(\log \mathbf{p}_i - \log(-\log(\mathbf{q}_i)))$
      $\hat{a}_{ij} = \begin{cases} 1 & j \in \mathbf{j}_{\{k\}} \\ 0 & \text{otherwise} \end{cases}$
  end
  $\hat{\mathbf{A}} \leftarrow \hat{a}_{ij}$                             ▷ Discrete Latent Unweighted Graph Prediction
```

Algorithm 3 describes how the neural snowflake processes node-level features to distances.

H Computational Implementation Details

In this appendix, we present additional information regarding the practical implementation of trainable snowflake activations and neural snowflakes, beyond the theoretical foundation discussed in the main text. We address certain instabilities encountered during the training process and propose methods to mitigate them. Our objective is to provide a deeper understanding of our findings, hoping that this will contribute to the advancement of neural snowflakes in future iterations.

Hardware and Symbolic Matrices. In line with previous work, for most of the experiments, we utilized GPUs such as the NVIDIA Tesla T4 Tensor Core with 16 GB of GDDR6 memory, NVIDIA P100 with 16 GB of CoWoS HBM2 memory, or NVIDIA Tesla K80 with 24 GB of GDDR5 memory.

Algorithm 3: Neural Snowflake Processing

```
Require: $\hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}_\mathbf{j}$                          ▷ Two Node Features Vectors
  return $s_{ij}$                                              ▷ Distance similarity measure
  $t \leftarrow ||\hat{\mathbf{x}}_\mathbf{i} - \hat{\mathbf{x}}_\mathbf{j}||_2$                                      ▷ Euclidean Distance
  $t^{(0)} \leftarrow t$
  For $l = 1$ to $I$
    $\hat{t}^{(l-1)} \leftarrow A^{(l)} t^{(l-1)}$                                 ▷ Linear Projection
    $\Sigma^{(l)} \leftarrow \sigma^{(l)}(\hat{t}^{(l-1)})$             ▷ Trainable Snowflake Activation (Equation 2)
    $t^{(l)} \leftarrow B^{(l)} \Sigma^{(l)} C^{(l)}$                              ▷ Linear Projections
  end
  $s_{ij} \leftarrow t_I^{1+|p|}$                                             ▷ Quasi-metric
```

These GPUs have limited memory capacities that are easily surpassed during backpropagation when dealing with datasets other than Cora and CiteSeer. One of the primary computational limitations of the Differentiable Graph Module and other latent graph inference techniques is the necessity to compute distances between latent representations for all nodes in order to generate the latent graph. While the discrete graph sampling method utilized by dDGM offers improved computational efficiency compared to its continuous counterpart, cDGM, due to the creation of sparse graphs that lighten the burden on convolutional operators, we encounter memory constraints when dealing with graph datasets containing a large number of nodes, on the order of 10^4 nodes. To determine whether a connection should be established, we must calculate distances between all points starting from a pointcloud. However, this poses a challenge as the computational complexity scales quadratically with the number of nodes in the graph. Consequently, as the graph size increases, the computation quickly becomes intractable. To address the issue of potential memory overflows, we adopt Kernel Operations (KeOps) (Charlier et al., 2021), as recommended by previous studies on latent graph inference. KeOps enables us to perform computations on large arrays by efficiently reducing them based on a mathematical formula.

Trainable Snowflake Activation Preliminary Experiments: Stability and Initialization. Although in equation 1 the α, β and γ parameters are introduced as trainable parameters for the sake of generality, we find that during backpropagation this exponential learnable terms can lead to instabilities, hence, we set them all to $\alpha = \beta = \gamma = 1$. In future research it could be explored how to stabilize these and whether this additional flexibility proves advantageous empirically. We run some initial experiments to assess the stability of the trainable snowflake activation. In particular we work with the homophilic benchmarks Cora and Citeseer and we incorporate the snowflake activation to dDGM with Euclidean space and which feeds its infered latent graph to a Graph Convolutional Network of 3 layers with ELU activation functions. That is, the snowflake activation takes as input the euclidean distance between latent graph nodes computed by the dDGM.

