A Additional Background

A.1 Differential Geometry, Riemmanian Manifolds, and Product Manifolds

Differential geometry is a mathematical discipline that employs rigorous mathematical techniques, including calculus and other mathematical tools, to investigate the properties and structure of smooth manifolds. These manifolds, despite bearing a local resemblance to Euclidean space, may exhibit global topological features that distinguish them from Euclidean space. The fundamental objective of differential geometry is to elucidate the geometric properties of these spaces, such as curvature, volume, and length, through rigorous mathematical analysis and inquiry.

A Riemannian manifold generalizes Euclidean space to more curved spaces, and it plays a critical role in differential geometry and general relativity. A Riemannian manifold is a smooth manifold M equipped with a Riemannian metric g, which assigns to each point p in M an inner product on the tangent space T_pM. This inner product is a smoothly varying positive definite bilinear form on the tangent space given by $g_p : T_pM \times T_pM \rightarrow \mathbb{R}$, which satisfies the following properties: symmetry, $g_p(X, Y) = g_p(Y, X)$ for all $X, Y \in T_pM$; linearity, $g_p(aX + bY, Z) = ag_p(X, Z) + bg_p(Y, Z)$ for all $X, Y, Z \in T_pM$ and scalars $a, b \in \mathbb{R}$; positive definiteness: $g_p(X, X) > 0$ for all non-zero $X \in T_pM$. The metric g_p induces a norm $||.||_p$ on T_pM given by $||X||_p = \sqrt{g_p(X, X)}$. This norm allows us to define the length of curves on the manifold, and hence a notion of distance between points. Specifically, given a curve $c : [0, 1] \rightarrow M$ and a partition $0 = t_0 < t_1 < ... < t_n = 1$, the length of the curve c over the partition is given by: $L(c) = \sum ||c'(t_i)||c(t_i)$, where $c'(t_i)$ denotes the tangent vector to c at time t_i, and $|| \cdot ||c(t_i)$ denotes the norm induced by the metric at the point $c(t_i)$. The total length of the curve is obtained by taking the limit of the sum as the partition becomes finer.

On the other hand, a product manifold results from taking the Cartesian product of two or more manifolds, resulting in a manifold with a natural product structure. This enables researchers to examine the local and global geometry of the product manifold by studying the geometry of its individual factors. Riemannian manifolds and product manifolds are essential mathematical concepts that have far-reaching applications in diverse fields such as physics, engineering, and computer science. In the appendix of this paper, we provide a more in-depth look at these concepts, including their properties and significance.

A.2 Constant Curvature Model Spaces

A manifold $\mathcal{M}$ is a topological space that can be locally identified with Euclidean space using smooth maps. In Riemannian geometry, a Riemannian manifold or Riemannian space $(\mathcal{M}, g)$ is a real and differentiable manifold $\mathcal{M}$ where each tangent space has an associated inner product g known as the Riemannian metric. This metric varies smoothly from point to point on the manifold and allows for the definition of geometric properties such as lengths of curves, curvature, and angles. The Riemannian manifold $\mathcal{M} \subseteq \mathbb{R}^N$ is a collection of real vectors that is locally similar to a linear space and exists within the larger ambient space $\mathbb{R}^N$.

Curvature is effectively a measure of geodesic dispersion. When there is no curvature geodesics stay parallel, with negative curvature they diverge, and with positive curvature they converge. Constant curvature model spaces are Riemannian manifolds with a constant sectional curvature, which means that the curvature is constant in all directions in a 2D plane (this can be generalized to higher dimensions). These spaces include the Euclidean space, the hyperbolic space, and the sphere. The Euclidean space has zero curvature, while the hyperbolic space has negative curvature and the sphere has positive curvature. Constant curvature model spaces are often used as reference spaces for comparison in geometric analysis and machine learning on non-Euclidean data.

Euclidean space,

$$\mathbb{E}_{K_{\mathbb{E}}}^{d_{\mathbb{E}}} = \mathbb{R}^{d_{\mathbb{E}}} \tag{1}$$

is a flat space with curvature $K_{\mathbb{E}} = 0$. We note that in this context, the notation $d_{\mathbb{E}}$ is used to represent the dimensionality. In contrast, hyperbolic and spherical spaces possess negative and positive curvature, respectively. We define hyperboloids $\mathbb{H}_{K_{\mathbb{H}}}^{d_{\mathbb{H}}}$ as

$$\mathbb{H}^{d_{\mathbb{H}}}_{K_{\mathbb{H}}} = \{\mathbf{x}_p \in \mathbb{R}^{d_{\mathbb{H}}+1} : \langle \mathbf{x}_p, \mathbf{x}_p \rangle_{\mathcal{L}} = 1/K_{\mathbb{H}}\}, \tag{2}$$

where $K_{\mathbb{H}} < 0$ and $\langle \cdot, \cdot \rangle_{\mathcal{L}}$ is the Lorentz inner product

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_1 y_1 + \sum_{j=2}^{d_{\mathbb{H}}+1} x_j y_j, \;\; \forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^{d_{\mathbb{H}}+1}, \tag{3}$$

and hyperspheres $\mathbb{S}^{d_{\mathbb{S}}}_{K_{\mathbb{S}}}$ as

$$\mathbb{S}^{d_{\mathbb{S}}}_{K_{\mathbb{S}}} = \{\mathbf{x}_p \in \mathbb{R}^{d_{\mathbb{S}}+1} : \langle \mathbf{x}_p, \mathbf{x}_p \rangle_2 = 1/K_{\mathbb{S}}\}, \tag{4}$$

where $K_{\mathbb{S}} > 0$ and $\langle \cdot, \cdot \rangle_2$ is the standard Euclidean inner product

$$\langle \mathbf{x}, \mathbf{y} \rangle_2 = \sum_{j=1}^{d_{\mathbb{S}}+1} x_j y_j, \;\; \forall\, \mathbf{x}, \mathbf{y} \in \mathbb{R}^{d_{\mathbb{S}}+1}, \tag{5}$$

Table 1 summarizes the key operators in Euclidean, hyperbolic, and spherical spaces with arbitrary curvatures. It is worth noting that the three model spaces discussed above can cover any curvature value within the range of $(-\infty, \infty)$. However, there is a potential issue with the hyperboloid and hypersphere becoming divergent as their respective curvatures approach zero. This results in the manifolds becoming flat, and their distance and metric tensors do not become Euclidean at zero-curvature points, which could hinder curvature learning. Therefore, stereographic projections are considered to be suitable alternatives to these spaces as they maintain a non-Euclidean structure and inherit many of the properties of the hyperboloid and hypersphere.

Table 1: Relevant operators (exponential maps and distances between two points) in Euclidean, hyperbolic, and spherical spaces with arbitrary constant curvatures.

Space Model	$exp_{\mathbf{x}_p}(\mathbf{x})$	$\mathfrak{d}(\mathbf{x}, \mathbf{y})$
$\mathbb{E}$, Euclidean	$\mathbf{x}_p + \mathbf{x}$	$\|\|\mathbf{x} - \mathbf{y}\|\|_2$
$\mathbb{H}$, hyperboloid	$\cosh\left(\sqrt{-K_{\mathbb{H}}}\|\|\mathbf{x}\|\|\right)\mathbf{x}_p + \sinh\left(\sqrt{-K_{\mathbb{H}}}\|\|\mathbf{x}\|\|\right)\frac{\mathbf{x}}{\sqrt{-K_{\mathbb{H}}}\|\|\mathbf{x}\|\|}$	$\frac{1}{\sqrt{-K_{\mathbb{H}}}} \operatorname{arccosh}\left(K_{\mathbb{H}}\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}\right)$
$\mathbb{S}$, hypersphere	$\cos\left(\sqrt{K_{\mathbb{S}}}\|\|\mathbf{x}\|\|\right)\mathbf{x}_p + \sin\left(\sqrt{K_{\mathbb{S}}}\|\|\mathbf{x}\|\|\right)\frac{\mathbf{x}}{\sqrt{K_{\mathbb{S}}}\|\|\mathbf{x}\|\|}$	$\frac{1}{\sqrt{K_{\mathbb{S}}}} \arccos\left(K_{\mathbb{S}}\langle \mathbf{x}, \mathbf{y} \rangle_2\right)$

A.3 Constructing Product Manifolds

In this appendix, we provide additional details about the generation of product manifolds. In mathematics, a product manifold is a type of manifold obtained by taking the Cartesian product of two or more manifolds. Informally, a product manifold can be thought of as a space formed by taking multiple smaller spaces and merging them in a way that maintains their individual structures. For instance, the surface of a sphere can be seen as a product manifold consisting of two real lines that intersect at each point on the sphere. Product manifolds play a crucial role in various fields of mathematics, including physics, topology, and geometry.

