Improving Hyperbolic Representations via Gromov-Wasserstein Regularization

Yifei Yang[1], Wonjun Lee[2], Dongmian Zou[3(✉)], and Gilad Lerman[2]

[1] Wuhan University, Wuhan 430072, Hubei, China
yfyang@whu.edu.cn
[2] University of Minnesota, Minneapolis, MN 55455, USA
{lee01273,lerman}@umn.edu
[3] Duke Kunshan University, Kunshan 215316, Jiangsu, China
dongmian.zou@duke.edu

Abstract. Hyperbolic representations have shown remarkable efficacy in modeling inherent hierarchies and complexities within data structures. Hyperbolic neural networks have been commonly applied for learning such representations from data, but they often fall short in preserving the geometric structures of the original feature spaces. In response to this challenge, our work applies the Gromov-Wasserstein (GW) distance as a novel regularization mechanism within hyperbolic neural networks. The GW distance quantifies how well the original data structure is maintained after embedding the data in a hyperbolic space. Specifically, we explicitly treat the layers of the hyperbolic neural networks as a transport map and calculate the GW distance accordingly. We validate that the GW distance computed based on a training set well approximates the GW distance of the underlying data distribution. Our approach demonstrates consistent enhancements over current state-of-the-art methods across various tasks, including few-shot image classification, as well as semi-supervised graph link prediction and node classification.

Keywords: Gromov-Wasserstein distance · Hyperbolic neural networks · Few-shot learning · Semi-supervised learning

1 Introduction

The success of many machine learning applications relies on effectively capturing the geometric intricacies of complex data structures. This challenge necessitates a careful selection of the underlying space that aligns with the inherent properties of data. In this context, hyperbolic spaces have emerged as a popular choice for modeling data with hierarchical structures such as lexical databases [23] and phylogenetic trees [28]. This is attributed to their distinctive capacity to

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-73007-8_13.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2025
A. Leonardis et al. (Eds.): ECCV 2024, LNCS 15140, pp. 211–227, 2025.
https://doi.org/10.1007/978-3-031-73007-8_13

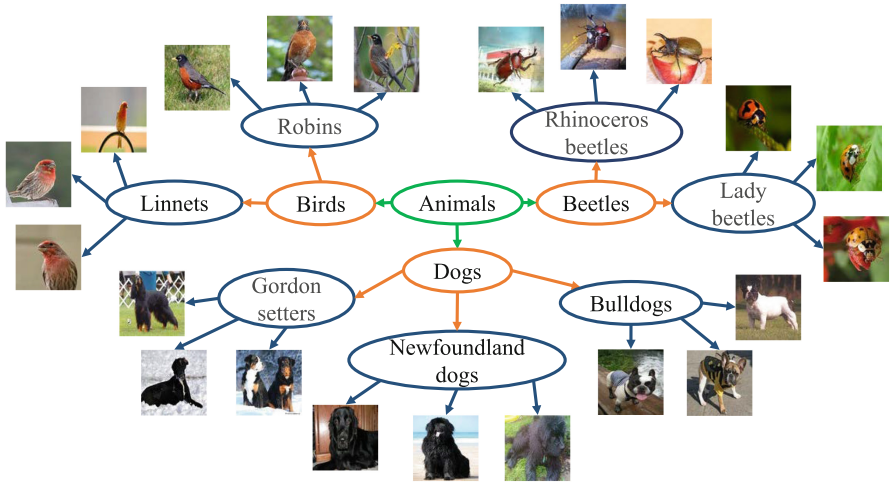

Fig. 1. Illustration of hierarchical structures in image data. The images are taken from the MiniImagenet dataset.

embody exponential growth and branching patterns, a feature rooted in their constant negative curvature. Such a geometric framework can reflect the tree-like architectures that are ubiquitous across numerous real-world networks.

Interestingly, hyperbolic spaces also play a significant role in computer vision [2,14], particularly in tasks like image classification and object recognition where hierarchical relationships among instances and concepts are common. For example, within the broader category of animals, there are dogs, which further subdivide into breeds such as bulldogs, Gordon setters, Newfoundland dogs, *etc.*. This is illustrated in Fig. 1. Hyperbolic embeddings can capture these hierarchies, thereby potentially improving classification performance and providing more interpretable models.

In crafting hyperbolic representations from non-hyperbolic features, two main strategies emerge: principled design and data-driven learning. The principled approach, exemplified by methods like Sarkar's algorithm [29], focuses on constructing embeddings that rigorously preserve the inherent geometric structure from data. This method particularly excels at maintaining hierarchical relationships within the data, ensuring that the embedded representation reflects structure of the original dataset with low distortion. On the other hand, the data-driven approach leverages the capabilities of end-to-end neural networks to learn the hyperbolic embedding directly from data. Such neural networks are well-known as hyperbolic neural networks (HNNs), which are pioneered in [10]. HNNs prioritize the development of hyperbolically coherent neural operations that are expressive and mirror the Euclidean counterparts. When dealing with input that is primarily Euclidean, HNNs employ straightforward operations, such as exponential maps, to transition features into the hyperbolic space, thereafter learning to extract deep features in a manner driven by the data. This method

adapts the embedding to specific tasks, potentially unveiling intricate patterns and relationships inherent within data.

In our work, we aim to establish a methodology that integrates the benefits of low-distortion embedding into data-driven learning. In the context of learning where generalization is the goal, we can view hyperbolic embedding as a method of embedding the data distribution into hyperbolic spaces. Therefore, we need to compare the distributions before and after the embedding. This necessitates a comparison of data distributions before and after embedding, which is complicated by the fundamental differences in geometric principles, such as curvature and dimensionality, that govern Euclidean and hyperbolic spaces. Direct comparisons between such disparate spaces are inherently complex and challenging.

