PROJECTIONS OF MODEL SPACES FOR LATENT GRAPH INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks leverage the connectivity structure of graphs as an inductive bias. Latent graph inference focuses on learning an adequate graph structure to diffuse information on. In this work we employ stereographic projections of the hyperbolic and spherical model spaces, as well as products of Riemannian manifolds, for the purpose of latent graph inference. Stereographically projected model spaces achieve comparable performance to their non-projected counterparts, while providing theoretical guarantees that avoid divergence of the spaces when the curvature tends to zero.

1 INTRODUCTION

Differential geometry has been widely used in physics; for instance, it has laid the mathematical foundations of the theory of general relativity as well as the gauge theory of quantum fields (Isham, 1989). Moreover, recent work within the machine learning community has started leveraging ideas which stem from differential geometry and topology to improve the performance of learning algorithms (Hensel et al., 2021; Bodnar et al., 2022; Barbero et al., 2022a;b). Real-world data, which may manifest itself in high-dimensional observational space, can often be described via a lower-dimensional manifold according to the manifold hypothesis (Fefferman et al., 2013). Learning the underlying manifold which best describes the data can be a challenging task, but it is at the same time crucial for data modeling (Chen et al., 2022; Du et al., 2020; 2021). We can hypothesize that even data which may only be observable in the form of disconnected point clouds, could be explained using a connected *latent graph*, which is itself a discretization of a *latent manifold*. Such latent graph can be applied to diffuse information through layers of Graph Neural Networks (GNNs) which are able to leverage geometric priors as inductive biases. At the same time, a better latent graph can be generated if an appropriate embedding space (or latent manifold) to describe the observed data is found.

Several recent works have focused on optimizing the generation of latent graphs alongside the rest of the GNN model parameters (Sáez de Ocáriz Borde et al., 2022; Kazi et al., 2022; Cosmo et al., 2020) as well as on preprocessing the original input graph provided by the dataset (Topping et al., 2021; Gasteiger et al., 2019). This idea is known as *latent graph inference*. Other graph structure learning techniques include those by Franceschi et al. (2019); Chen et al. (2020); Jin et al. (2020). The discrete Differentiable Graph Module (dDGM) by Kazi et al. (2022) generates latent graphs based on Euclidean latent representations of the GNN layer activations. Sáez de Ocáriz Borde et al. (2022) recently incorporated Riemannian geometry to this approach to generate more complex latent representations of the data, which need not be Euclidean. In particular, they used product manifolds of model spaces (the Euclidean plane, the hyperboloid, and the hypersphere) to approximate complex latent manifolds which help generate better latent graphs for improved downstream performance. In this work, we build upon this idea and introduce projections of model spaces to generate product manifolds. The use of these projections affords us improved theoretical properties, which guarantee that the model spaces will not diverge at zero curvature locations and that their distance and metric tensors will become Eucledian in these areas of the manifold. Furthemore, the use of these projected model spaces enables the creation of more diverse product manifold combinations while retaining computational tractability.

2 BACKGROUND

2.1 GRAPH NEURAL NETWORKS

GNNs are a class of artificial neural networks that have gained popularity due to their ability to exploit the geometric prior present in data as an inductive bias (Bronstein et al., 2021). The main limitation of these models is that they require an input adjacency matrix that describes the connectivity of the graph on which they diffuse information. Furthermore, the initial connectivity structure is kept fixed even if it is unfriendly for message passing (Topping et al., 2021). GNNs generalize classical artificial neural networks to graphs $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes and $\mathcal{E}$ the set of edges. Each node in the graph has an associated d-dimensional feature vector $\mathbf{x}_v$. These can be grouped into a single matrix $\mathbf{X}$, and we can represent the set of all edges $\mathcal{E}$ via the adjacency matrix $\mathbf{A}$. GNN layers leverage these two matrices to transform features into a new set of latent features for each node $\mathbf{H}^{(l)} = f\left(\mathbf{H}^{(l-1)}, \mathbf{A}\right)$, most often sharing information in the 1-hop neighbourhood only; hence, stacking multiple layers allows the network to share information between more neighbors. Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017) and Graph Attention Networks (GATs) (Veličković et al., 2018) are two examples of well-known GNN architectures.