A product manifold can be defined as the Cartesian product:

$$\mathcal{P} = \bigtimes_{i=1}^{n_{\mathcal{P}}} \mathcal{M}^{d_i}_{Ki} \tag{6}$$

Here, K_i and d_i represent the curvature and dimensionality of the space, respectively. Points $\mathbf{x}_p \in \mathcal{P}$ are expressed using their coordinates:

$$\mathbf{x}_p = concat\left(\mathbf{x}_p^{(1)}, \mathbf{x}_p^{(2)}, ..., \mathbf{x}_p^{(n_{\mathcal{P}})}\right) : \mathbf{x}_p^{(i)} \in \mathcal{M}_{K_i}^{d_i}. \tag{7}$$

Also, the metric of the product manifold decomposes into the sum of the constituent metrics

$$g_{\mathcal{P}} = \sum_{i=1}^{n_{\mathcal{P}}} g_i, \tag{8}$$

hence, $(\mathcal{P}, g_{\mathcal{P}})$ is also a Riemannian manifold. It should be noted that the signature of the product space is parametrized with several degrees of freedom. These include the number of components used in the product space, the type of model spaces employed, their dimensionality, and curvature. If we restrict $\mathcal{P}$ to be composed of the Euclidean plane $\mathbb{E}_{K_{\mathbb{E}}}^{d_{\mathbb{E}}}$, hyperboloids $\mathbb{H}_{K_j^{\mathbb{H}}}^{d_j^{\mathbb{H}}}$, and hyperspheres $\mathbb{S}_{K_k^{\mathbb{S}}}^{d_k^{\mathbb{S}}}$ of constant curvature, we can rewrite Equation 6 as

$$\mathcal{P} = \mathbb{E}_{K_{\mathbb{E}}}^{d_{\mathbb{E}}} \times \left(\bigtimes_{j=1}^{n_{\mathbb{H}}} \mathbb{H}_{K_j^{\mathbb{H}}}^{d_j^{\mathbb{H}}} \right) \times \left(\bigtimes_{k=1}^{n_{\mathbb{S}}} \mathbb{S}_{K_k^{\mathbb{S}}}^{d_k^{\mathbb{S}}} \right), \tag{9}$$

where $K_{\mathbb{E}} = 0$, $K_j^{\mathbb{H}} < 0$, and $K_k^{\mathbb{S}} > 0$. Hence, $\mathcal{P}$ would have a total of $1 + n_{\mathbb{H}} + n_{\mathbb{S}}$ component spaces, and total dimension $d_{\mathbb{E}} + \Sigma_{j=1}^{n_{\mathbb{H}}} d_j^{\mathbb{H}} + \Sigma_{k=1}^{n_{\mathbb{S}}} d_k^{\mathbb{S}}$. Distances between two points in the product manifold can be computed aggregating the distance contributions $\mathfrak{d}_i$ from each manifold composing the product manifold:

$$\mathfrak{d}_{\mathcal{P}}(\overline{\mathbf{x}}_{p_1}, \overline{\mathbf{x}}_{p_2}) = \sqrt{\sum_{i=1}^{n_{\mathcal{P}}} \mathfrak{d}_i \left(\overline{\mathbf{x}}_{p_1}^{(i)}, \overline{\mathbf{x}}_{p_2}^{(i)}\right)^2}. \tag{10}$$

The overline denotes that the points $\mathbf{x}_{p_1}$ and $\mathbf{x}_{p_2}$ have been adequately projected on to the product manifold using the exponential map before computing the distances.

A.4 The Rationale Behind using Product Manifolds of Model Spaces

Most arbitrary Riemannian manifolds do not have closed-form solutions for their exponential maps and geodesic distances between points, as well as for other relevant mathematical notions. The exponential map takes a point on the manifold and exponentiates a tangent vector at that point to produce another point on the manifold. The geodesic distance between two points on a manifold is the shortest path along the manifold connecting those points.

For simple, well-studied manifolds like Euclidean space, hyperbolic space, and spherical space, closed-form solutions for exponential maps and geodesic distances are available. This is because these manifolds have constant curvature, which allows for more straightforward calculations. However, for most other manifolds, the exponential map and geodesic distances must typically be computed numerically or approximated using specialized algorithms. These computations often involve solving differential equations, and depending on the manifold's curvature and geometry, these solutions can be quite complex and may not have closed-form expressions. For certain types of manifolds, like product manifolds where each factor is a well-understood manifold, the computation of exponential maps and geodesic distances can sometimes be simplified due to the separability of the metric. However, scaling these methods to more general, complex manifolds can be challenging and computationally intensive.

As mentioned, closed-form solutions for exponential maps and geodesic distances are not common for arbitrary Riemannian manifolds, and computational methods often rely on numerical techniques and algorithms due to the complexity of these calculations. These properties have made product manifolds the most attractive tool to endow the latent space with richer structure while retaining computational traceability, see (**???????**). This motivates us to use product manifolds in our setup, which keeps our work in line with the state-of-the-art approaches in the literature. We highlight that one of the central contributions of our paper is to shed light on the problem of NLGS, and provide an initial solution to the problem in the context of the current state-of-the-art in geometric latent space modelling. Future research avenues could explore generalizations of this which account for new techniques that emerge in the literature, potentially comparing a more general set of latent manifolds.

A.5 Geometry of Negative Curvature: Further Discussion

Lastly, in this appendix, we aim to provide an intuition regarding closed balls in spaces of curvature smaller than zero. The hyperbolic spaces we consider in this article are hyperbolic manifolds (that is, hyperbolic in the differentiable sense). However, there are other generalisations of the notion of hyperbolicity that carry over to the setting of geodesic metric spaces (such as Gromov hyperbolicity). This theory unifies the two extremes of spaces of negative curvature, the real hyperbolic spaces $\mathbb{H}^n$ and trees. Recall that a tree is a simple graph that does not have closed paths. Trees provide a great intuition for the metric phenomena that we should expect in spaces of negative curvature. For example, let us denote by G an infinite planar grid (observe that this graph is far from being a tree), and let us denote by T a regular tree of valency 4 (this means that every vertex in T is connected to exactly other four vertices). The 2-dimensional Euclidean space $\mathbb{E}^2$ is quasi-isometric to G and the 2-dimensional real hyperbolic space $\mathbb{H}^2$ is quasi-isometric (but not isometric) to T. This observation is due to Švarc and Milnor. Recall that the graphs G and T are endowed with the natural path metric. A ball of radius $n > 0$ in G has between n^2 and $4n^2$ points (polynomial in n), while a ball of radius n in T has exactly $(4^n - 1)/3$ points (exponential in n). Now let us come back to $\mathbb{H}^3$, where the "continuous" analogue of this principle is also true; that is, that there exists a constant $c > 1$ such that for every $n > 0$, one needs at least c^n balls of radius one to cover a ball of radius n in $\mathbb{H}^3$. If we only want to understand the local geometry of $\mathbb{H}^3$, we can re-scale this principle to say the following. Let $B_1^{\mathbb{H}^3}$ denote the ball of radius one in $\mathbb{H}^3$. For all positive integers $n > 0$, one needs at least c^n balls of radius $1/n$ to cover $B_1^{\mathbb{H}_3}$. This tree-like behaviour should be compared with the opposite principle that holds in $\mathbb{E}^3$, being that for all $n > 0$, one needs at most $8n^3$ (a polynomial bound) of balls of radius one to cover $B_1^{\mathbb{E}_3}$. Essentially, we should think of $B_1^{\mathbb{H}_3}$ as a manifold that is isomorphic to $B_1^{\mathbb{E}^3}$ which is difficult to cover by balls of smaller radius, in fact, it will be coarsely approximated by a finite tree of balls of small radius (the underlying tree being bigger as the chosen radius is smaller). Furthermore, the philosophy that travelling around $B_1^{\mathbb{H}_3}$ becomes more and more expensive as we leave the origin is made accurate with the consideration of divergence functions: geodesics in $\mathbb{H}^3$ witness exponential divergence. More precisely, there exists a constant $c > 1$ such that for every $r > 0$ and for every geodesic $\gamma : \mathbb{R} \longrightarrow \mathbb{H}^3$ passing through the origin at time $t = 0$ it is true that if α is a path that connects $\gamma(-r)$ and $\gamma(r)$ and that lies entirely outside of the ball of radius $d(0, \gamma(r))$, then α must have length at least c^r. Again, this is better appreciated when put in comparison with the Euclidean space $\mathbb{E}^3$. Any geodesic (any line) $\gamma : \mathbb{R} \longrightarrow \mathbb{E}^3$ passing through the origin at time $t = 0$ will have the property that for all $r > 0$ there is a path of length $\pi r + \varepsilon$, with $\varepsilon > 0$ arbitrarily small, that lies entirely outside of the ball of radius r with centre at the origin. The length $\pi r + \varepsilon$ is linear in r, as opposed to the exponential length that is required in hyperbolic spaces.

B Further Details on the Gromov-Hausdorff Algorithm for Neural Latent Geometry Search

As discussed in Section **??**, there are limitations to the Hausdorff distance as a metric for comparing point sets that represent discretized versions of continuous manifolds. The original algorithm by **?** for computing the Hausdorff distance assumes that the point sets reside in the same space with the same dimensionality, which is a limiting requirement. However, the Gromov-Hausdorff distance proposed in this work allows us to compare metric spaces that are not embedded in a common ambient space, but it is not exactly computable. We will focus on providing an upper bound for $\mathrm{d}_{\mathrm{GH}}(\mathbb{E}^n, \mathbb{S}^n)$ and describing an algorithm for obtaining upper bounds for $\mathrm{d}_{\mathrm{GH}}(\mathbb{E}^n, \mathbb{H}^n)$ and $\mathrm{d}_{\mathrm{GH}}(\mathbb{S}^n, \mathbb{H}^n)$ too. As previously mentioned, $\mathbb{E}^n$ and $\mathbb{S}^n$ can be isometrically embedded into $\mathbb{E}^{6n-6}$ in many ways, which provides a common underlying space for computing Hausdorff distances between $\mathbb{E}^n$, $\mathbb{S}^n$, and $\mathbb{H}^n$. The embeddings of different spaces require making choices: in particular, there is no way to exactly compute the infimum in the definition of Gromov-Hausdorff distance given in Section **??**:

$$\mathrm{d}_{\mathrm{GH}}(A, B) = \inf_{(X,f,g)\in \mathrm{ES}(A,B)} \mathrm{d}_{\mathrm{H}}^{X,f,g}(f(A), g(B)). \tag{11}$$

so we resort to numerical approximations which are later described in this appendix.

To estimate the mutual Gromov-Hausdorff distance between the infinite smooth spaces $\mathbb{E}^2$ and $\mathbb{H}^2$, the first step is to approximate them using finite discrete versions. This is done by considering well-distributed collections of points in $\mathbb{E}^2$ and $\mathbb{H}^2$, obtained by applying the exponential map to a collection of points in $\mathbb{R}^2$. Multiple isometric embeddings of $\mathbb{E}^2$ into $\mathbb{R}^6$ are also considered. The estimation is obtained by computing the minimum of the Hausdorff distance between the point sets obtained from the collections in $\mathbb{R}^6$ and the isometric embedding of $\mathbb{H}^2$ into $\mathbb{R}^6$. The same applies for spherical space.