To address this challenge, we rely on the Gromov-Wasserstein (GW) distance [19], a tool proven effective in comparing distributions across diverse metric spaces. Specifically, we employ the Gromov-Monge (GM) formulation, an alternative form of the GW distance, to compute the explicit transport map between heterogeneous metric spaces. The HNN layers induce the embedding distribution, thus serving as the transport map. Consequently, the GM formulation is a more suitable approach for the corresponding HNN layers. However, implementing the GM distance poses computational challenges due to its complex constraints. Therefore, in light of this, we extend the GM-based embedding technique introduced in [16] to compute the geometry-preserving transport map from data distribution in Euclidean space to hyperbolic spaces.

The contribution of the current work is summarized as follows:

1. We propose a novel methodology that incorporates the GM distance into an end-to-end learning framework for hyperbolic representation learning. This approach ensures a more faithful representation of the intrinsic hierarchical structures. Moreover, we verify that our approach maintains computational efficiency by analyzing its time complexity.
2. We validate that the embedding faithfully represents an embedding of distributions from the original domain by providing an upper bound for the sampling error. Therefore, regularization on the training sample generalizes to the underlying data distribution.
3. We provide empirical evidence showing that our regularization strategy leads to consistent improvements over existing baseline methods when applied to datasets with hierarchical structures, including both image and graph data.

2 Related Works

Hyperbolic Embedding. The adoption of hyperbolic geometries for representing hierarchical connections between data points has attracted considerable attention. The exploration began when researchers considered low-distortion embedding of tree or tree-like graphs into the hyperbolic domain. In their seminal work, Sarkar [29] showed that a tree can be embedded into the Poincaré disk with arbitrarily low distortion. Sala *et al.* [28] generalized the embedding to high dimensional Poincaré balls and discussed embedding general graphs into

trees. Sonthalia and Gilbert [32] bypassed the intermediary step of general graph structures, opting instead to directly infer tree architectures from the data itself. In addition to combinatorial methods, gradient-based methods are also used in hyperbolic embedding [23,24].

Hyperbolic Neural Networks. More recently, the advancement in hyperbolic representation learning has evolved to be more closely aligned with specific end tasks and specific datasets, such as text [34], images [14], biology [41], molecular [18], and knowledge graph [6]. This has led to the emergence of hyperbolic neural networks (HNNs) as a preferred framework for learning their representations. Numerous HNN architectures have emerged [3,5,10,11,18,24,31]. Comprehensive surveys on HNN include [26,40]. These HNNs define operations that conform with a choice of coordinate system in the hyperbolic space. However, when communicating between the Euclidean features and their hyperbolic embeddings, these HNNs take very simple approaches. For instance, Ganea *et al.* [10] directly used logarithmic and exponential maps at the origin, leading to unavoidable distortions. There are also very recent works addressing representations learned by HNNs. For instance, CO-SNE [12] reduces the dimensions of hyperbolic features for visualization, which is an analogy to t-SNE in the Euclidean domain. Parallel to this, Fan *et al.* [9] developed a systematic method for embedding data into nested hyperbolic spaces, which not only results in dimensionality reduction but also extends to design of HNNs. Nikolentzos *et al.* [25] applied the Weisfeiler-Leman algorithm to capture hierarchy in data and built HNNs accordingly. However, none of the above methods have considered comparing distributions before and after embedding into the hyperbolic space.

GW Distance. GW distance, a variant of optimal transport distance, has been widely utilized to quantify structural differences between different distributions. Memoli provided a detailed introduction and study of GW distance in [19,20]. Subsequently, Sturm [33] presented a more in-depth mathematical exposition of GW distance, focusing on its geodesic structure and gradient flow. Following these foundational works, GW distance has found applications in various fields, including computer vision [27,30], recommendation systems [17], natural language processing [1], generative models [4,16,22,35]. In particular, Lee *et al.* [16] employed the GM formulation to achieve geometry-preserving dimension reduction within a generative modeling framework. The discussion on the GM formulation can also be traced to [8,21]. Moreover, GW distance has been applied to specific data types such as graphs [38,39] and specific neural network architectures such as transformers [13]. To the best of our knowledge, GW distance has not yet been explored in the context of hyperbolic neural networks.

3 Methodology

3.1 Preliminaries of Hyperbolic Geometry

Hyperbolic geometry, characterized by its constant negative curvature, represents a non-Euclidean geometry essential for capturing complex hierarchical representations. We review operations that are defined in two different isometric coordinate systems, namely the Poincaré ball (Poincaré disk) model and the Lorentz (hyperboloid) model.

Poincaré Ball Model [10,31]**.** The Poincaré ball model for an n-dimensional hyperbolic space with curvature $-c$ is defined by

$$\mathbb{B}_c^n = \{\mathbf{x} \in \mathbb{R}^n : c\,\|\mathbf{x}\| < 1, c > 0\}\,, \tag{1}$$

with the corresponding Riemannian metric given by $\mathfrak{g}^{\mathbb{B}} = (\lambda_{\mathbf{x}}^c)^2 \boldsymbol{I}$, where $\lambda_{\mathbf{x}}^c = 2(1 - c\,\|\mathbf{x}\|^2)^{-1}$ is the conformal factor.

The analogy of a linear layer in HNN depends on two important operations, namely Möbius addition and multiplication. The Möbius addition $\oplus$ is defined as:

$$\mathbf{x} \oplus_c \mathbf{y} = \frac{\left(1 + 2c\,\langle \mathbf{x}, \mathbf{y}\rangle + c\,\|\mathbf{y}\|^2\right)\mathbf{x} + \left(1 - c\,\|\mathbf{x}\|^2\right)\mathbf{y}}{1 + 2c\,\langle \mathbf{x}, \mathbf{y}\rangle + c^2\,\|\mathbf{x}\|^2\,\|\mathbf{y}\|^2}. \tag{2}$$

The Möbius multiplication $\mathbf{M} \otimes_c \mathbf{x}$ is defined as:

$$\mathbf{M} \otimes_c \mathbf{x} = \frac{1}{\sqrt{c}} \tanh\left(\frac{\|\mathbf{M}\mathbf{x}\|}{\mathbf{M}\mathbf{x}} \tanh^{-1}(\sqrt{c}\,\|\mathbf{x}\|)\right) \frac{\mathbf{M}\mathbf{x}}{\|\mathbf{M}\mathbf{x}\|}. \tag{3}$$

Finally, the geodesic distance between two points $\mathbf{x}$ and $\mathbf{y}$ in the Poincaré ball can be expressed as:

$$d_{\mathbb{B}}^c(\mathbf{x}, \mathbf{y}) = \frac{2}{\sqrt{c}} \tanh^{-1}\left(\sqrt{c}\,\|-\mathbf{x} \oplus_c \mathbf{y}\|\right). \tag{4}$$

Unless otherwise specified, we follow the common choice to take $c = 1$.