2.2 LATENT GRAPH INFERENCE LEVERAGING PRODUCT MANIFOLDS

Latent graph inference, in the case of the dDGM module, consists on finding some measure of similarity between the latent node features to connect similar nodes and generate good latent graphs that are optimized based on a downstream task loss. In particular the probability of there being an edge between each node in the graph is $p_{ij}^{(l)}(\mathbf{\Theta}^{(l)}) = \exp(\varphi(f_{\mathbf{\Theta}}^{(l)}(\mathbf{x}_i^{(l)}), f_{\mathbf{\Theta}}^{(l)}(\mathbf{x}_j^{(l)}); T))$, where $\mathbf{\Theta}$ are the model parameters, T is a learnable temperature hyperparameter, and most importantly, φ is a measure of similarity between the features of each node in the graph. It is of key importance to find a meaningful and flexible measure of similarity φ to improve downstream performance. This can be done using a distance metric based on the geodesic distance defined on a manifold $\mathcal{P}$. For computational tractability we define $\mathcal{P}$ to be a product manifold composed of Riemannian manifolds of constant curvature so that the probability of there existing an edge between node i and j in layer l of the GNN can be expressed as

$$p_{ij}^{(l)} = \exp(-T\mathfrak{d}_{\mathbb{P}}(f_{\mathbf{\Theta}}^{(l)}(\mathbf{x}_i^{(l)}), f_{\mathbf{\Theta}}^{(l)}(\mathbf{x}_j^{(l)}))), \tag{1}$$

where $\mathfrak{d}_{\mathbb{P}}$ is the distance geodesic function between two points on the product manifold, and $\exp(x) = e^{-x}$ in this context. Based on this we can sample the edges of our latent graphs $\mathcal{E}^{(l)}(\mathbf{X}^{(l)}; \mathbf{\Theta}^{(l)}, T, k) = \{(i, j_{i,1}), (i, j_{i,2}), ..., (i, j_{i,k}) : i = 1, ..., N\}$, where k is the number of sampled connections using the Gumbel Top-k trick. These edges are used to generate latent adjacency matrices $\mathbf{A}^{(l)}$, as described in Kazi et al. (2022).

A manifold is a topological space in which all of its points have neighbourhoods that are homeomorphic to distinct open subsets of Euclidean hyperplanes. In particular, we define a Riemannian manifold $(\mathcal{M}, g)$ as a real and differentiable manifold $\mathcal{M}$ for which each tangent space has an inner product g. This is called the Riemannian metric and varies smoothly between points on the manifold, meaning that it is continuous. The three so-called model spaces with constant curvature are the Euclidean plane, $\mathbb{E}_{K_{\mathbb{E}}}^{d_{\mathbb{E}}} = \mathbb{R}^{d_{\mathbb{E}}}$, where the curvature $K_{\mathbb{E}} = 0$; the hyperboloid, $\mathbb{H}_{K_{\mathbb{H}}}^{d_{\mathbb{H}}} = \{\mathbf{x}_p \in \mathbb{R}^{d_{\mathbb{H}}+1} : \langle \mathbf{x}_p, \mathbf{x}_p \rangle_{\mathcal{L}} = 1/K_{\mathbb{H}}\}$, where $K_{\mathbb{H}} < 0$ and $\langle \cdot, \cdot \rangle_{\mathcal{L}}$ is the Lorentz inner product; and the hypersphere, $\mathbb{S}_{K_{\mathbb{S}}}^{d_{\mathbb{S}}} = \{\mathbf{x}_p \in \mathbb{R}^{d_{\mathbb{S}}+1} : \langle \mathbf{x}_p, \mathbf{x}_p \rangle_2 = 1/K_{\mathbb{S}}\}$, where $K_{\mathbb{S}} > 0$ and $\langle \cdot, \cdot \rangle_2$ is the standard Euclidean inner product. Each of these model spaces has an associated exponential maps to project the Euclidean outputs of GNN layers to each of the manifolds, and distance functions to compute distances between points on the manifolds, which are discussed in Sáez de Ocáriz Borde et al. (2022). Product manifolds can be constructed using the Cartesian product $\mathcal{P} = \times_{i=1}^{n_{\mathcal{P}}} \mathcal{M}_{K_i}^{d_i}$, of other manifolds with curvature K_i and dimensionality d_i. The distance between two points on the manifold can be computed by aggregating the distance contributions from each manifold composing the product manifold as follows

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

where the overline denotes that the points $\overline{\mathbf{x}}_{p_1}$ and $\overline{\mathbf{x}}_{p_2}$ have been adequately projected on to the product manifold using the exponential map before computing the distances, and $\mathfrak{d}_i$ is the distance

function in each of the individual model spaces used to construct the overall product manifold. Note that the exponential map to project points from an Euclidean tangent plane, $\mathcal{T}_p\mathcal{P}$, onto the product manifold, $\mathcal{P}$, which is the concatenation of the exponential maps for each of its model space components. In turn, the manifold signature is the number of model spaces used to generate product manifolds as well as their individual dimensionalities.