We observe that in this particular configuration the p parameter that controls how much the quasi-metric deviates from being a metric can be a problematic parameter during training. Interestingly, this does not seem to be the case, when running synthetic experiments with a full neural snowflake (see Appendix I). This could be attributed to the fact that in the literature in this setup the dDGM is trained with a learning rate of 10^{-2}, which is too aggressive to update the p parameter. For Cora the dDGM leveraging Euclidean space and using the original dataset graph as inductive bias achieves an accuracy of 82.40 ± 3.22 (mean $\pm$ standard deviation), using the off-the-shelf snowflake activation we obtain 40.33 ± 11.62 which presents a clear drop in performance. The observed large standard deviation indicates that the model frequently becomes trapped in local minima during optimization. This can be mitigated by either setting $p = 0$ and learning a metric, or by using a different optimizer with a lower learning rate (10^{-4}, for example) to update the parameter p. This configurations lead to accuracies of 85.48 ± 2.74 and 86.11 ± 3.72 respectively for Cora, which clearly surpass the performance using Euclidean space. In the case of Citeseer using the original dDGM we get an accuracy of 73.40 ± 1.64, using a snowflake activation with $p = 0$ we obtain 73.85 ± 2.34, and using a learnable p with a learning rate of 10^{-4} we achieve 74.40 ± 2.08. Note that for the cases in which we learn p with a different optimizer, p is initialized at $p = 10^{-8}$ to start from a metric and slowly learn a quasi-metric. These experiments show that the quasi-metric relaxation can provide the snowflake activation with additional representation capabilities and flexibility but should be used with care. For all the rest of experiments in this paper we learn a quasi-metric when implementing the snowflake activation but use a slower optimizer for the p parameter. In these experiments the coefficients C_1, C_2, and C_3 were initialized to 1. We use the absolute value function during training to ensure they stay non-negative. Note that these experiments were conducted using the Gumbel Top-k trick edge sampling algorithm with $k = 7$ and an embedding dimensionality of 4.

Neural Snowflakes. From a computational perspective, it is more reliable to assign fixed coefficients a and b, instead of backpropagating through them. Specifically, we set $a = 1$ and $b = 1$ for all experiments. Matrices A, B, and C weights are initialized by sampling from a uniform distribution ranging from 0 to 1 and normalized by the matrix dimensions. For example, in the case of A, the weights are sampled from a distribution between 0 and $1/(d_{A1}d_{A2})$, where d_{A1} is the number of rows and d_{A2} is the number of columns of the matrix. We experimentally observe that other initializations such as drawing the weights from a Gaussian or using Xavier initialization can lead to instabilities and exploding numbers in the forward pass. To guarantee non-negativity of all weights throughout training, we apply an absolute function activation to the weights. p is initialized to $p = 1e - 8 \approx 0$, so that we start from a metric space and gradually learn a quasi-metric space. Furthermore, it should be noted that the learnable coefficient p is exclusively applied in the last layer of the neural snowflake model. This design choice allows us to track the coefficient C, which represents the relaxation of the triangle inequality. In our synthetic experiments, we employ a readily available neural snowflake model, while for latent graph inference, we utilize a weighted skip connection. The neural snowflake model takes the Euclidean distance between latent representations as input. We observe that employing a skip connection and initiating training with an almost Euclidean metric proves beneficial, particularly during the early stages of training.

I Experimental Results Supplementary Material

Within this appendix, we provide supplementary information regarding the experimental results discussed in the main text. This encompasses details about the train and test splits, the precise training configurations applied, as well as supplementary visual representations illustrating the evolution of the model training process, along with additional experiments and their corresponding results.

Synthetic Experiments. All models are trained using the Adam optimizer with a learning rate of 1×10^{-4}, for 40 epochs and with a batch size of 1,000. After approximately 20 epochs, we notice a tendency for learning to reach a plateau, particularly when dealing with neural snowflakes. As mentioned in the main text, for our experimental analysis, we concentrate on fully connected graphs. In this setup, the node coordinates are sampled randomly from a multivariate Gaussian distribution within a 100-dimensional hypercube in Euclidean space, represented as $\mathbb{R}^{100}$. The graph weights are determined based on the metrics outlined in Table 2. The training sets have 4,000,000 data points and the test sets 10,000. We observe very little discrepancy between the performance of the models for training and testing sets.

The MLP model, which works in isolation and aims at approximating the metrics using Euclidean space $\|\text{MLP}(\mathbf{x}) - \text{MLP}(\mathbf{y})\|$, has a total of 5422 parameters, and its composed of 10 linear layers with 20 hidden dimensions and ReLU activation functions. The neural snowflake learning the metric on $\mathbb{R}^2$: $f(\|\text{MLP}(\mathbf{x}) - \text{MLP}(\mathbf{y})\|)$, is composed of 2 layers with hidden dimension of 20. Note that in Section 3.1 we define $A^{(i)}$ as a $\tilde{d}_i \times d_{i-1}$ matrix, $B^{(i)}$ as a $d_i \times \tilde{d}_i$-matrix. However, in this experiments we set $\tilde{d}_i = d_i = 20$. The MLP used alongside the neural snowflake to project the node features in $\mathbb{R}^{100}$ to $\mathbb{R}^2$ consists of 5 layers with hidden dimension 20 with a total of 3,322 model parameters and also uses ReLU activations. Lastly, for the third case in which the neural snowflake learns directly in $\mathbb{R}^{100}$: $f(\|\mathbf{x} - \mathbf{y}\|)$ we reuse the same neural snowflake architecture as before with a total of 847 learnable parameters.