Lorentz Model [5,6]**.** The Lorentz model of an n-dimensional hyperbolic space $\mathbb{L}^n$ is a manifold embedded in a $(n+1)$-dimensional Minkowski space. More specifically, this Minkowski space contains the same points as $\mathbb{R}^{n+1}$, but with an inner product $\langle \cdot, \cdot \rangle_{\mathbb{L}}$ defined by

$$\langle \mathbf{x}, \mathbf{y}\rangle_{\mathbb{L}} = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i, \mathbf{x} = (x_0, \cdots, x_n), \mathbf{y} = (y_0, \cdots, y_n) \in \mathbb{R}^{n+1}. \tag{5}$$

The Lorentz model $\mathbb{L}^n$ contains points with $\langle \mathbf{x}, \mathbf{x}\rangle_{\mathbb{L}} = -c$, where $-c$ is the curvature. That is,

$$\mathbb{L}^n = \left\{\mathbf{x} = (x_0, \cdots, x_n) \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{x}\rangle_{\mathbb{L}} = -c, x_0 > 0\right\}, \tag{6}$$

The geodesic distance between two points $\mathbf{x}$ and $\mathbf{y}$ in the Lorentz model $\mathbb{L}^n$ is expressed as

$$d_{\mathbb{L}}^c(\mathbf{x}, \mathbf{y}) = \frac{1}{\sqrt{c}} \cosh^{-1}\left(-\langle \mathbf{x}, \mathbf{y}\rangle_{\mathbb{L}}\right). \tag{7}$$

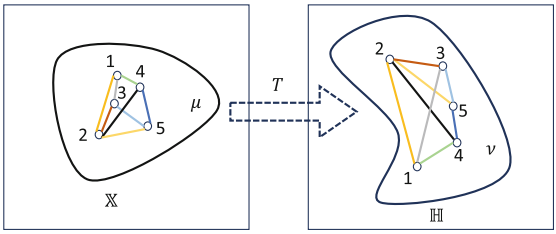

Fig. 2. Illustration of the transport map T. The feature distribution μ defined on $\mathbb{X}$ is pushed forward to $\nu = T_{\#}\mu$ on $\mathbb{H}$.

3.2 GW Regularization

To effectively utilize HNNs for learning from data, extending the geometry-preserving embedding technique introduced in [16], we employ a regularized learning approach focused on preserving the intrinsic geometric structures of the input features onto hyperbolic spaces. This involves employing a cost function capable of comparing two measures across distinct geometric spaces.

To bridge this gap, we consider the Gromov-Wasserstein (GW) distance [19] as it is adept at quantifying the similarity between probability distributions defined on distinct metric spaces. For two probability distributions μ and ν in a Euclidean feature space $\mathbb{X}$ and a hyperbolic space $\mathbb{H}$, respectively, the GW distance is defined as:

$$\mathrm{GW}\left(\mu,\nu\right) : := \min_{\pi\in\Pi(\mu,\nu)} \mathop{\mathbb{E}}_{((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}'))\sim\pi^2} \left[\left|c_{\mathbb{X}}(\mathbf{x},\mathbf{x}') - c_{\mathbb{H}}(\mathbf{y},\mathbf{y}')\right|^2\right] \tag{8}$$

where $c_{\mathbb{X}}$ and $c_{\mathbb{H}}$ represent the cost functions in the two spaces, and $\Pi(\mu,\nu)$ denotes the set of transport plans between μ and ν, which contains all joint distributions whose first and second marginals are given by μ and ν, respectively. By comparing the pairwise distances between two probability distributions to evaluate the similarity of geometric structures across different metric spaces.

When working with HNNs, an explicit map T from $\mathbb{X}$ to $\mathbb{H}$ is induced by the HNN layers, as illustrated in Fig. 2. The minimization problem presented in (8) can be reformulated as a Gromov-Monge (GM) distance minimization problem concerning a map T, defined as

$$\mathrm{GM}\left(\mu,\nu\right) := \min_{T_{\#}\mu=\nu} \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{x}')\sim\mu^2} \left[\left|c_{\mathbb{X}}(\mathbf{x},\mathbf{x}') - c_{\mathbb{H}}(T(\mathbf{x}),T(\mathbf{x}'))\right|^2\right]. \tag{9}$$

In an HNN, when embedding the probability distribution μ from a Euclidean space to a hyperbolic space, we can consider minimizing the following cost over $T \in \mathcal{H}$, where $\mathcal{H}$ is a hypothesis class determined by the HNN:

$$\mathrm{GM}(T;\mu) := \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{x}')\sim\mu^2} \left[\left|c_{\mathbb{X}}(\mathbf{x},\mathbf{x}') - c_{\mathbb{H}}(T(\mathbf{x}),T(\mathbf{x}'))\right|^2\right]. \tag{10}$$

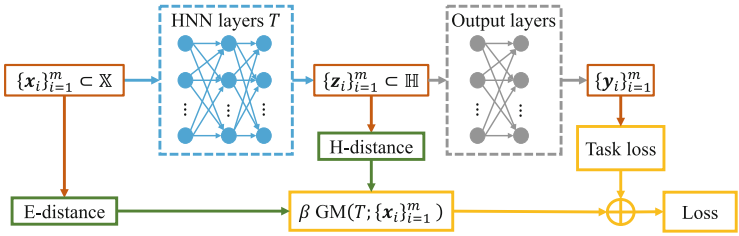

Fig. 3. The framework of our GW regularization.