3 Latent Graph Inference using Projections of Model Spaces

With the three previously detailed model spaces, it is possible to get any value of the curvature in the range $(-\infty, \infty)$. However, a motivating factor to introduce projections of model spaces is the fact that points on the hyperboloid and hypersphere become divergent as $K_{\mathbb{S}} \to 0$ and $K_{\mathbb{H}} \to 0$. This effectively means that the manifolds become flat to stay consistent with their respective definitions. Furthermore, their distance and metric tensors do not become Eucledian at zero-curvature locations of the manifold, which may hinder curvature learning (Skopek et al., 2019). We therefore consider the stereographic projections as suitable alternatives to these spaces, as they maintain a non-Euclidean structure and inherit many of the properties of the hyperboloid and the hypersphere.

3.1 The Poincaré Ball Model

Hyperbolic space is infinite; nevertheless, we can create a finite projection of hyperbolic space onto a flat surface. This is because, in principle, there are infinite many points within a disk. The Poincaré ball, also referred to as Poincaré disk in 2D, is a model of n-dimensional hyperbolic geometry which embeds all points in an n-dimensional hypersphere. It is a stereographic projection of the hyperboloid, meaning that it preserves angles but not distances. In particular, the distances between points grows exponentially as we move towards the outside of the Poincaré ball compared to their Euclidean distances. From here on, we will define the Poincaré ball as $\mathbb{P}_{K_{\mathbb{P}}}^{d_{\mathbb{P}}}$, where $K_{\mathbb{P}}$ denotes the curvature of the space and $d_{\mathbb{P}}$ its dimensionality. Following (Cetin et al., 2022), distances between points in a Poincaré ball are given by

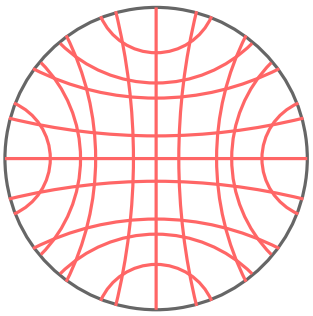

Figure 1: Grid of geodesics for the Poincaré ball.

$$\mathfrak{d}_{\mathbb{P}} = \frac{1}{\sqrt{-K^{\mathbb{P}}}} \text{arccosh}\left(1 - \frac{2K^{\mathbb{P}}||\mathbf{x}-\mathbf{y}||_2^2}{(1+K^{\mathbb{P}}||\mathbf{x}||_2^2)(1+K^{\mathbb{P}}||\mathbf{y}||_2^2)}\right), \tag{3}$$

and its exponential map is in turn $exp_{\mathbf{x}_p}^{\mathbb{P}}(\mathbf{x}) = \tanh(\sqrt{-K^{\mathbb{P}}}||\mathbf{x}||_2)\frac{\mathbf{x}}{\sqrt{-K^{\mathbb{P}}}||\mathbf{x}||_2}$ where the curvature $K_{\mathbb{P}} < 0$. The negative curvature is inherited from the hyperboloid. The GNN output at layer l passes through a parametric transformation to obtain $\hat{\mathbf{x}}^{(l+1)} = f_{\boldsymbol{\Theta}}^{(l)}(\mathbf{x}^{(l)})$. $\hat{\mathbf{x}}^{(l+1)}$ is originally assumed to be on the tangent plane of the Poincaré manifold $\mathbb{P}_{K_{\mathbb{P}}}^{d_{\mathbb{P}}}$, so that $\hat{\mathbf{x}}^{(l+1)} \in \mathcal{T}_p\mathbb{P}_{K_{\mathbb{P}}}^{d_{\mathbb{P}}}$ and this is later projected to the actual manifold via the exponential map to obtain $\overline{\mathbf{x}}^{(l+1)} = exp_{\mathbf{x}_p}^{\mathbb{P}}(\hat{\mathbf{x}}^{(l+1)}) \in \mathbb{P}_{K_{\mathbb{P}}}^{d_{\mathbb{P}}}$. The geodesic distance function in Equation 3 can finally be applied to the projected points. Hence, the exponential map and distance function give us a principled way of mapping the latent features to the Poincaré ball.