B.1 Generating Points in the Corresponding Balls of Radius One

All our computations will be done in dimension two, so we will simply precise how to generate points in $B_{\mathbb{E}^2}$, $B_{\mathbb{S}^2}$ and $B_{\mathbb{H}^2}$. For $B_{\mathbb{E}^2}$ and $B_{\mathbb{S}^2}$, we will use very elementary trigonometry. For $B_{\mathbb{H}^2}$, we will need an explicit description of the exponential map of $\mathbb{H}^2$. Using the descriptions

$$B_{\mathbb{E}^2} = \{(r\cos(t), r\sin(t)) : r \in [0,1], t \in [0,2\pi)\}, \tag{12}$$

and

$$B_{\mathbb{S}^2} = \{(\sin(\beta)\cos(\alpha), \sin(\beta)\sin(\alpha), \cos(\beta)) : \alpha \in [0,2\pi), \beta \in [0,1]\}, \tag{13}$$

it will be easy to generate collections of points in $B_{\mathbb{E}^2}$ and $B_{\mathbb{S}^2}$. As we anticipated above in the outline of our strategy to estimate $\mathrm{d}_{\mathrm{GH}}(B_{\mathbb{E}^2}, B_{\mathbb{H}^2})$, in order to give explicit well-distributed collection of points in $B_{\mathbb{H}^2}$, it is enough to give a well-distributed collection of points in $B_{\mathbb{E}^2}$ and consider its image under the exponential map $\exp_0 : B_{\mathbb{E}^2} \to B_{\mathbb{H}^2}$, where we have identified $B_{\mathbb{E}^2}$ with the ball of radius one of $\mathbb{R}^2 \cong T_0\mathbb{H}^2$, i.e. the tangent space of $\mathbb{H}^2$ at the point 0. However, we should remark that our description of $\mathbb{H}^n$ will not be any of the three most standard ones: namely the *hyperboloid model*, the *the Poincaré ball model*, or the *upper half-plane model*. We describe $\mathbb{H}^n$ as the differentiable manifold $\mathbb{R}^n$ (of Cartesian coordinates $x, y_1, \ldots, y_{n-1}$) equipped with the Riemannian metric $g_{-1} = \mathrm{d}x^2 + e^{2x}(\mathrm{d}y_1^2 + \cdots \mathrm{d}y_{n-1}^2)$. This metric is complete and has constant sectional curvature of -1, which implies that $(\mathbb{R}^n, g_{-1})$ is isometric to $\mathbb{H}^n$. In the hyperboloid model of $\mathbb{H}^2$, we view this space as a submanifold of $\mathbb{R}^3$ (although with a distorted metric, not the one induced from the ambient $\mathbb{R}^3$). In this model of $\mathbb{H}^2$, one can explicitly describe by the following assignment $\exp_0 : \mathbb{R}^2 \to \mathbb{H}^2$ by

$$(x,y) \mapsto \left(\frac{x\sinh(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}, \frac{y\sinh(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}, \cosh(\sqrt{x^2+y^2}) \right). \tag{14}$$

We use this explicit formula of the exponential map with coordinates in the hyperboloid model of $\mathbb{H}^2$, to give a explicit formula for the exponential map with coordinates in the model of $\mathbb{H}^2$ that we introduced above, denoted by $(\mathbb{R}^2, g_{-1})$. In order to change coordinates from the hyperboloid model to the Poincaré ball model, we use the following isometry (which is well-known and can be thought of as a hyperbolic version of the stereographic projection from $\mathbb{S}^2$ to $\mathbb{R}^2$):

$$(x,y,z) \mapsto \left(\frac{x}{1+z}, \frac{y}{1+z} \right). \tag{15}$$

To change coordinates from the Poincaré ball model to the upper half-plane model, one can use the following isometry:

$$(x,y) \mapsto \left(\frac{-2y}{(x-1)^2+y^2}, \frac{1-x^2-y^2}{(x-1)^2+y^2} \right). \tag{16}$$

The previous assignment comes from the standard Möbius transformation $z \mapsto \frac{z+i}{iz+1}$ that identifies the Euclidean ball of radius one of $\mathbb{C}$ and the upper half plane of $\mathbb{C}$ as Riemann surfaces (which we give explicitly in this case, although it is known to exist and to be unique by the classical Riemann mapping theorem). Finally, to go from coordinates in the upper half-plane model of $\mathbb{H}^2$ to our model $(\mathbb{R}^2, g_{-1})$, we use the isometry

$$(x,y) \to (-\log y, x). \tag{17}$$

B.2 Embedding of $\mathbb{H}^n$ into $\mathbb{E}^{6n-6}$

We want to define an isometric embedding of $B^1_{\mathbb{H}^n}$ into $\mathbb{E}^{6n-6}$ (F from Section **??**). This higher dimensional space is our candidate to fit in the three geometries $\mathbb{E}^n$, $\mathbb{H}^n$ and $\mathbb{S}^n$ to compute their Hausdorff dimensions as subspaces of $\mathbb{E}^{6n-6}$ and hence estimate their Gromov-Hausdorff distances. Before describing such embedding, we introduce several preliminary auxiliary functions.

Let $\chi(t) = \sin(\pi t) \cdot e^{-\sin^{-2}(\pi t)}$ for non-integer values of t. A priori, the inverse of $\sin(0) = 0$ does not make sense but since $\lim_{t\to 0^+} \chi(t) = \lim_{t\to 0^-} \chi(t) = 0$, we can set $\chi(0) = 0$ so it is still continuous. In fact, it is smooth and, in particular, integrable. We can say the same at all points when t is an integer, so we set $\chi(t) = 0$ for all integers t and we obtain an smooth function χ defined on $\mathbb{R}$.

$$A = \int_0^1 \chi(t)dt. \tag{18}$$

We also define

$$\psi_1(x) = \sqrt{\frac{1}{A} \cdot \int_0^{1+x} \chi(t)\,\mathrm{d}t}, \tag{19}$$

and

$$\psi_2(x) = \sqrt{\frac{1}{A} \cdot \int_0^{x} \chi(t)\,\mathrm{d}t}. \tag{20}$$

We set c to be the constant

$$c = 2\max\left\{G_1, G_2\right\}, \tag{21}$$

defined in terms of the following

$$G_1 = \left\| \frac{\mathrm{d}}{\mathrm{d}x}\left(\sinh(x) \cdot \psi_1(x)\right)\right\|_{L^\infty[-2,2]}, \tag{22}$$

$$G_2 = \left\| \frac{\mathrm{d}}{\mathrm{d}x}\left(\sinh(x) \cdot \psi_2(x)\right)\right\|_{L^\infty[-2,2]}, \tag{23}$$

$$h(x,y) = \frac{\sinh(x)}{c}\Big(\psi_1(x)\cos(c\cdot y), \psi_1(x)\sin(c\cdot y), \psi_2(x)\cos(c\cdot y), \psi_2(x)\sin(c\cdot y)\Big), \tag{24}$$

$$\psi(x,y) = \Big(\sinh^{-1}(ye^x), \log(\sqrt{e^{-2x}+y^2})\Big). \tag{25}$$

We also define $f_0(x,y) = \Big(\int_0^{\sinh^{-1}(ye^x)} \sqrt{1-\varepsilon(t)^2}\mathrm{d}t, \log(\sqrt{e^{-2x}+y^2}), h(\psi(x,y))\Big)$,

with ε being

$$\varepsilon = \frac{G_1^2 + G_2^2}{c^2}. \tag{26}$$

This way, we can set

$$f(x, y_1, \ldots, y_{n-1}) = \frac{1}{\sqrt{n-1}}\big(f_0(x, \sqrt{n-1}y_1), \ldots, f_0(x, \sqrt{n-1}y_{n-1}\big).$$

Recall that in our case, we use the function $F(\cdot) = f(n=2, \cdot)$ to map the set of points Q in $\mathbb{H}^2$ to $\mathbb{R}^6$, and for computing

$$\mathrm{d}_{\mathrm{GH}}(B_{\mathbb{E}^2}, B_{\mathbb{H}^2}) \approx \min_{i,j,k} \mathrm{d}_{\mathrm{H}}^{\mathbb{R}^6, f_k, F}(P_i, Q_j) = \min_{i,j,k} \mathrm{d}_{\mathrm{H}}^{\mathbb{R}^6}(f_k(P_i), F(Q_j)). \tag{27}$$

B.3 Estimation of the Gromov-Hausdorff distance between the Euclidean and Spherical Spaces

In this section we give the analytical derivation of the Gromov-Hausdorff distance between the Euclidean space $\mathbb{E}^n$ and the Spherical space $\mathbb{S}^n$. We first explain the case $n = 1$, where we can visually understand the situation in a better way because all distances will be measured in $\mathbb{E}^2$. Afterwards, we replicate the same argument for arbitrary n. Recall that the Spherical model $\mathbb{S}^n$ can be described as the metric subspace of $\mathbb{E}^{n+1}$ corresponding to the Euclidean sphere of radius one. Hence, as a subspace of $\mathbb{R}^{n+1}$, it corresponds to $\{(x_0, \ldots, x_n) : \sum_{i=0}^n x_i^2 = 1\}$.

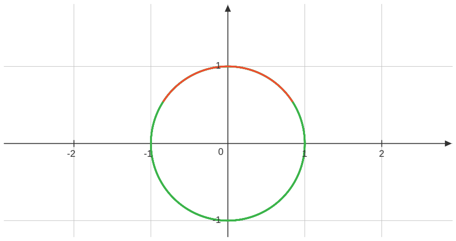

Figure 1: The one-dimensional model of spherical geometry $\mathbb{S}^1$ isometrically embedded in $\mathbb{R}^2$.