Thus, we have $\min_{T\in\mathcal{H}} \mathrm{GM}(\mu, T_{\#}\mu) = \min_{T\in\mathcal{H}} \mathrm{GM}(T;\mu)$. While the GM problem described in (9) is highly challenging due to its pushforward constraint, minimizing the cost function in (10) is comparatively easier, as it does not require the pushforward constraint.

In practical implementation, we can employ the simple cost function $c_{\mathbb{X}}(\mathbf{x},\mathbf{x}') = \|\mathbf{x}-\mathbf{x}'\|^2$ or other cost functions such as $c_{\mathbb{X}}(\mathbf{x},\mathbf{x}') = \log\left(1+\|\mathbf{x}-\mathbf{x}'\|^2\right)$. Additionally, the explicit form of $c_{\mathbb{H}}$ can be derived from Eqs. (4) and (7): we can use $c_{\mathbb{H}} = d^c_{\mathbb{H}}$ or $c_{\mathbb{H}} = \log(1+d^c_{\mathbb{H}})$, where $\mathbb{H} = \mathbb{B}$ or $\mathbb{H} = \mathbb{L}$.

When we have data points $\{\mathbf{x}_i\}_{i=1}^m$ sampled independently and identically distributed (i.i.d.) from μ, we consider an empirical version of the GM distance, namely,

$$\mathrm{GM}(T;\{\mathbf{x}_i\}_{i=1}^m) = \frac{1}{m(m-1)} \sum_{i,j=1}^{m} \left|c_{\mathbb{X}}(\mathbf{x}_i,\mathbf{x}_j) - c_{\mathbb{H}}(T(\mathbf{x}_i),T(\mathbf{x}_j))\right|^2. \tag{11}$$

The overall framework of our GW regularization is shown in Fig. 3. Here, $\{\mathbf{x}_i\}_{i=1}^m$ are the Euclidean features that are fed into the HNN layers denoted by T, $\{\mathbf{z}_i\}_{i=1}^m$ are the hyperbolic representations so that $\mathbf{z}_i = T(\mathbf{x}_i)$ for each i. The GM distance, calculated according to (11), is used as a penalty term, multiplied with a hyperparameter β and then added to the loss function associated with the specific task.

Generalization Analysis. In the tasks we consider, the ability to generalize beyond the given data points is essential. Although (11) only provides a formulation based on the training data, we anticipate that it will serve as an effective approximation of (10), provided that the training set $\{\mathbf{x}_i\}_{i=1}^m$ adequately captures the characteristics of the underlying distribution μ. This expectation is encapsulated in the theorem we present below, which formalizes the relationship between the pointwise formulation and its distributional approximation under the condition of a representative sample.

Theorem 1. *Given cost functions $c_{\mathbb{X}}$ and $c_{\mathbb{H}}$, suppose that there exists a constant $0 < \alpha \leq 1$ for which T satisfies the following bi-Lipschitz condition:*

$$\alpha c_{\mathbb{X}}(\mathbf{x}, \mathbf{x}') \leq c_{\mathbb{H}}(T(\mathbf{x}), T(\mathbf{x}')) \leq \frac{1}{\alpha} c_{\mathbb{X}}(\mathbf{x}, \mathbf{x}'), \tag{12}$$

for any $\mathbf{x}, \mathbf{x}' \in \mathbb{X}$. Let μ be a distribution defined on $\mathbb{X}$ and $\{\mathbf{x}_i\}_{i=1}^m$ be i.i.d. sampled from μ. Then,

$$|\mathrm{GM}(T; \{\mathbf{x}_i\}_{i=1}^m) - \mathrm{GM}(T; \mu)| \leq C \frac{(1/\alpha - 1)^2}{R^2} \tag{13}$$

holds with probability at least $1 - 2\exp\left(-\frac{mC}{8R^4}\right)$ where C is a constant depending on μ and $R := \max_{\mathbf{x}, \mathbf{x}' \in \mathrm{supp}(\mu)} c_{\mathbb{X}}(\mathbf{x}, \mathbf{x}')$.

The proof of Theorem 1 is presented in the appendix.

Complexity Analysis. Since the time complexity for calculating each Euclidean distance is $O(n)$ and the time complexity for calculating each hyperbolic distance (in either the Poincaré ball model or the Lorentz model) is $O(n)$, the time complexity for calculating the GW regularization term is $O(nm^2)$.

Note that, if we only consider linear layers in an L-layer Vanilla HNN, the time complexity is $O(mn^2L)$. In the few-shot learning task and the semi-supervised graph node-level tasks that we consider in this paper, the number of data points m is usually smaller than or comparable with n. Moreover, in tasks such as link prediction, one also needs to compare pairs of data points, leading to a time complexity of $O(mn^2L)$ in computing the task loss. Therefore, adding the regularization term will not lead to change of the order of time complexity.

4 Experiments

To validate the effectiveness of the GW regularization method, we conduct experiments on both image datasets and graph datasets. We present results on few-shot image classification in Sect. 4.1 and results on graph node classification and link prediction in Sect. 4.2. All the presented experiments are implemented on a server with RTX 4090 (24 GB) GPUs, where each implementation runs on a single GPU. The implementation of our proposed methods can be accessed at https://github.com/yyf1217/GW-Regularization.

4.1 Few-Shot Image Classification

We consider the important task of few-shot image classification where the goal is to recognize new categories with very few labeled examples per class. Unlike supervised image classification, this task aims to generalize from a small number of examples, typically one to five images per class. Learning structures of images is crucial since there is very limited labeled data.