3.2 The Stereographic Projection of the Hypersphere Model

The previous discussion also applies to the stereographic projection of the hypersphere. However, in this case the projected space has positive curvature. If we think about the lower-dimensional case in which we project a sphere onto a plane, in the polar aspect, the meridians project as straight lines originating at the pole and all angles between them remain truthful to those on the original manifold. On the other hand, parallels on the sphere would be shown as concentric circular arcs on the plane, with their spacing rapidly increasing from the pole, and distortion values remain the same along circular arcs surrounding the center point. The distance function for any arbitrary dimensionality is given by

$$\mathfrak{d}_{\mathbb{D}} = \frac{1}{\sqrt{K^{\mathbb{D}}}} \arccos\left(1 - \frac{2K^{\mathbb{D}}||\mathbf{x}-\mathbf{y}||_2^2}{(1+K^{\mathbb{D}}||\mathbf{x}||_2^2)(1+K^{\mathbb{D}}||\mathbf{y}||_2^2)}\right). \tag{4}$$

The exponential map from a tangent Euclidean plane to the stereographic projection of the hypersphere is $exp_{\mathbf{x}_p}^{\mathbb{D}}(\mathbf{x}) = \tan(\sqrt{K^{\mathbb{D}}}||\mathbf{x}||_2)\frac{\mathbf{x}}{\sqrt{K^{\mathbb{D}}}||\mathbf{x}||_2}$.

4 RESULTS

In Table 1 we display the results using both the Poincaré ball model and the stereographic projection of the hypersphere in isolation as embedding spaces, as well as part of product manifolds. We compare the results using the newly generated product manifolds against those previously obtained using the Euclidean plane, the hyperboloid, and the hypersphere for several homophilic, heterophilic, as well as real-world graph datasets. Note that the $*$ symbol after the dDGM* name denotes that the model does not use any graph provided by the dataset as an inductive bias. In other words, the dDGM* model generates the latent graph based solely on an input point cloud and its features $\mathbf{X}$. On the other hand, when the asterick is not present in the model name, the model uses the adjacency matrix $\mathbf{A}$ provided in the original dataset as inductive bias to generate the latent graph as well.

The stereographically projected models do sometimes outperform the original model spaces they were projected from and in general obtain comparable results. This is particularly the case for the homophilic datasets, Cora and Citeseer, and the TadPole dataset. However, in the case of Squirrel and Chameleon, for example, we do not observe any improvement with respect to the previous best-performing manifold signature combinations. Hence, we can conclude that using stereographic projections it is sometimes possible to outperform the original model space they were derived from while guaranteeing that the spaces do not become divergent as curvature tends to zero, and that their distance and metric tensors become Euclidean at zero-curvature locations of the manifold. More importantly, the introduction of these projections opens the possibility of constructing more diverse manifold signatures, while retaining the computational tractability of product manifolds.

Table 1: Results for benchmark homophilic, heterophilic, and real-world graph datasets. The **First**, **Second**, and **Third** best models for each dataset are highlighted in each table. k denotes the number of connections per node when implementing the Gumbel Top-k sampling algorithm.