The ball of radius one of $\mathbb{S}^1$, denoted by $B_{\mathbb{S}^1}$, is highlighted in red. Our estimation of the Gromov-Hausdorff distance between $\mathbb{E}^1$ and $\mathbb{S}^1$ is motivated by the following observation. In the y-axis, the red arc ranges from $\frac{1+\cos(1)}{2}$ to 1. If we consider the following blue segment, representing the ball of radius one of $\mathbb{E}^1$ (denoted by $B_{\mathbb{E}^1}$), we get

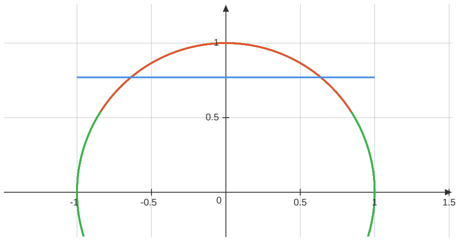

Figure 2: Comparison between $B_{\mathbb{E}^1}$ (in red) and $B_{\mathbb{S}^1}$ (in blue) inside $\mathbb{E}^2$.

$$\sup_{x \in B_{\mathbb{S}^1}} d_{\mathbb{E}^2}(x, B_{\mathbb{E}^1}) = \frac{1-\cos(1)}{2} \approx 0.23. \tag{28}$$

However, it can be seen that $\sup_{x \in B_{\mathbb{E}^1}} d_{\mathbb{E}^2}(x, B_{\mathbb{S}^1}) = 0.279$. More generally, if the blue line is chosen to be at the height $1 - x$, for some x lying in the closed interval $[0, 1-\cos(1)]$, it is not hard to see that, for such embedding $f : B_{\mathbb{E}^1} \to \mathbb{E}^2$,

$$dh^{\mathbb{E}^2}(f(B_{\mathbb{E}^1}), B_{\mathbb{S}^1}) = \max\left\{x, 1-x, \sqrt{(1-\sin(1))^2 + (1-\cos(1)-x)^2}\right\}, \tag{29}$$

whose minimum is attained exactly when $x = \sqrt{(1-\sin(1))^2 + (1-x-\cos(1))^2}$, i.e. when

$$x = \frac{(1-\sin(1))^2 + (1-\cos(1))^2}{2-2\cos(1)} \approx 0.257. \tag{30}$$

This can be seen in the following picture. The two pink lines represent the biggest lengths between points in $B_{\mathbb{E}^1}$ and $B_{\mathbb{S}^1}$, whose length is approximately equal to 0.257. Since we cannot compute exactly the Gromov-Hausdorff distance $\mathrm{d_{GH}}(\mathbb{S}^1, \mathbb{E}^1)$, choices have to be made. We have discussed why we would expect it to be somewhere in between 0.23 and 0.257. Since we are considering a very simple embedding for these estimations, it is reasonable to expect $\mathrm{d_{GH}}(\mathbb{S}^1, \mathbb{E}^1)$ to get closer

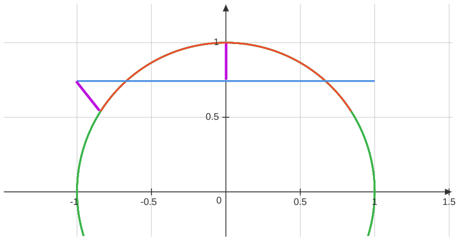

Figure 3: Comparison between $B_{\mathbb{E}^1}$ (in red) and $B_{\mathbb{S}^1}$ (in blue) inside $\mathbb{E}^2$ and a schematic of the biggest lengths (in pink) between the two point sets.

to the lowest number when considering embeddings of these spaces in more complicated higher dimensional spaces, as the definition of the Gromov-Hausdorff distance allows.

For an arbitrary n, we reduce the estimation of $\mathrm{d}_{\mathrm{GH}}(B_{\mathbb{E}^n}, B_{\mathbb{S}^n})$ to the one-dimensional case as follows. Analogously as we did for $n = 1$, given any value of x, we can consider $B_{\mathbb{E}^n}$ isometrically embedded in $\mathbb{E}^{n+1}$ by the map $f_x : B_{\mathbb{E}^n} \to \mathbb{R}^{n+1}$ defined by $f((x_0, x_1, \ldots, x_n)) = (x_0, x_1, \ldots, x_n, x)$. We view $B_{\mathbb{S}^n}$ isometrically embedded into $\mathbb{R}^{n+1}$ as the ball of radius one of $\mathbb{S}^n \subset \mathbb{R}^{n+1}$ with centre in the north pole of the sphere, i.e. the point with coordinates $(0, 0, \ldots, 0, 1)$. Given any unit vector u in $\mathbb{R}^{n+1}$ orthogonal to $\vec{n} = (0, 0, \ldots, 0, 1)$, we define π_u to be the two-dimensional plane linearly spanned by u and $\vec{n}$. It is clear that, as u ranges over the orthogonal space of $(0, 0, \ldots, 0, 1)$, the intersections $B_{\mathbb{S}^n} \cap \pi_u$ cover the whole $B_{\mathbb{S}^n}$ and the intersections $f_x(B_{\mathbb{E}^n}) \cap \pi_u$ cover the whole $f_x(B_{\mathbb{E}^n})$. Hence, in order to compute $\mathrm{d}_{\mathrm{H}}^{\mathbb{E}^{n+1}}(f_x(B_{\mathbb{E}^n}), B_{\mathbb{S}^n})$, it suffices to compute $\mathrm{d}_{\mathrm{H}}^{\mathbb{E}^{n+1}}(f_x(B_{\mathbb{E}^n}) \cap \pi_u, B_{\mathbb{S}^n} \cap \pi_u)$ for all unit vectors u orthogonal to $\vec{n}$. Moreover, for two such unit vectors u and v, there is a rigid motion (a rotation) of the whole ambient space $\mathbb{R}^{n+1}$, that fixes $\vec{n}$ and that maps isometrically $f_x(B_{\mathbb{E}^n}) \cap \pi_u$ to $f_x(B_{\mathbb{E}^n}) \cap \pi_v$ and $B_{\mathbb{S}^n} \cap \pi_u$ to $B_{\mathbb{S}^n} \cap \pi_v$. In particular,

$$\mathrm{d}_{\mathrm{H}}^{\mathbb{E}^{n+1}}(f_x(B_{\mathbb{E}^n}) \cap \pi_u, B_{\mathbb{S}^n} \cap \pi_u) = \mathrm{d}_{\mathrm{H}}^{\mathbb{E}^{n+1}}(f_x(B_{\mathbb{E}^n}) \cap \pi_v, B_{\mathbb{S}^n} \cap \pi_v).$$

So we can do the previous computation for the specific value of $u = (0, 0, \ldots, 1, 0)$ (for clarity, where we mean the unit vector where the n-th coordinate is equal to 1 and the rest are zero). Crucially, the projection of $\mathbb{R}^{n+1}$ unto $\mathbb{R}^2$ (by projecting onto the last two coordinates) restrict to isometries on $B_{\mathbb{S}^n} \cap \pi_u$ and $B_{\mathbb{E}^n} \cap \pi_u$, from where it is deduced, analogously as in 29, that, as long as $x \in [0, 1 - \cos(1)]$, we have the following:

$$\mathrm{d}_{\mathrm{H}}^{\mathbb{E}^{n+1}}(f_x(B_{\mathbb{E}^n}) \cap \pi_u, B_{\mathbb{S}^n} \cap \pi_u) = \max\left\{x, 1 - x, \sqrt{(1 - \sin(1))^2 + (1 - \cos(1) - x)^2}\right\}.$$

B.4 Computational Implementation of the Gromov-Hausdorff Distance between the Remaining Model Spaces of Constant Curvature

In this appendix, we provide further details regarding how the Gromov-Hausdorff distances between the remaining manifolds (between $B_{\mathbb{E}^2}$ and $B_{\mathbb{H}^2}$, and $B_{\mathbb{S}^2}$ and $B_{\mathbb{H}^2}$) were approximated computationally. Note that as discussed in Section **??**, these distances will be leveraged to compare not only model spaces but product manifolds as well.

B.4.1 Discretizing the Original Continuous Manifolds

This section discuss how the the points in the corresponding balls of radius one described in Appendix B.1 are generated from a practical perspective. The Euclidean, hyperbolic, and spherical spaces are continuous manifolds. To proximate the Gromov-Hausdorff distance between them we must discretize the spaces. A more fine-grained discretization results in a better approximation. As previously mentioned in Section **??**, to compare $\mathbb{E}^n$, $\mathbb{S}^n$, and $\mathbb{H}^n$, we can take closed balls of radius one in each space. Since these spaces are homogeneous Riemannian manifolds, every ball of radius one is isometric. We can estimate or provide an upper bound for their Gromov-Hausdorff distance by using this method.

In the case of the Euclidean plane, we sample points from the ball, $B_{\mathbb{E}^2} = \{(r\cos(t), r\sin(t)) : r \in [0, 1], t \in [0, 2\pi)\}$, with a discretization of $10,000$ points in both r and t. To obtain points in $B_{\mathbb{H}^2}$ we will use the points in $B_{\mathbb{E}^2}$ as a reference, and apply the exponential map, and the change of coordinates described in Appendix B.1 to convert points in $B_{\mathbb{E}^2}$ to points in $B_{\mathbb{H}^2}$. However, due to numerical instabilities instead of sampling from $B_{\mathbb{E}^2}$, we will restrict ourselves to $B'_{\mathbb{E}^2} = \{(r\cos(t), r\sin(t)) : r \in [0.00000001, 0.97], t \in [0, 2\pi)\}$ and use a discretization of $10,000$ as before. Lastly to sample points for the spherical space, $B_{\mathbb{S}^2} = \{(\sin(\beta)\cos(\alpha), \sin(\beta)\sin(\alpha), \cos(\beta)) : \alpha \in [0, 2\pi), \beta \in [0, 1]\}$, we use a discretization of 100 for both α and β. The granualirity of the discretization was chosen as a trade-off between resolution and computational time. We observed that the Gromov-Hausdorff distance stabilized for our discretization. However, given the nature of the Gromov-Hausdorff distance, it is difficult to conclude whether some unexpected behaviour could be observed with greater discretization.

B.4.2 Calculating Constants for the Embedding Function

The next step is to obtain a computational approximation of the mapping function defined in Appendix B.2. The embedding of $\mathbb{H}^n$ into $\mathbb{E}^{6n-6}$ requires computing several constants. Using a standard integral solvers $A \approx 0.141328$, (Equation 18) can be approximated. To calculate the constant $c \approx 10.255014502464228$, we discretize the input $x \in [-2, 2]$ using a step size of 10^{-8} to compute G_1 and G_2 and use a for loop to calculate the max in Equation 21. Note that this requires to constantly reevaluate the integrals for $\psi_1(x)$ and $\psi_2(x)$. Likewise, G_1 and G_2 are also used to obtain ε in Equation 26, which is in turn used to calculate the mapping function that maps points in $\mathbb{H}^2$ to the higher dimensional embedding Euclidean space $\mathbb{E}^6$. We use F to map $B_{\mathbb{H}^2}$ to $\mathbb{E}^6$, and we keep those points fixed in space.