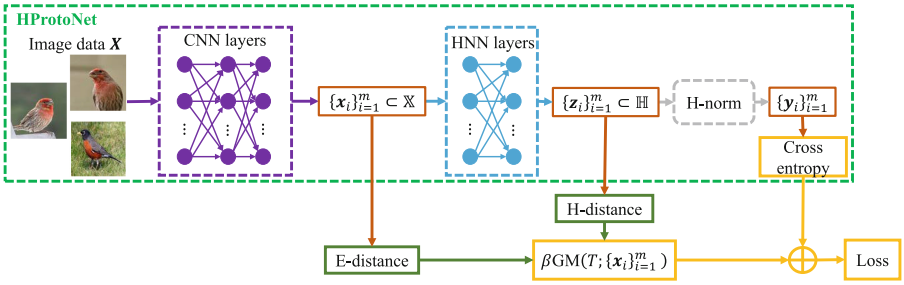

Fig. 4. The framework of the GW regularized hyperbolic ProtoNet model.

Datasets. We consider two widely used image datasets, namely *MiniImageNet* [36] and *Caltech-UCSD Birds-200-2011 (CUB)* [37]. These datasets have been used in [14], where the hyperbolic ProtoNet model excels Euclidean baselines. The MiniImageNet dataset is a subset of the ImageNet dataset [7]. It comprises images from 100 classes, with 600 images per class. The classes have a 64/16/20 split for training/validation/test. The CUB dataset consists of 200 categories of bird images, totaling 11,788 pictures. In this dataset, the training/validation/test split is 100/50/50.

Setting. We apply our GW regularization on the hyperbolic ProtoNet model [14], which primarily consists of convolutional neural network (CNN) layers used for feature extraction from Euclidean image data and subsequent HNN layers for classification. In [14], the HNN layers are taken to be a simple exponential map from the Euclidean space to the Poincaré ball. In our implementation, we utilize the publicly available code for the hyperbolic ProtoNet model from https://github.com/leymir/hyperbolic-image-embeddings.

For clarity, we illustrate the framework for this task in Fig. 4, which is based on Fig. 3. We take the features $\{\mathbf{x}_i\}_{i=1}^m$ from the output of the CNN layers, and then the HNN layers process them into $\{\mathbf{z}_i\}_{i=1}^m$. Following this, the hyperbolic distance between each hyperbolic embedding $\mathbf{z}_i$ and the origin of the Poincaré ball is computed according to (4), which is then used to calculate the maximum class probability. Finally, the cross-entropy loss is computed using the maximum class probability and the labels. We add $\beta\,\mathrm{GM}(T; \{\mathbf{x}_i\}_{i=1}^m)$ to the cross-entropy loss to formulate our regularized loss function.

We consider various CNN architectures including Conv4, ResNet10, ResNet12, and ResNet18. The HNN layers within the model incorporate linear layers from [10]. Additionally, ReLU is used as the activation function across the model. For Conv4, the embedding dimension is set to 1,600, while for ResNet10, ResNet12, and ResNet18, the embedding dimension is set to 512. The initial learning rate is 0.001. We consider both the 1-Shot 5-Way and 5-Shot 5-Way settings, which means there are five classes involved for the task and only one/five examples per class for training. In all experiments for the 1-Shot 5-Way task, a consistent curvature parameter of -0.08 is employed. Similarly, for the 5-Shot

5-Way task, we uniformly set the curvature to -0.01 throughout all experiments. Both curvatures agree with the recommened settings in the original hyperbolic ProtoNet. The hyperparameter β is selected according to the validation set. We report the initial learning rate as well as the best β in Table 1.

Table 1. Hyperparameters used in few-shot image classification. "ResNet" abbreviated as "Res" due to space constraints.

		Initial learning rate (lr)				β			
		Conv4	Res10	Res12	Res18	Conv4	Res10	Res12	Res18
MiniImageNet	1-Shot 5-Way	0.0005	0.001	0.001	0.001	0.6	0.5	0.1	0.7
	5-Shot 5-Way	0.01	0.001	0.001	0.001	14	0.1	0.5	0.5
CUB	1-Shot 5-Way	0.001	0.001	0.001	0.001	0.02	0.015	0.15	0.1
	5-Shot 5-Way	0.01	0.001	0.001	0.001	0.07	0.01	0.25	0.1

Results. We present our numerical results for both the 1-Shot 5-Way and 5-Shot 5-Way settings in Table 2. We randomly sample 10,000 data points from the test set to evaluate the performance of the model and report 95% confidence intervals for all results. Each entry reports the average classification accuracy, including means and standard deviations.

Table 2. Accuracy (%) results for few-shot image classification.

	MiniImageNet		CUB	
	1-Shot 5-Way	5-Shot 5-Way	1-Shot 5-Way	5-Shot 5-Way
Conv4	53.77 ± 0.20	71.33 ± 0.16	64.66 ± 0.23	80.29 ± 0.16
Conv4+GW	55.57 ± 0.22	73.04 ± 0.16	68.67 ± 0.23	80.54 ± 0.16
ResNet10	56.45 ± 0.21	65.39 ± 0.17	72.01 ± 0.22	80.75 ± 0.15
ResNet10+GW	57.50 ± 0.22	67.62 ± 0.17	72.67 ± 0.22	83.73 ± 0.14
ResNet12	55.96 ± 0.22	70.32 ± 0.17	75.77 ± 0.22	82.90 ± 0.15
ResNet12+GW	56.95 ± 0.22	73.04 ± 0.16	77.24 ± 0.21	85.66 ± 0.14
ResNet18	57.04 ± 0.22	68.43 ± 0.16	71.86 ± 0.22	85.31 ± 0.13
ResNet18+GW	58.51 ± 0.22	71.64 ± 0.16	72.64 ± 0.22	85.83 ± 0.13

From the results, we observe that our GW regularization consistently boosts the accuracy of the model. This improvement is evident in both datasets and both scenarios, regardless of the underlying CNN architecture. For most CNN architectures, the improvement is more evident in the 5-Shot 5-Way scenario than in the 1-Shot 5-Way scenario. We believe that this can be attributed to the increased number of training instances available in the 5-Shot scenario, which provides a richer structure that GW regularization can leverage.