	HOMOPHILIC DATASETS			HETEROPHILIC DATASETS			REAL-WORLD DATASETS
	Cora	**CiteSeer**		**Squirrel**	**Chameleon**		**TadPole**
Nodes	2,708	3,327		5,201	2,277		564
Features	1,433	3,703		2,089	2,325		30
Classes	7	6		5	5		3
New Models	Accuracy (%) ± Std		**New Models**	Accuracy (%) ± Std		**New Models**	Accuracy (%) ± Std
Connections k	7	7	Connections k	3	5	Connections k	3
GCN-dDGM-P	**85.07 ± 5.42**	**73.94 ± 1.97**	GCN-dDGM*-P	21.38 ± 2.66	22.9 ± 3.97	GAT-dDGM*-P	90.54 ± 4.23
GCN-dDGM-D	76.00 ± 6.90	71.98 ± 4.09	GCN-dDGM*-D	32.67 ± 1.73	46.91 ± 3.11	GAT-dDGM*-D	57.32 ± 11.33
Connections k	5	3	Connections k	3	5	Connections k	3
GCN-dDGM-HP	**85.07 ± 3.41**	73.31 ± 1.92	GCN-dDGM*-HP	20.63 ± 1.74	23.74 ± 3.16	GAT-dDGM*-HP	86.25 ± 6.29
GCN-dDGM-SD	83.96 ± 3.52	37.19 ± 24.88	GCN-dDGM*-SD	32.59 ± 2.62	**48.46 ± 1.47**	GAT-dDGM*-SD	87.85 ± 4.05
GCN-dDGM-EP	85.07 ± 4.46	73.28 ± 1.95	GCN-dDGM*-EP	20.53 ± 2.27	21.27 ± 4.56	GAT-dDGM*-EP	88.21 ± 4.80
GCN-dDGM-ED	59.22 ± 25.61	46.86 ± 26.24	GCN-dDGM*-ED	33.73 ± 2.26	43.88 ± 2.94	GAT-dDGM*-ED	88.92 ± 4.64
GCN-dDGM-HD	64.70 ± 26.32	50.81 ± 27.48	GCN-dDGM*-HD	21.03 ± 1.95	21.94 ± 4.04	GAT-dDGM*-HD	**91.07 ± 4.37**
GCN-dDGM-EHP	82.67 ± 4.19	**73.82 ± 2.64**	GCN-dDGM*-EHP	21.73 ± 3.06	22.16 ± 3.48	GAT-dDGM*-EHP	88.75 ± 3.75
GCN-dDGM-EHD	52.40 ± 27.38	30.30 ± 21.17	GCN-dDGM*-EHD	20.36 ± 1.55	21.45 ± 3.65	GAT-dDGM*-EHD	90.71 ± 3.36
GCN-dDGM-ESP	83.48 ± 8.93	73.55 ± 1.70	GCN-dDGM*-ESP	20.53 ± 2.40	22.95 ± 6.08	GAT-dDGM*-ESP	90.00 ± 4.67
GCN-dDGM-ESD	51.92 ± 27.41	25.21 ± 16.82	GCN-dDGM*-ESD	33.19 ± 1.84	45.63 ± 2.64	GAT-dDGM*-ESD	89.82 ± 6.34
GCN-dDGM-EPP	**85.96 ± 4.01**	73.79 ± 3.11	GCN-dDGM*-EPP	22.46 ± 4.47	19.42 ± 1.81	GAT-dDGM*-EPP	88.39 ± 4.31
GCN-dDGM-EPD	47.03 ± 26.07	47.56 ± 26.51	GCN-dDGM*-EPD	19.84 ± 1.40	21.54 ± 4.87	GAT-dDGM*-EPD	89.82 ± 3.09
GCN-dDGM-EDD	41.11 ± 21.95	31.11 ± 22.91	GCN-dDGM*-EDD	33.15 ± 3.08	44.27 ± 3.68	GAT-dDGM*-EDD	**91.07 ± 3.09**
GCN-dDGM-EHSPD	30.03 ± 3.16	30.24 ± 22.61	GCN-dDGM*-EHSPD	19.48 ± 1.17	21.18 ± 4.29	GAT-dDGM*-EHSPD	**91.07 ± 3.82**
Baselines	Accuracy (%) ± Std		**Baselines**	Accuracy (%) ± Std		**Baselines**	Accuracy (%) ± Std
Connections k	7	7	Connections k	3	5	Connections k	3
GCN-dDGM*-E	82.11 ± 4.24	72.35 ± 1.92	GCN-dDGM*-E	34.35 ± 2.34	**48.90 ± 3.61**	GAT-dDGM*-E	90.36 ± 3.21
GCN-dDGM*-H	84.68 ± 3.31	70.43 ± 4.95	GCN-dDGM*-H	**35.00 ± 2.35**	48.28 ± 4.11	GAT-dDGM*-H	88.75 ± 3.91
GCN-dDGM*-S	80.44 ± 5.26	72.89 ± 2.00	GCN-dDGM*-S	33.12 ± 2.22	**48.63 ± 3.12**	GAT-dDGM*-S	**90.89 ± 4.55**
Connections k	5	3	Connections k	3	5	Connections k	3
GCN-dDGM-HH	76.09 ± 7.11	71.27 ± 2.09	GCN-dDGM*-HH	**34.38 ± 1.07**	48.33 ± 4.14	GAT-dDGM*-HH	89.68 ± 5.70
GCN-dDGM-SS	65.96 ± 9.46	59.16 ± 5.96	GCN-dDGM*-SS	34.06 ± 2.20	48.28 ± 3.07	GAT-dDGM*-SS	89.82 ± 4.79
GCN-dDGM-EH	82.32 ± 4.71	72.89 ± 1.64	GCN-dDGM*-EH	34.37 ± 1.72	47.58 ± 3.85	GAT-dDGM*-EH	90.36 ± 4.16
GCN-dDGM-ES	81.44 ± 5.80	71.87 ± 3.20	GCN-dDGM*-ES	33.38 ± 1.86	47.49 ± 3.60	GAT-dDGM*-ES	89.46 ± 5.56
GCN-dDGM-HS	82.59 ± 4.50	72.77 ± 2.76	GCN-dDGM*-HS	**34.65 ± 2.45**	47.84 ± 2.67	GAT-dDGM*-HS	86.43 ± 5.82
GCN-dDGM-EHH	**86.63 ± 3.25**	**75.42 ± 2.39**	GCN-dDGM*-EHH	33.19 ± 1.92	44.27 ± 2.96	GAT-dDGM*-EHH	87.68 ± 9.95
GCN-dDGM-EHS	83.58 ± 4.39	69.98 ± 2.70	GCN-dDGM*-EHS	34.17 ± 2.23	47.58 ± 4.54	GAT-dDGM*-EHS	**92.68 ± 3.52**
GCN	83.11 ± 2.29	69.97 ± 2.06	GCN	24.19 ± 2.56	32.56 ± 3.53	GAT	N/A
MLP	58.92 ± 3.28	59.48 ± 2.14	MLP	30.44 ± 2.55	40.35 ± 3.37	MLP	87.68 ± 3.52