B.4.3 Optimizing the Embedding Functions for the Euclidean and Spherical Spaces

Next to approximate the Gromov-Hausdorff distances:

$$\mathrm{d_{GH}}(B_{\mathbb{E}^2}, B_{\mathbb{H}^2}) \approx \min_{i,j,k} \mathrm{d}_{\mathrm{H}}^{\mathbb{R}^6, f_k, F}(P_i^H, Q_j) = \min_{i,j,k} \mathrm{d}_{\mathrm{H}}^{\mathbb{R}^6}(f_k(P_i^H), F(Q_j)), \tag{31}$$

and,

$$\mathrm{d_{GH}}(B_{\mathbb{S}^2}, B_{\mathbb{H}^2}) \approx \min_{i,j,k} \mathrm{d}_{\mathrm{H}}^{\mathbb{R}^6, g_k, F}(P_i^S, Q_j) = \min_{i,j,k} \mathrm{d}_{\mathrm{H}}^{\mathbb{R}^6}(g_k(P_i^S), F(Q_j)), \tag{32}$$

we must optimize f_k and g_k. These are the functions used to embed $B_{\mathbb{E}^2}$ and $B_{\mathbb{S}^2}$ in $\mathbb{E}^{6n-6}$. Note that in practice, $B_{\mathbb{E}^2}$ and $B_{\mathbb{S}^2}$ are discretized into P_i^H and P_i^S, respectively.

To optimize for f_k we consider all possible permutations of the basis vectors of $\mathbb{E}^6$: $\{e_1, e_2, e_3, e_4, e_5, e_6\}$ for each f_k we consider two elements $\{e_i, e_j\}$ and use those dimensions to embed $\mathbb{E}^2$ in $\mathbb{E}^6$. In principle, we should also consider a small offset given by $F(\mathbf{0}) = \mathbf{0}$, but it is zero regardless. Additionally, during the optimization we also add a small vector (with all entries but a single dimension between zero) to the mapping function to translate the plane in different directions by an offset of between -0.5 and 0.5, with a total of a 100 steps between these two quantities.

To optimize for g_k we follow a similar procedure in which we consider all permutations of the basis vectors. Note however, that in this case we would have three basis vectors instead of two, given how $B_{\mathbb{S}^2} = \{(\sin(\beta)\cos(\alpha), \sin(\beta)\sin(\alpha), \cos(\beta)) : \alpha \in [0, 2\pi), \beta \in [0, 1]\}$ is sampled. For each permutation family, P_i^S, we also consider its negative counterpart, $-P_i^S$, to compute the Gromov-Hausdorff distance as well as experimenting with offsetting the mapping function.

B.5 Derivation of Number of Product Manifold Combinations in the Search Space

Here, we derive the exact number of nodes in the graph search space in our setting. In particular, we consider the case with three model spaces (the Eucledian plane, the hyperboloid and the hypershere). The growth of the search space can be modelled with growth of a tree, as depicted in Figure 4.

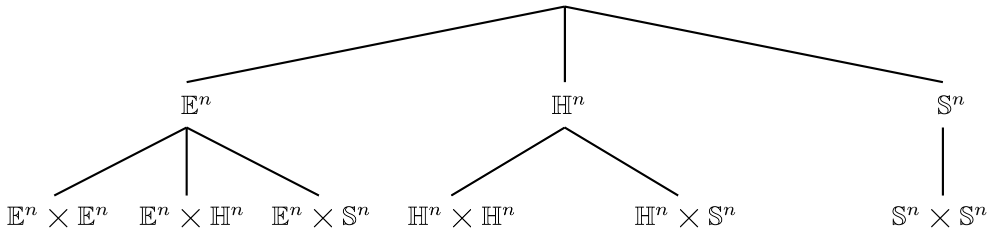
$\mathbb{E}^n$
$\mathbb{H}^n$
$\mathbb{S}^n$
$\mathbb{E}^n \times \mathbb{E}^n$
$\mathbb{E}^n \times \mathbb{H}^n$
$\mathbb{E}^n \times \mathbb{S}^n$
$\mathbb{H}^n \times \mathbb{H}^n$
$\mathbb{H}^n \times \mathbb{S}^n$
$\mathbb{S}^n \times \mathbb{S}^n$

Figure 4: The growth of the graph search space as a function of the number of model spaces used to generate product manifolds can be represented as a tree. Note that as discussed in Section **??** we assume commutativity: $\mathcal{M}_i \times \mathcal{M}_j = \mathcal{M}_j \times \mathcal{M}_i$.

To calculate the number of elements in the search space, we define the total number of products at level h of the tree as N. We have that $N(h) = N_E(h) + N_H(h) + N_S(h)$, where $N_E(h)$ is the number of Euclidean spaces added to the product manifolds at depth h of the tree, and $N_H(h)$ and $N_S(h)$ represent the same for the hyperboloid and the hypersphere respectively. By recursion, we can write

$$N_E(h) = N_E(h-1) = 1 \tag{33}$$

$$\begin{aligned} N_H(h) &= N_E(h-1) + N_H(h-1) && (34) \\ &= 1 + h && (35) \end{aligned}$$

$$\begin{aligned} N_S &= N_E(h-1) + N_H(h-1) + N_S(h-1) && (36) \\ &= \frac{h(h+1)}{2} N_E(1) + h N_H(1) + N_S(1) && (37) \\ &= \frac{h(h+1)}{2} + h + 1 && (38) \end{aligned}$$

hence we have that

$$\begin{aligned} N(h) &= 1 + (1+h) + (1 + h + \frac{h(h+1)}{2}) && (39) \\ &= 3 + \frac{5}{2}h + \frac{1}{2}h^2 && (40) \end{aligned}$$

To be consistent with the previous notation, we write the above in terms of the number of products n_p

$$N(n_p) = 3 + \frac{5}{2}n_p + \frac{1}{2}n_p^2 \tag{41}$$

The total number of nodes in the graph search space for a number of product n_p is then

$$N_T = \sum_{i=1}^{n_p} N(i) \tag{42}$$

B.6 Visualizing the Graph Search Space

In this appendix, we give a visual depiction of the graph search space we construct for neural latent geometry search. Figure 5 and Figure 6 provide plots of the graph search space as the size of the product manifolds considered increases. For product manifolds composed of a high number of model spaces, visualizing the search space becomes difficult. We omit the strengths of the connections for visual clarity in this plots.

If we focus on Figure 5, we use `e`, `h`, and `s` to refer to the model spaces of constant curvature $\mathbb{E}^2$, $\mathbb{H}^2$, and $\mathbb{S}^2$. In this work, we have considered model spaces of dimensionality two, but the graph search space would be analogous if we were to change the dimensionality of the model spaces. Nodes that include more than one letter, such as `hh`, `ss`, `ee`, etc, refer to product manifolds. For example, hh would correspond to the product manifold $\mathbb{H}^2 \times \mathbb{H}^2$. As discussed in Section **??**, we can see that connections are only allowed between product manifolds that differ by a single model space. To try to clarify this further, we can see that `e`, `h` and `s` are all interconnected since they only differ by a single model space (one deletion and one addition). Likewise, `e` is connected to `ee`, `eh`, and `ee` since they only differ by a single model space (one addition). However, `e` is not connected to `hs` (one deletion and two additions) nor is `ee` connected to `ss` (two deletions and two additions).

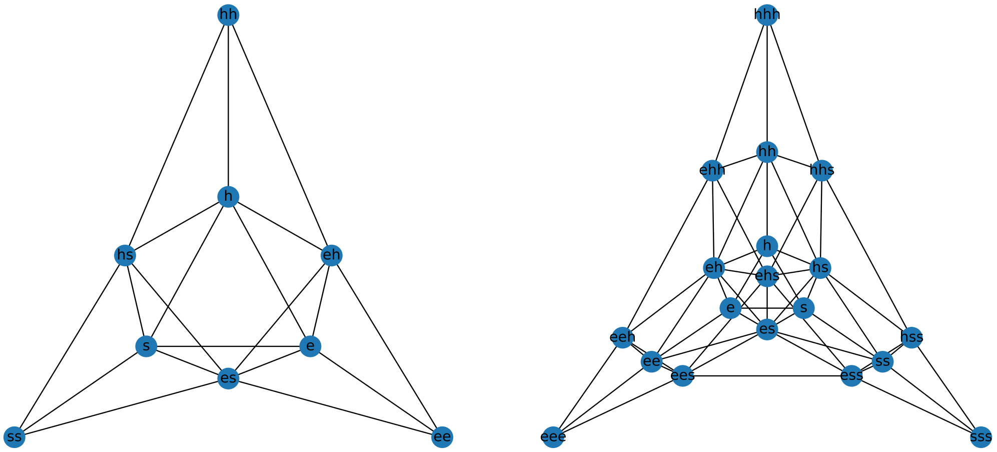

Figure 5: Graph search spaces for $n_p = 2$ (left) and $n_p = 3$ (right). Strength of connectivity is not depicted in the graph.

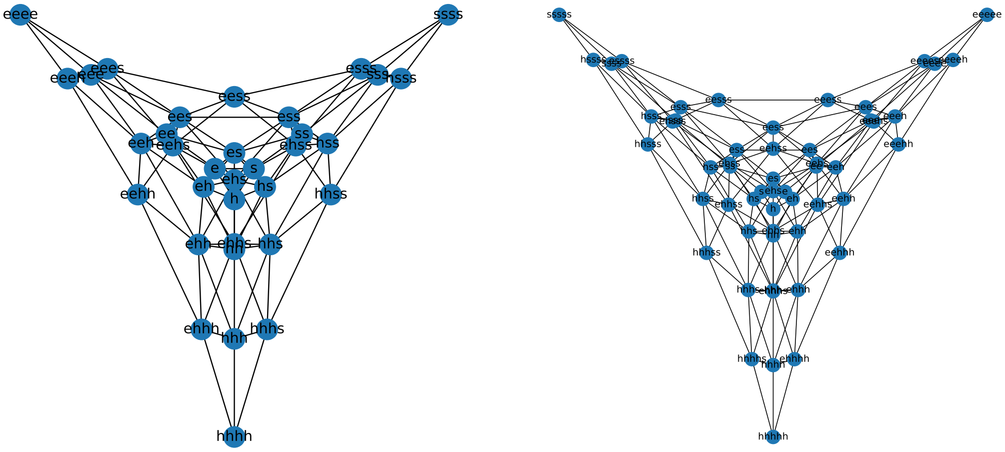

Figure 6: Graph search spaces for $n_p = 4$ (left) and $n_p = 5$ (right). Strength of connectivity is not depicted in the graph.