Runtime. We report the average run time for training one epoch in Table 3. As anticipated, incorporating GW regularization results in extended epoch training times. However, the increase in runtime is not significant, aligning with the time complexity analysis in Sect. 3.2. This observation underscores the efficiency of our GW regularization approach, ensuring that the added computational demand does not impose a significant burden on the overall training process.

Table 3. Average runtime (in seconds) for training one epoch in few-shot image classification.

	MiniImageNet		CUB	
	1-Shot 5-Way	5-Shot 5-Way	1-Shot 5-Way	5-Shot 5-Way
Conv4	31.2	25.2	32.4	32.9
Conv4+GW	33.6	26.4	35.1	32.9
ResNet10	37.2	28.8	38.4	34.9
ResNet10+GW	40.8	34.8	43.2	38.4
ResNet12	42.0	36.0	46.8	42.0
ResNet12+GW	45.6	40.8	50.4	45.6
ResNet18	55.2	43.2	60.0	52.8
ResNet18+GW	57.6	49.2	62.4	57.6

4.2 Semi-supervised Link Prediction and Node Classification

We consider graph link prediction and node classification, which are widely benchmarked semi-supervised tasks in graph deep learning literature.

Datasets. We consider nine datasets for node classification, namely *Cora, Disease, Airport, Cornell, Texas, Wisconsin, Chameleon, Squirrel, and Actor*. For the first three datasets, namely Disease, Airport, and Cora, we also perform link prediction. We briefly introduce the datasets in the appendix and summarize the statistics of the graphs in Table 4.

Table 4. Statistics of graph datasets.

	Disease-LP	Disease-NC	Airport	Cora	Cornell	Texas	Wisconsin	Chameleon	Squirrel	Actor
# nodes	2,665	1,044	3,188	2,708	183	183	251	2,277	5,201	7,600
# edges	2,265	2,265	15,837	4,488	280	295	466	31,421	198,493	26,752
# features	11	1,000	11	1,433	1,703	1,703	1,703	2,325	2,089	931
# classes	N/A	2	4	7	5	5	5	5	5	5

In link prediction tasks, we randomly split edges into 85%/5%/10% for training, validation, and test sets. For node classification tasks, we use a

70%/15%/15% splits for Airport, Cornell, Texas, Wisconsin, Chameleon, Squirrel, and Actor, we use 30%/10%/60% splits for Disease, and we use standard splits with 20 train examples per class for Cora. The above split settings agree with standard settings in the baseline papers.

Setting. For the graph datasets, we consider the following three HNNs: the vanilla HNN [10], HGCN [5], and HyboNet [6]. We utilize the publicly available code for HNN and HGCN from https://github.com/HazyResearch/hgcn and HyboNet from https://github.com/chenweize1998/fully-hyperbolic-nn.

For each model, we apply GW distance to the input node features and their hyperbolic representations after applying the Euclidean-to-hyperbolic operation. The regularization framework is the same as Fig. 3.

In these experiments, the HNN models share a common architecture consisting of HNN layers and output layers. These models employ different HNN layers to extract features from graph data and generate hyperbolic embeddings $\{\mathbf{z}_i\}_{i=1}^m$. Specifically, the Vanilla HNN and HGCN employ the Poincaré model to build linear layers and activation. HGCN also has aggregation operation, facilitating message passing. Differently, HyboNet employs the linear layers and aggregation in the Lorentz model. The output layers are used to accomplish various tasks. For node classification, the obtained hyperbolic embeddings are directly projected to the Euclidean space. A linear layer is then used to predict the class probabilities, with which the negative log-likelihood loss is computed. For link prediction, the hyperbolic distances between the hyperbolic embeddings are computed. Subsequently, the Fermi-Dirac algorithm [15] is employed to transform the hyperbolic distances into the edge probabilities. Finally, cross-entropy losses are computed using these edge probabilities and the true edge labels.

For simplicity, the curvature of the hyperbolic space is set as -1 for all experiments, which is a common setting for graph datasets. Tables 5 and 6 list hyperparameters such as initial learning rate, the number of the HNN layers, weight decay, dropout, and embedding dimensions. The hyperparameter β from validation is also listed for different datasets.

Table 5. Hyperparameters used in the Vanilla HNN and HGCN.

	Disease		Airport		Cora		Cornell	Texas	Wisconsin	Chameleon	Squirrel	Actor
	LP	NC	LP	NC	LP	NC	NC	NC	NC	NC	NC	NC
Initial lr	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
# layers	2	2	2	2	2	2	2	2	2	2	2	2
Weight decay	0.0	0.001	0.0005	0.0	0.005	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Dropout	0.0	0.1	0.0	0.0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
# embeddings	3	16	16	16	16	16	16	16	16	16	5	5
HNN β	0.9	0.08	2	1.25	0.5	0.35	0.25	0.1	0.15	0.02	0.001	0.03
HGCN β	0.3	0.09	2	1.5	0.1	0.6	0.25	0.15	0.2	0.08	0.01	0.03

Table 6. Hyperparameters used in HyboNet.

	Disease		Airport		Cora		Cornell	Texas	Wisconsin	Chameleon	Squirrel	Actor
	LP	NC	LP	NC	LP	NC	NC	NC	NC	NC	NC	NC
Initial lr	0.005	0.005	0.01	0.02	0.02	0.02	0.005	0.005	0.005	0.005	0.01	0.01
# layers	2	4	2	6	2	3	2	2	2	2	2	2
Weight decay	0.0	0.0	0.0	0.0001	0.001	0.01	0.0	0.0	0.0	0.0	0.001	0.001
Dropout	0.0	0.1	0.0	0.0	0.7	0.9	0.2	0.2	0.2	0.2	0.2	0.2
# embeddings	16	16	16	16	16	16	16	16	16	16	16	16
β	0.3	0.07	1.5	1.1	0.25	0.25	0.13	0.1	0.1	1.3	2.5	0.01

Results. For the link prediction task, we report the AUC scores in Table 7. For the node classification task, we report the F1 scores, in Table 8. Each score reports the average from three random runs as well as the standard deviation. For clarity, we just use "HNN" in place of the Vanilla HNN in the tables.