Additionally, we provide some plots of the training loss function evolution during learning for the synthetic graph embedding experiments in Figure 2. As we can observe from the plots the neural snowflakes learning in $\mathbb{R}^{100}$ converge faster, whereas neural snowflakes in $\mathbb{R}^2$ tend to get stuck in local minima at the beginning of training and eventually achieve comparable performance to their higher-dimensional counterparts. On the contrary, MLPs operating in $\mathbb{R}^2$ demonstrates significantly poorer performance, they achieve a higher loss with a higher variance during the training process. In the main text we provided results for the synthetic experiments in terms of the test set performance; we additionally provide results for the training set in Table 6.

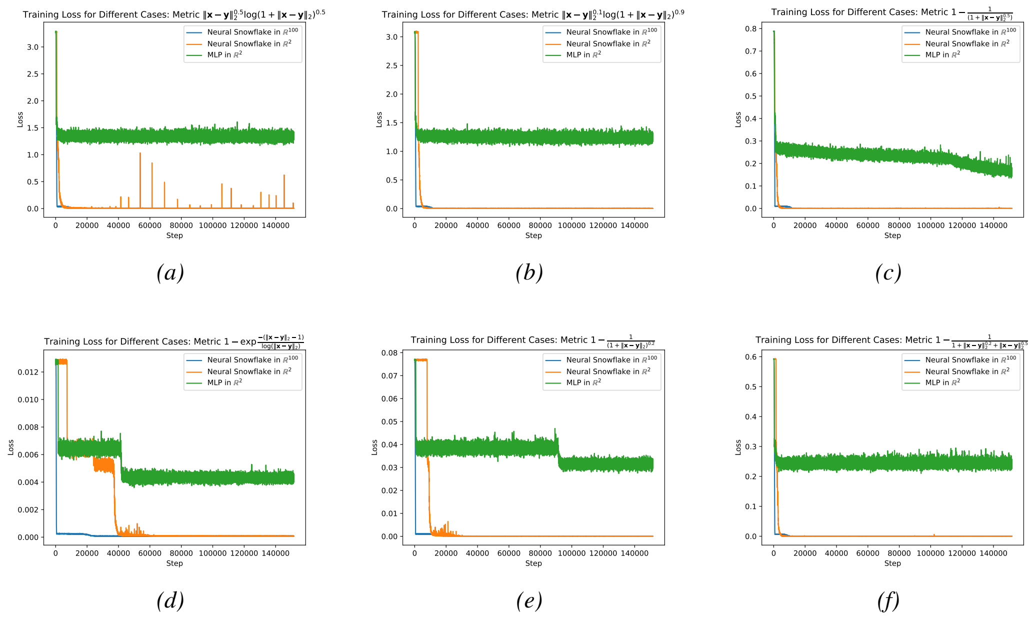

Figure 2: Training losses for synthetic graph embedding experiments. We compare using Euclidean space for encoding the weighted graphs to using snowflake quasi-metric spaces.

Table 6: Results for synthetic graph embedding experiments, mean square error for training set. The Neural Snowflake models are able to learn the metric better with substantially lesser number of model parameters.

	MLP	**Neural Snowflake (+ MLP)**	**Neural Snowflake**
No. Parameters	5422	4169	847
Embedding space, $\mathbb{R}^n$	2	2	100
Metric		Mean Square Error	
$\|\mathbf{x}-\mathbf{y}\|^{0.5}\log(1+\|\mathbf{x}-\mathbf{y}\|)^{0.5}$	1.3299	0.0037	**0.0029**
$\|\mathbf{x}-\mathbf{y}\|^{0.1}\log(1+\|\mathbf{x}-\mathbf{y}\|)^{0.9}$	1.2242	0.0032	**0.0031**
$1-\frac{1}{(1+\|\mathbf{x}-\mathbf{y}\|^{0.5})}$	0.0777	**0.00004**	**0.00004**
$1-\exp\frac{-(\|\mathbf{x}-\mathbf{y}\|-1)}{\log(\|\mathbf{x}-\mathbf{y}\|)}$	0.1649	0.00009	**0.00008**
$1-\frac{1}{(1+\|\mathbf{x}-\mathbf{y}\|)^{0.2}}$	0.0314	**0.00005**	**0.00005**
$1-\frac{1}{1+\|\mathbf{x}-\mathbf{y}\|^{0.2}+\|\mathbf{x}-\mathbf{y}\|^{0.5}}$	0.2420	**0.00002**	**0.00002**

Snowflake activation. Before working with neural snowflakes, we evaluate the performance of snowflake activations. This is a straightforward way of augmenting existing latent graph inference algorithms with additional representation power. In appendix H, we have already covered the details regarding the activation function's computational implementation. We intend to adhere to the specifications outlined in that section. We start by running some preliminary studies in which we compare the performance of a GCN equipped with a dDGM module using different latent embedding metric spaces. To ensure a fair comparison in terms of latent space geometry, we have set the dimensionality of the latent embedding space to 4. By keeping the dimensionality fixed, we ensure that we only modify the manifold used for embedding the representations, without altering the dimensionality of the embedding space itself. We start by evaluating the different algorithms on benchmark homophilic graph datasets such as Cora, CiteSeer, CS, and Physics. We start by comparing our model to previous metric spaces used in the literature such as Euclidean space (Kazi et al., 2022). In Table 7 we can observe that the snowflake activation helps achieve higher accuracies, leading to improvements between $1\% - 5\%$.