5 CONCLUSION

In this work we have proposed exploring the application of stereographically projected non-Euclidean model spaces for latent graph inference. These model spaces encode non-Euclidean geometries in a lower-dimensional manifold, and avoid divergence as their curvature tends to zero unlike their original counterparts. Effectively, these new projected spaces allow us to generate even more diverse product manifolds to try to infer the latent space of the data. However, given the large number of possible product manifold combinations, it has become apparent that future research should involve devising a principled way of inferring the signature of the product manifold.

REFERENCES

Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, Michael Bronstein, Petar Veličković, and Pietro Liò. Sheaf neural networks with connection laplacians. In *Topological, Algebraic and Geometric Learning Workshops 2022*, pp. 28–36. PMLR, 2022a.

Federico Barbero, Cristian Bodnar, Haitz Sáez de Ocáriz Borde, and Pietro Liò. Sheaf attention networks. In *NeurIPS 2022 Workshop on Symmetry and Geometry in Neural Representations*, 2022b. URL https://openreview.net/forum?id=LIDvgVjpkZr.

Cristian Bodnar, Francesco Di Giovanni, Benjamin Paul Chamberlain, Pietro Liò, and Michael M Bronstein. Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. *arXiv preprint arXiv:2202.04579*, 2022.

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

Edoardo Cetin, Benjamin Chamberlain, Michael Bronstein, and Jonathan J Hunt. Hyperbolic deep reinforcement learning. *arXiv preprint arXiv:2210.01542*, 2022.

Boyuan Chen, Kuang Huang, Sunand Raghupathi, Ishaan Preetam Chandratreya, Qi Du, and Hod Lipson. Automated discovery of fundamental variables hidden in experimental data. *Nature Computational Science*, 2022.

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, 2020.

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

Yilun Du, Shuang Li, and Igor Mordatch. Compositional visual generation with energy based models. In *Neural Information Processing Systems*, 2020.

Yilun Du, Katherine Collins, Joshua B. Tenenbaum, and Vincent Sitzmann. Learning signal-agnostic manifolds of neural fields. In *Neural Information Processing Systems*, 2021.

Charles Fefferman, Sanjoy K. Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *arXiv: Statistics Theory*, 2013.

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

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. 2019.

Felix Hensel, Michael Moor, and Bastian Alexander Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, 2021.

Christopher John Isham. Modern differential geometry for physicists. 1989.

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.

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.

Ondrej Skopek, Octavian-Eugen Ganea, and Gary Bécigneul. Mixed-curvature variational autoencoders. *arXiv preprint arXiv:1911.08411*, 2019.

Haitz Sáez de Ocáriz Borde, Anees Kazi, Federico Barbero, and Pietro Liò. Latent graph inference using product manifolds, 2022. URL https://arxiv.org/abs/2211.16199.

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

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