Table 7. AUC (%) results of the link prediction task.

	Disease	Airport	Cora
HNN	92.83 ± 1.83	93.56 ± 0.30	88.80 ± 1.29
HNN+GW	94.45 ± 0.58	94.23 ± 0.33	89.76 ± 2.07
HGCN	92.41 ± 1.78	93.42 ± 0.15	93.37 ± 0.15
HGCN+GW	93.79 ± 1.15	95.81 ± 0.02	93.48 ± 0.22
HYBONET	95.65 ± 0.57	96.18 ± 0.05	92.04 ± 0.37
HYBONET+GW	96.58 ± 0.57	96.44 ± 0.02	93.33 ± 0.22

From Table 7, it is evident that our GW regularization consistently enhances the AUC scores across different network architectures and across all datasets. Similarly, Table 8 showcases that GW regularization again demonstrates a positive impact on model performance across various datasets and architectures. In many scenarios of node classification, GW regularization actually achieves substantial improvement, such as HNN+GW for Disease, HGCN+GW for Cornell and HyboNet+GW for Wisconsin.

Runtime. We report the average run time for training one epoch for link prediction in Table 9. Similarly to the previous experiment, the runtime for GW regularized training is longer, but not significantly. This again aligns with the complexity analysis in Sect. 3.2.

Table 8. F1 (%) result of the node classification task.

	Disease	Airport	Cora	Chameleon	Squirrel
HNN	58.40 ± 2.53	87.40 ± 1.34	53.73 ± 0.91	71.43 ± 1.63	47.89 ± 2.21
HNN+GW	65.62 ± 1.98	89.38 ± 0.67	55.33 ± 0.35	72.53 ± 0.19	49.44 ± 1.05
HGCN	93.04 ± 1.64	86.96 ± 0.22	79.47 ± 1.01	77.72 ± 0.90	51.79 ± 0.58
HGCN+GW	95.28 ± 0.40	87.91 ± 0.72	80.00 ± 0.62	79.18 ± 1.64	52.70 ± 1.24
HYBONET	85.70 ± 0.60	93.13 ± 0.19	73.53 ± 0.75	82.05 ± 0.55	74.04 ± 2.42
HYBONET+GW	87.66 ± 1.21	94.46 ± 0.20	79.40 ± 0.95	83.58 ± 0.28	77.27 ± 0.78

		Cornell	Texas	Wisconsin	Actor
HNN		93.94 ± 1.31	94.69 ± 1.31	93.21 ± 3.86	42.63 ± 1.73
HNN+GW		95.45 ± 2.28	96.97 ± 1.32	98.15 ± 1.85	45.03 ± 1.73
HGCN		83.33 ± 3.47	78.79 ± 4.73	75.31 ± 2.83	36.39 ± 1.00
HGCN+GW		88.64 ± 2.28	81.82 ± 3.94	83.33 ± 1.86	39.19 ± 0.67
HYBONET		78.79 ± 4.73	67.42 ± 3.47	74.07 ± 1.86	45.74 ± 2.13
HYBONET+GW		81.82 ± 3.93	74.24 ± 1.31	80.87 ± 3.86	49.64 ± 0.38

Table 9. Average runtime (in seconds) for training one epoch in link prediction.

	Disease	Airport	Cora
HNN	0.0301	0.0881	0.0549
HNN+GW	0.0313	0.0884	0.0563
HGCN	0.0489	0.1075	0.0743
HGCN+GW	0.0505	0.1078	0.0747
HYBONET	0.0301	0.0631	0.0374
HYBONET+GW	0.0323	0.0639	0.0379

5 Conclusion

In this paper, we have delved into the integration of the GW distance as a novel regularization term within the realms of hyperbolic neural networks, with a particular emphasis on leveraging the GM formulation. This approach has allowed for a sophisticated comparison of distributions across the Euclidean space and the hyperbolic space, capturing the essence of the underlying structures while not increasing the order of the time complexity. We have demonstrated that the hyperbolic embeddings of the data points resonate closely with distributions where they are sampled from.

Our regularization method has been rigorously tested and validated across diverse datasets including image data for the few-shot classification task and

graph data for the semi-supervised link prediction and node classification tasks, showcasing a uniform elevation in performance metrics.

Our future work involves more efficient methods in large scale settings, where the number of instances is much larger than feature dimensions. Additionally, we are intrigued by the potential of integrating our GW regularization framework with emerging geometric deep learning architectures for complex data structures, where the flexibility with the choice of underlying spaces is crucial.

Acknowledgement. YY and DZ acknowledge funding from National Natural Science Foundation of China (NSFC) under award number 12301117, WL acknowledges funding from the National Institute of Standards and Technology (NIST) under award number 70NANB22H021, and GL acknowledges funding from NSF award DMS 2124913.