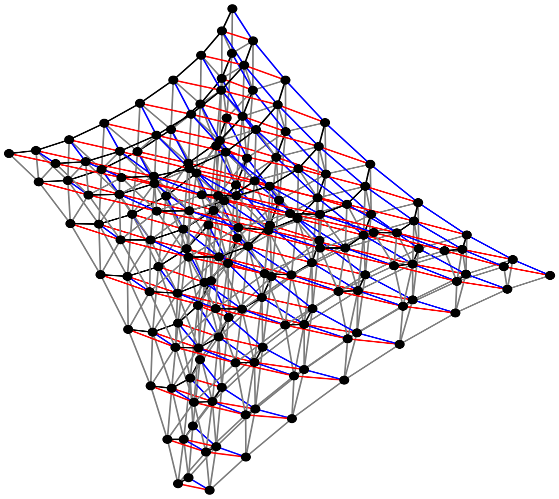

Figure 7: Graph search spaces for $n_p = 8$.

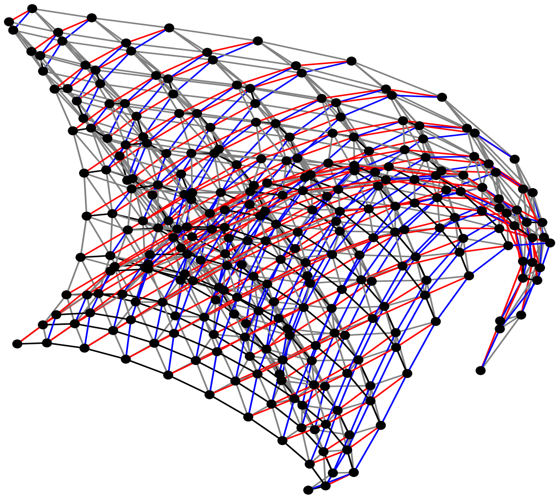

Figure 8: Graph search spaces for $n_p = 10$.

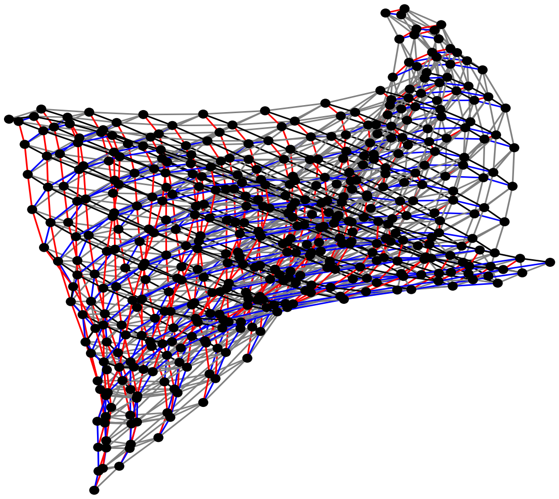

Figure 9: Graph search spaces for $n_p = 12$.

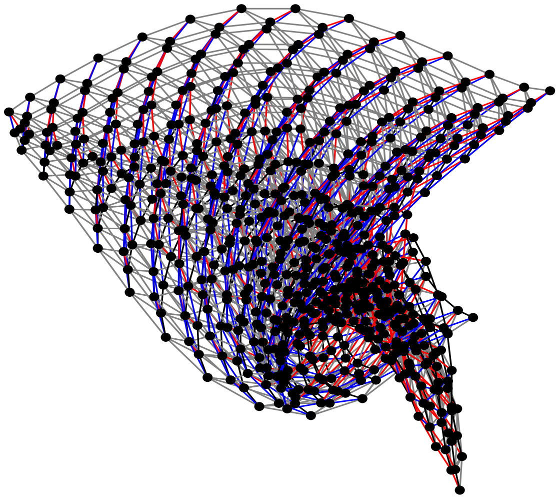

Figure 10: Graph search spaces for $n_p = 14$.

B.7 Motivating Gromov-Hausdorff Distances for Comparing Latent Geometries

The use of Gromov-Hausdorff distances can be motivated from a theoretical and practical perspective. From a theoretical perspective, the Gromov-Hausdorff distance offers a way to gauge the similarity between metric spaces. This is achieved by casting their representation within a shared space, all the while maintaining the original pairwise distances. In the context of machine learning models, particularly those involving generative models (such as VAEs or GANs), the latent space assumes a pivotal role in shaping the characteristics of the generated data. This also holds true in the case of reconstruction tasks, such as those carried out by autoencoders. Through the application of a metric that factors in their inherent geometries, the aim is to capture the concept of resemblance in data generation, reconstruction, and other downstream tasks which will be heavily affected by the choice of latent geometry.

While traditional metrics like Euclidean or cosine distances might suffice for some cases, they do not always capture the complex and nonlinear relationships between data points in high-dimensional spaces. The Gromov-Hausdorff distance considers the overall structure and shape of these spaces, rather than just individual distances, which can lead to better generalization and robustness. This is especially relevant when dealing with complex data distributions or high-dimensional latent spaces. Moreover, another appealing aspect of Gromov-Hausdorff distance is its invariance to isometric transformations, such as rotations, translations, and reflections. This is a desirable property when comparing latent spaces, as the metric used should reflect the similarity in shape and structure, rather than being influenced by trivial transformations.

While the direct relationship between Gromov-Hausdorff distance and model performance might not be immediately obvious, we argue that if two models have similar latent space geometries, this suggests that they might capture similar underlying data structures, which could lead to similar performance on downstream tasks. This is known as an assumption of *smoothness* in the NAS literature, see **?**. However, it is important to note that this relationship is not guaranteed, and the effectiveness of Gromov-Hausdorff distance as a proxy measure depends on the specific application and the nature of the models being compared.

B.8 Method Scalability

The proposed approach sidesteps scalability concerns by preventing the need to recalibrate GH coefficients, as long as one adheres to latent spaces derived from products of model spaces of dimension two. If a higher-dimensional latent space was needed, this could be solved by adding additional “graph slices" with augmented dimensions (as depicted in Figure **??**). These slices can be integrated with the lower-dimensional counterpart of the graph search space following the procedure described in the paper. The number of coefficients remains constant at 3, rendering the Gromov-Hausdorff distances free from extra computation overhead when increasing the dimension.

Moreover, in alignment with the manifold hypothesis, data is expected to exist within a low-dimensional manifold. This discourages the exploration of higher dimensions for latent space modeling. It is important to note, nonetheless, that the paper has already delved into relatively large search spaces, which include the exploration of high-dimensional latent spaces.

C Background on Bayesian Optimization

Bayesian optimization (BO) is a query-based optimization framework for black-box functions that are expensive to evaluate. It builds a probabilistic model of the objective function using past queries to automate the selection of meta-parameters and minimize computational costs. BO seeks to find the global optimum of a black-box function $\mathfrak{f} : \mathcal{X} \to \mathbb{R}$ by querying a point $\mathbf{x}_n \in \mathcal{X}$ at each iteration n and obtaining the value $y_n = \mathfrak{f}(\mathbf{x}_n) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is the noise in the observation. To do so, BO employs a surrogate to model the function $\mathfrak{f}(\cdot)$ being minimized given the input and output pairs $\mathcal{D}_t = \{(\mathbf{x}_i, y_i = \mathfrak{f}(x_i)\}_{i=1}^{N}$, and selects the next point to query by maximizing an *acquisition function*. In our work, we use the Gaussian Process (GP)(**?**) as the surrogate model and expected improvement (EI)(**?**) as the acquisition function.

Preliminaries. Bayesian optimization (BO) is particularly beneficial in problems for which evaluation is costly, behave as a black box, and for which it is impossible to compute gradients with respect to the loss function. This is the case when tuning the hyperparameters of machine learning models. In our case, we will use it to find the optimal product manifold signature. Effectively, Bayesian optimization allows us to automate the selection of critical meta-parameters while trying to minimize computational cost. This is done building a probabilistic proxy model for the objective using outcomes recorded in past experiments as training data. More formally, BO tries to find the global optimum of a black-box function $\mathfrak{f} : \mathcal{X} \to \mathbb{R}$. To do this, a point $\mathbf{x}_n \in \mathcal{X}$ is queried at every iteration n and yields the value $y_n = \mathfrak{f}(\mathbf{x}_n) + \varepsilon$, where $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, is a noisy observation of the function evaluation at that point. BO can be phrased using decision theory

$$L(\mathfrak{f}, \{\mathbf{x}_n\}_{n=1}^N) = N \times c + \min_{1 \leqslant n \leqslant N} \mathfrak{f}(\mathbf{x}_n), \tag{43}$$

where N is the maximum number of evaluations, c is the cost incurred by each evaluation, and the loss L is minimized by finding a lower $\mathfrak{f}(\mathbf{x})$. In general, Equation 43 is intractable and the closed-form optimum cannot be found, so heuristic approaches must be used. Instead of directly minimizing the loss function, BO employs a surrogate model to model the function $\mathfrak{f}(\cdot)$ being minimized given the input and output pairs $\mathcal{D}_t = \{(\mathbf{x}_i, y_i = \mathfrak{f}(x_i)\}_{i=1}^N$. Furthermore, in order to select the next point to query, an *acquisition function* is maximized. In this work, the surrogate model is chosen to be a Gaussian Process (GP) (**?**) and expected improvement (EI) (**?**) is used as the acquisition function.