Table 7: Latent graph inference results across a variety of benchmark homophilic graph datasets. For all experiment the latent space dimensionality used to infer latent graphs is fixed to 4, and the Gumbel top-k trick is used with $k = 7$.

			Cora	**CiteSeer**	**CS**	**Physics**
Model	Metric Space	Input Graph	Accuracy (%) $\pm$ Standard Deviation			
DGM	Euclidean	Yes	82.40$\pm$3.22	73.40$\pm$1.64	85.45$\pm$2.23	95.91$\pm$0.51
DGM	Snowflake	Yes	86.11$\pm$3.72	74.40$\pm$2.08	89.54$\pm$2.48	96.08$\pm$0.46
DGM	Euclidean	No	62.03$\pm$6.20	65.15$\pm$4.84	84.37$\pm$1.20	95.11$\pm$0.33
DGM	Snowflake	No	67.33$\pm$4.10	64.63$\pm$9.24	87.63$\pm$5.25	95.32$\pm$0.45
MLP	Euclidean	No	58.92$\pm$3.28	59.48$\pm$2.14	87.80$\pm$1.54	94.91$\pm$0.30

Next, we increase the dimensionality of the latent space from 4 to 8 and evaluate whether this trend still persists. In Table 8, we can observe that as we increase the dimensionality of the latent space used for inferring the latent graph the performance increases when using both Euclidean or snowflake quasi-metric spaces. We can also see that the difference between the two becomes less significant, except for the CS dataset in which the snowflake activation still performs substantially better.

Table 8: Latent graph inference results with latent space dimensionality fixed to 8, and the Gumbel top-k trick is used with $k = 7$.

			Cora	**CiteSeer**	**CS**	**Physics**
Model	Metric Space	Input Graph	Accuracy (%) $\pm$ Standard Deviation			
DGM	Euclidean	Yes	85.77$\pm$3.64	73.67$\pm$2.30	90.50$\pm$1.89	96.08$\pm$0.41
DGM	Snowflake	Yes	85.41$\pm$3.70	74.19$\pm$2.08	92.98$\pm$0.66	96.15$\pm$0.54
DGM	Euclidean	No	68.37$\pm$5.39	68.10$\pm$2.80	88.17$\pm$2.64	95.27$\pm$0.41
DGM	Snowflake	No	69.51$\pm$4.42	66.86$\pm$2.82	88.67$\pm$3.21	95.36$\pm$0.23
MLP	Euclidean	No	58.92$\pm$3.28	59.48$\pm$2.14	87.80$\pm$1.54	94.91$\pm$0.30

In the specific scenario presented in Table 9, we observe a contrasting effect. As the latent space dimension is reduced to only 2, the performance of both models deteriorates. Nonetheless, it is notable that the use of the snowflake metric space demonstrates greater resilience compared to its Euclidean counterpart. In fact, we can observe performance discrepancies of up to 20% between the two. This is in line with previous synthetic experiments, and demonstrates that snowflake quasi-metric spaces are more efficient at compressing the same information in low-dimensional spaces.

Table 9: Latent graph inference results with latent space dimensionality fixed to 2, and the Gumbel top-k trick is used with $k = 7$.

			Cora	**CiteSeer**	**CS**	**Physics**
Model	Metric Space	Input Graph	Accuracy (%) $\pm$ Standard Deviation			
DGM	Euclidean	Yes	59.25$\pm$14.60	70.00$\pm$2.15	62.15$\pm$2.92	92.01$\pm$2.74
DGM	Snowflake	Yes	79.44$\pm$6.50	69.51$\pm$3.95	79.75$\pm$2.57	94.29$\pm$2.91
DGM	Euclidean	No	42.40$\pm$8.20	60.93$\pm$4.07	69.93$\pm$2.17	86.05$\pm$2.77
DGM	Snowflake	No	64.18$\pm$3.46	64.61$\pm$6.14	81.70$\pm$5.35	93.45$\pm$2.93
MLP	Euclidean	No	58.92$\pm$3.28	59.48$\pm$2.14	87.80$\pm$1.54	94.91$\pm$0.30