Gaussian Processes. A GP is a collection of random variables such that every finite collection of those random variables has a multivariate normal distribution. A GP is fully specified by a *mean function*, $\mu(\mathbf{x})$, and *covariance function* (or *kernel*), denoted as $k(\mathbf{x}, \mathbf{x}')$. The prior over the mean function is typically set to zero, and most of the complexity of the model hence stems from the kernel function. Given t iterations of the optimization algorithm, with an input $\mathbf{x}_t = [x_1, \ldots, x_t]^T$ and output $\mathbf{y}_t = [y_1, \ldots, y_t]^T$, the posterior mean and variance are given by

$$\mu(x_{t+1}|\mathcal{D}_t) = \mathbf{k}(x_{t+1}, \mathbf{x}_t)[\mathbf{K}_{1:t} + \sigma_n^2 \mathbf{I}_t]^{-1}\mathbf{y}_t, \tag{44}$$

and

$$\sigma(x_{t+1}|\mathcal{D}_t) = \mathbf{k}(x_{t+1}, x_{t+1}) - \mathbf{k}(x_{t+1}, \mathbf{x}_t)[\mathbf{K}_{1:t} + \sigma_n^2 \mathbf{I}_t]^{-1}\mathbf{k}(\mathbf{x}_t, x_{t+1}), \tag{45}$$

where we define $[\mathbf{K}_{1:t}]_{i,j} = k(x_i, x_j)$ is the (i, j)-th element of the Gram matrix.

Expected Improvement. EI is a widely used acquisition function for Bayesian optimization. EI improves the objective function through a greedy heuristic which chooses the point which provides the greatest expected improvement over the current best sample point. EI is calculated as

$$\alpha_{\text{EI}}(\mathbf{x}) = \sigma(\mathbf{x})[\Gamma(\mathbf{x})\Phi(\Gamma(\mathbf{x})) + \mathcal{N}(\Gamma(\mathbf{x})|0, 1)], \tag{46}$$

where

$$\Gamma(\mathbf{x}) = \frac{\mathfrak{f}(\mathbf{x}_{best}) - \mu(\mathbf{x})}{\sigma(\mathbf{x})}, \tag{47}$$

and $\Phi(\cdot)$ denotes the CDF of a standard normal distribution.

D Experimental Setup

In this section we provide additional details for the implementation of the experimental setup, including how the datasets are generated, relevant theoretical background, and the hyperparameters used for BO over our discrete graph search space.

D.1 Synthetic Experiments

To assess the effectiveness of the proposed graph search space, we conduct a series of tests using synthetic data. Our objective is to establish a scenario in which we have control over the latent space and manually construct the optimal latent space product manifold. To achieve this, we initiated the process by generating a random vector, denoted as $\mathbf{x}$, and mapping it to a predetermined "ground truth" product manifold $\mathcal{P}_T$. This mapping is performed using the exponential map for the product manifold $\mathcal{P}_T$, which can be derived based on the model spaces of constant curvature that constitute it. Subsequently, the resulting projected vector is decoded via a neural network with fixed weights, denoted as f_θ, to yield a reference signal $y^{\mathcal{P}_T}$. The rationale behind employing a frozen network is to introduce a non-linear mapping from the latent space to the signal space. This approach aims to mimic the behavior of a trained network in a downstream task, while disregarding the specific task or model weights involved. By utilizing a frozen network, we can capture the essence of a non-linear mapping without relying on the exact details of the task or specific model weights. The primary goal is to recover this reference signal using the neural latent geometry search (NLGS) approach. To generate the remaining dataset, we employ the same random vector and mapped it to several other product manifolds, denoted as $\{\mathcal{P}_i\}_{i \in n_{\mathcal{P}}}$, comprising an equivalent number of model spaces as $\mathcal{P}_T$ but with distinct signatures. Decoding the projected vector using the same neural network produces a set of signals, denoted as $\{y^{\mathcal{P}_i}\}_{i \in n_{\mathcal{P}}}$. To populate the nodes of the graph search space, we assign the corresponding value of mean squared error (MSE) between $y^{\mathcal{P}_T}$ and $y^{\mathcal{P}_i}$ for each pair. In this manner, our search algorithm aims to identify the node within the graph that minimizes the error, effectively determining the latent geometry capable of recovering the original reference signal. A schematic of the proposed method and specifics on the construction of the network used to decode the signal are shown in Figure 11 and Table 2.

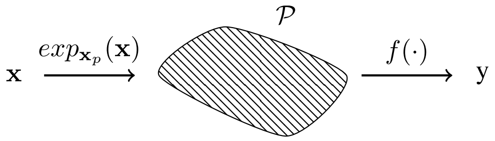

Figure 11: Schematic of procedure used to generate synthetic datasets. We model f as an MLP.

Table 2: Summary of network to generate synthetic datasets.

Model
Linear (data dim, 100) - ELU
Linear (100, 100) - ELU
Linear (100, 100) - ELU
Linear (100, 5) - ELU

D.2 Autoencoders

Autoencoders are a type of neural network architecture that can be used for unsupervised learning tasks. They are composed of two main parts: an encoder and a decoder. The encoder takes an input and compresses it into a low-dimensional representation, while the decoder takes that representation and generates an output that tries to match the original input. The goal of an autoencoder is to learn a compressed representation of the input data that captures the most important features, and can be used for tasks such as image denoising, dimensionality reduction, and anomaly detection. They have been used in a wide range of applications, including natural language processing, computer vision, and audio analysis.

Table 3: Autoencoder architecture summary.

Component	**Layers**
Encoder	Conv2d (1, 20, 3) - BatchNorm2d - SiLU
	Conv2d (20, 20, 3) - BatchNorm2d - SiLU
	⋮
	(9 repetitions)
	⋮
	Conv2d (20, 2, 3) - BatchNorm2d - SiLU
	Flatten - Linear - SiLU
Decoder	Linear - SiLU - Unflatten
	ConvTranspose2d (2, 20, 3) - BatchNorm2d - SiLU
	ConvTranspose2d (20, 20, 3) - BatchNorm2d - SiLU
	⋮
	(9 repetitions)
	⋮
	ConvTranspose2d (20, 1, 3) - BatchNorm2d - Sigmoid

The autoencoder's objective is to learn a compressed representation (latent space) of the input data and use it to reconstruct the original input as accurately as possible. In our experiments, the encoder takes an input image and applies a series of convolutional layers with batch normalization and SiLU activation functions, reducing the image's dimensions while increasing the number of filters or feature maps. The final output of the encoder is a tensor of shape $(\texttt{batchsize}, \texttt{latentdim}, 6, 6)$ or $(\texttt{batchsize}, \texttt{latentdim}, 10, 10)$, where `latentdim` is the desired size of the latent space. After flattening this tensor, a fully connected layer is used to map it to the desired latent space size. The encoder's output is a tensor of shape $(\texttt{batchsize}, \texttt{latentdim})$, which represents the compressed representation of the input image. The decoder takes the latent space representation as input and applies a series of transpose convolutional layers with batch normalization and SiLU activation functions, gradually increasing the image's dimensions while decreasing the number of feature maps. The final output of the decoder is an image tensor of the same shape as the input image. The loss functions used for training are the `MSELoss` (mean squared error) and `BCELoss` (binary cross-entropy) from the `PyTorch` library. A summary of the autoencoder is provided in Table 3.

D.3 Latent Graph Inference

Graph Neural Networks (GNNs) leverage the connectivity structure of graph data to achieve state-of-the-art performance in various applications. Most current GNN architectures assume a fixed topology of the input graph during training. Research has focused on improving diffusion using different types of GNN layers, but discovering an optimal graph topology that can help diffusion has only recently gained attention. In many real-world applications, data may only be accessible as a point cloud of data, making it challenging to access the underlying but unknown graph structure. The majority of Geometric Deep Learning research has relied on human annotators or simplistic pre-processing algorithms to generate the graph structure, and the correct graph may often be suboptimal for the task at hand, which may benefit from rewiring. Latent graph inference refers to the process of inferring the underlying graph structure of data when it is not explicitly available. In many real-world applications, data may only be represented as a point cloud, without any knowledge of the graph structure. However, this does not mean that the data is not intrinsically related, and its connectivity can be utilized to make more accurate predictions.

For these experiments we reproduce the architectures described in **?**, see Table 4 and Table 5 for the original architectures. In our case, we use the GCN-dDGM model leveraging the original input graph inductive bias.

Table 4: Summary of model architectures for experiments for Cora and CiteSeer.

		Model		
		MLP	**GCN**	**GCN-dDGM**
No. Layer parameters	Activation	Layer type		
		N/A	N/A	dDGM
(No. features, 32)	ELU	Linear	Graph Conv	Graph Conv
		N/A	N/A	dDGM
(32, 16)	ELU	Linear	Graph Conv	Graph Conv
		N/A	N/A	dDGM
(16, 8)	ELU	Linear	Graph Conv	Graph Conv
(8, 8)	ELU	Linear	Linear	Linear
(8, 8)	ELU	Linear	Linear	Linear
(8, No. classes)	-	Linear	Linear	Linear

Table 5: dDGM* and dDGM architectures for Cora and CiteSeer.

		dDGM*	**dDGM**
No. Layer parameters	Activation	Layer type	
(No. features, 32)	ELU	Linear	Linear
(32, 16 per model space)	ELU	Linear	Graph Conv
(16 per model space, 4 per model space)	Sigmoid	Linear	Graph Conv

D.3.1 Differentiable Graph Module

In their work, **?** presented a general method for learning the latent graph by leveraging the output features of each layer. They also introduced a technique to optimize the parameters responsible for generating the latent graph. The key concept is to use a similarity metric between the latent node features to generate optimal latent graphs for each layer l. In this context, $\mathbf{X}^{(0)}$ and $\mathbf{A}^{(0)}$ represent the original input node feature matrix and adjacency matrix, respectively. So that

$$\mathbf{X}^{(0)} = \begin{bmatrix} -\mathbf{x}_1^{(0)}- \\ -\mathbf{x}_2^{(0)}- \\ \vdots \\ -\mathbf{x}_n^{(0)}- \end{bmatrix}, \tag{48}$$

and $\mathbf{A}^{(0)} = \mathbf{A}$ if the adjacency matrix from the dataset, or $\mathbf{A}^{(0)} = \mathbf{I}$ if $\mathcal{G} = (\mathcal{V}, \varnothing)$. The proposed architecture in **?** consists of two primary components: the Differentiable Graph Module (DGM) and the Diffusion Module. The Diffusion Module, $\mathbf{X}^{(l+1)} = g_{\boldsymbol{\phi}}(\mathbf{X}^{(l)}, \mathbf{A}^{(l)})$, may be one (or multiple) standard GNN layers. The first DGM module takes the original node features and connectivity information and produces an updated adjacency matrix

$$\mathbf{X'}^{(l=1)}, \mathbf{A}^{(l=1)} = \text{DGM}(\mathbf{X}^{(0)}, \mathbf{A}^{(0)}). \tag{49}$$

In principle, the DGM module utilizes information from previous layers to generate adjacency matrices at each layer

$$\mathbf{X'}^{(l+1)}, \mathbf{A}^{(l+1)} = \text{DGM}(concat(\mathbf{X}^{(l)}, \mathbf{X'}^{(l)}), \mathbf{A}^{(l)}). \tag{50}$$

To do so, a measure of similarity is used

$$\varphi(\mathbf{x'}_i^{(l+1)}, \mathbf{x'}_j^{(l+1)}) = \varphi(f_{\boldsymbol{\Theta}}^{(l)}(\mathbf{x}_i^{(l)}), f_{\boldsymbol{\Theta}}^{(l)}(\mathbf{x}_j^{(l)})). \tag{51}$$

In summary, the proposed approach utilizes a parameterized function $f_{\boldsymbol{\Theta}^{(l)}}$ with learnable parameters to transform node features and a similarity measure φ to compare them. The function can be an MLP or composed of GNN layers if connectivity information is available. The output of the similarity measure is used to create a fully-connected weighted adjacency matrix for the continuous differentiable graph module (cDGM) approach or a sparse and discrete adjacency matrix for the discrete Differentiable Graph Module (dDGM) approach, with the latter being more computationally efficient and recommended by the authors of the DGM paper (**?**). To improve the similarity measure φ and construct better latent graphs, the approach employs product spaces and Riemannian geometry. Additionally, an extra loss term is used to update the learnable parameters of the dDGM module.

Lastly, we will examine the dDGM module, which utilizes the Gumbel Top-k (**?**) technique to generate a sparse k-degree graph by stochastically sampling edges from the probability matrix $\mathbf{P}^{(l)}(\mathbf{X}^{(l)}; \mathbf{\Theta}^{(l)}, T)$, which is a stochastic relaxation of the kNN rule, where each entry corresponds to

$$p_{ij}^{(l)} = \exp(-\varphi(T)(\mathbf{x}'^{(l+1)}_i, \mathbf{x}'^{(l+1)}_j)). \tag{52}$$

T being a learnable parameter. The primary similarity measure utilized in **?** involved computing the distance between the features of two nodes in the graph embedding space

$$p_{ij}^{(l)} = \exp(-T\Delta(\mathbf{x}'^{(l+1)}_i, \mathbf{x}'^{(l+1)}_j)) = \exp(-T\Delta(f^{(l)}_{\mathbf{\Theta}}(\mathbf{x}^{(l)}_i), f^{(l)}_{\mathbf{\Theta}}(\mathbf{x}^{(l)}_j)), \tag{53}$$

where $\Delta(\cdot, \cdot)$ denotes a generic measure of distance between two points. They assumed that the latent features laid in an Euclidean plane of constant curvature $K_{\mathbb{E}} = 0$, so that

$$p_{ij}^{(l)} = \exp(-T\mathfrak{d}_{\mathbb{E}}(f^{(l)}_{\mathbf{\Theta}}(\mathbf{x}^{(l)}_i), f^{(l)}_{\mathbf{\Theta}}(\mathbf{x}^{(l)}_j))), \tag{54}$$

where $\mathfrak{d}_{\mathbb{E}}$ is the distance in Euclidean space. Based on

$$\text{argsort}(\log(\mathbf{p}^{(l)}_i) - \log(-\log(\mathbf{q}))) \tag{55}$$

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

$$\mathcal{E}^{(l)}(\mathbf{X}^{(l)}; \mathbf{\Theta}^{(l)}, T, k) = \{(i, j_{i,1}), (i, j_{i,2}), ..., (i, j_{i,k}) : i = 1, ..., N\}, \tag{56}$$

k being the number of sampled connections using the Gumbel Top-k trick. The Gumbel Top-k approach utilizes the categorical distribution $\frac{p^{(l)}_{ij}}{\Sigma_r p^{(l)}_{ir}}$ for sampling, and the resulting unweighted adjacency matrix $\mathbf{A}^{(l)}(\mathbf{X}^{(l)}; \mathbf{\Theta}^{(l)}, T, k)$ is used to represent $\mathcal{E}(\mathbf{X}^{(l)}; \mathbf{\Theta}^{(l)}, T, k)$. It is worth noting that including noise in the edge sampling process can generate random edges in the latent graphs, which can serve as a form of regularization.

Finally, we can summarize a multi-layer GNN using the dDGM module as

$$\mathbf{X}'^{(l+1)} = f^{(l)}_{\mathbf{\Theta}}(concat(\mathbf{X}^{(l)}, \mathbf{X}'^{(l)}), \mathbf{A}^{(l)}), \tag{57}$$

$$\mathbf{A}^{(l+1)} \sim \mathbf{P}^{(l)}(\mathbf{X}'^{(l+1)}), \tag{58}$$

$$\mathbf{X}^{(l+1)} = g_{\boldsymbol{\phi}}(\mathbf{X}^{(l)}, \mathbf{A}^{(l+1)}). \tag{59}$$

Equations 57 and 58 belong to the dDGM module, while Equation 59 is associated with the Diffusion Module. In our study, we extend Equation 52 to measure distances without relying on the assumption utilized in **?**, which restricts the analysis to fixed-curvature spaces, specifically to Euclidean space where $K_{\mathbb{E}} = 0$.

D.4 Naive Bayesian Optimization

Naive BO in the main text considers performing BO over a fully-connected unweighted graph. Such a graph is latent geometry agnostic, that is, it does not include any metric geometry inductive bias to provide the optimization algorithm with a sense of closeness between the candidate latent geometries.

E Ablation of the Gromov-Hausdorff Coefficients

We evaluate the impact of Gromov-Hausdorff coefficients on the graph search space by conducting an additional set of experiments. We compare the use of an unweighted, pruned graph search space with the introduction of Gromov-Hausdorff coefficients in the datasets considered in this paper. The results are show in Figures 12 to 15 below.

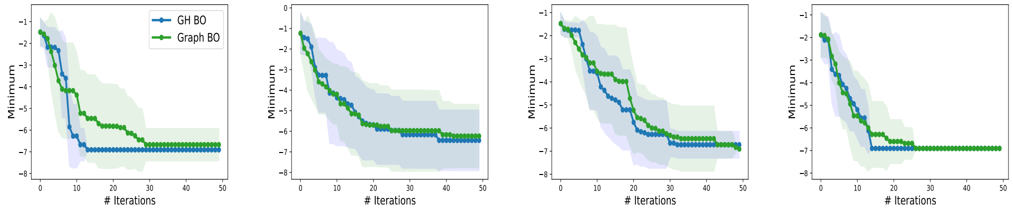

Figure 12: Results (mean and standard deviation over 10 runs) for candidate latent geometries involving product manifolds composed of 13 model spaces. For each plot a different ground truth product manifold $\mathcal{P}_T$ is used to generate the reference signal.

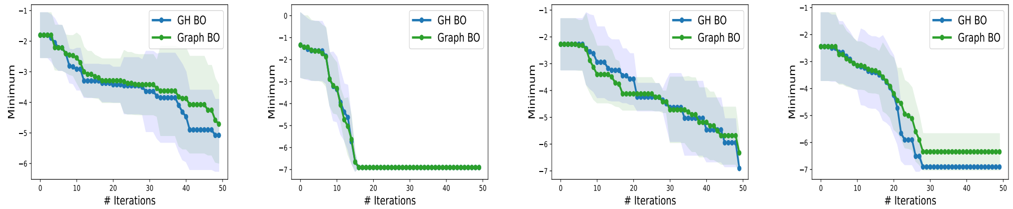

Figure 13: Results (mean and standard deviation over 10 runs) for candidate latent geometries involving product manifolds composed of 15 model spaces. For each plot a different ground truth product manifold $\mathcal{P}_T$ is used to generate the reference signal.

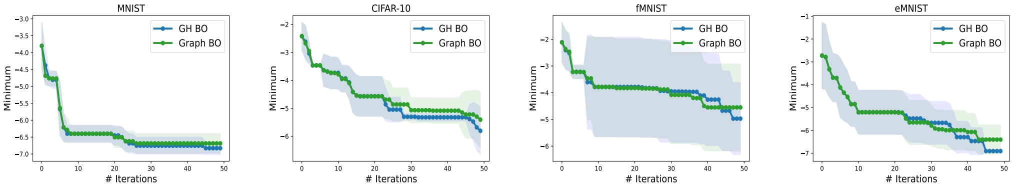

Figure 14: Results (mean and standard deviation over 10 runs) on image reconstruction tasks, for $n_p = 7$.

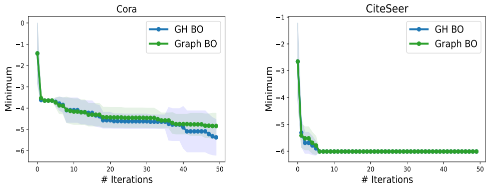

Figure 15: Results (mean and standard deviation over 10 runs) on latent graph inference tasks.

The plots above show that in the majority of cases, the GH weights appear to either match or enhance the performance of the search algorithm. This observation is particularly pronounced in synthetic tasks, which can be intuitively justified due to the significant impact of changes in the latent geometry on the network's output in such a setup. However, we highlight that a large amount of the performance gains in the algorithm seem to come from the inductive bias endowed to the search algorithm through the graph search space. This could be due to the fact that in real world tasks, searching for the optimal dimension is more important for reconstruction than finding the optimal latent geometry in a particular dimension. Since the GH coefficients are unity between dimensions, this makes the performance of the two search spaces similar.

It should be noted that setting the inverse of the Gromov-Hausdorff distance as the edges of a search graph is only one potential way to add a geometric inductive bias into the search space. Future work could use similar mechanisms as those developed in this paper but in a different context in order to improve optimization performance. Finally, we would like to highlight that a more thorough analysis of the links between the performance of neural networks and the similarity of their latent geometries is a relevant question that merits further study, which is left for future work.