References

1. Alvarez-Melis, D., Jaakkola, T.S.: Gromov-Wasserstein alignment of word embedding spaces. arXiv preprint arXiv:1809.00013 (2018)
2. Atigh, M.G., Schoep, J., Acar, E., van Noord, N., Mettes, P.: Hyperbolic image segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2022)
3. Bachmann, G., Bécigneul, G., Ganea, O.: Constant curvature graph convolutional networks. In: International Conference on Machine Learning, pp. 486–496. PMLR (2020)
4. Bunne, C., Alvarez-Melis, D., Krause, A., Jegelka, S.: Learning generative models across incomparable spaces. In: International Conference on Machine Learning, pp. 851–861. PMLR (2019)
5. Chami, I., Ying, Z., Ré, C., Leskovec, J.: Hyperbolic graph convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 32, pp. 4868–4879 (2019)
6. Chen, W., et al.: Fully hyperbolic neural networks. In: Annual Meeting of the Association for Computational Linguistics, pp. 5672–5686 (2022)
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE (2009)
8. Dumont, T., Lacombe, T., Vialard, F.X.: On the existence of Monge maps for the Gromov-Wasserstein problem (2022)
9. Fan, X., Yang, C.H., Vemuri, B.C.: Nested hyperbolic spaces for dimensionality reduction and hyperbolic NN design. In: IEEE Conference on Computer Vision and Pattern Recognition (2022)
10. Ganea, O., Bécigneul, G., Hofmann, T.: Hyperbolic neural networks. In: Advances in Neural Information Processing Systems, vol. 31, pp. 5345–5355 (2018)
11. Gulcehre, C., et al.: Hyperbolic attention networks. In: International Conference on Learning Representations (2019). https://openreview.net/forum?id=rJxHsjRqFQ
12. Guo, Y., Guo, H., Yu, S.X.: Co-SNE: dimensionality reduction and visualization for hyperbolic data. In: IEEE Conference on Computer Vision and Pattern Recognition (2022)
13. Huang, Y., Zhang, T., Zhu, H.: Improving word alignment by adding Gromov-Wasserstein into attention neural network. In: Journal of Physics: Conference Series, vol. 2171, p. 012043. IOP Publishing (2022)

14. Khrulkov, V., Mirvakhabova, L., Ustinova, E., Oseledets, I., Lempitsky, V.: Hyperbolic image embeddings. In: IEEE Conference on Computer Vision and Pattern Recognition (2020)
15. Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., Boguná, M.: Hyperbolic geometry of complex networks. Phys. Rev. E **82**(3), 036106 (2010)
16. Lee, W., Yang, Y., Zou, D., Lerman, G.: Monotone generative modeling via a Gromov-Monge embedding. arXiv e-prints (2023)
17. Li, X., et al.: Gromov-Wasserstein guided representation learning for cross-domain recommendation. In: ACM International Conference on Information & Knowledge Management, pp. 1199–1208 (2022)
18. Liu, Q., Nickel, M., Kiela, D.: Hyperbolic graph neural networks. In: Advances in Neural Information Processing Systems, vol. 32, pp. 8230–8241 (2019)
19. Mémoli, F.: On the use of Gromov-Hausdorff distances for shape comparison. In: PBG@Eurographics (2007)
20. Mémoli, F.: Gromov-Wasserstein distances and the metric approach to object matching. Found. Comput. Math. **11**, 417–487 (2011)
21. Mémoli, F., Needham, T.: Distance distributions and inverse problems for metric measure spaces. Stud. Appl. Math. **149**(4), 943–1001 (2022)
22. Nakagawa, N., Togo, R., Ogawa, T., Haseyama, M.: Gromov-Wasserstein autoencoders. In: International Conference on Learning Representations (2023). https://openreview.net/forum?id=sbS10BCtc7
23. Nickel, M., Kiela, D.: Poincaré embeddings for learning hierarchical representations. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
24. Nickel, M., Kiela, D.: Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In: International Conference on Machine Learning (2018)
25. Nikolentzos, G., Chatzianastasis, M., Vazirgiannis, M.: Weisfeiler and Leman go hyperbolic: learning distance preserving node representations. In: International Conference on Artificial Intelligence and Statistics, pp. 1037–1054 (2023)
26. Peng, W., Varanka, T., Mostafa, A., Shi, H., Zhao, G.: Hyperbolic deep neural networks: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **44**(12), 10023–10044 (2021)
27. Peyré, G., Cuturi, M., Solomon, J.: Gromov-Wasserstein averaging of kernel and distance matrices. In: International Conference on Machine Learning, pp. 2664–2672. PMLR (2016)
28. Sala, F., De Sa, C., Gu, A., Ré, C.: Representation tradeoffs for hyperbolic embeddings. In: International Conference on Machine Learning (2018)
29. Sarkar, R.: Low distortion delaunay embedding of trees in hyperbolic plane. In: van Kreveld, M., Speckmann, B. (eds.) GD 2011. LNCS, vol. 7034, pp. 355–366. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-25878-7_34
30. Schmitzer, B., Schnörr, C.: Modelling convex shape priors and matching based on the Gromov-Wasserstein distance. J. Math. Imaging Vis. **46**, 143–159 (2013)
31. Shimizu, R., Mukuta, Y., Harada, T.: Hyperbolic neural networks++. In: International Conference on Learning Representations (2021)
32. Sonthalia, R., Gilbert, A.: Tree! I am no tree! I am a low dimensional hyperbolic embedding. In: Advances in Neural Information Processing Systems, vol. 33, pp. 845–856 (2020)
33. Sturm, K.T.: The Space of Spaces: Curvature Bounds and Gradient Flows on the Space of Metric Measure Spaces, vol. 290. American Mathematical Society (2023)
34. Tifrea, A., Bécigneul, G., Ganea, O.E.: Poincaré glove: hyperbolic word embeddings. In: International Conference on Learning Representations (2018)

35. Titouan, V., Flamary, R., Courty, N., Tavenard, R., Chapel, L.: Sliced Gromov-Wasserstein. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
36. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in Neural Information Processing Systems, vol. 29 (2016)
37. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD birds-200-2011 dataset (2011)
38. Xu, H., Luo, D., Carin, L.: Scalable Gromov-Wasserstein learning for graph partitioning and matching. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
39. Xu, H., Luo, D., Zha, H., Carin, L.: Gromov-Wasserstein learning for graph matching and node embedding. In: International Conference on Machine Learning, pp. 6932–6941. PMLR (2019)
40. Yang, M., et al.: Hyperbolic graph neural networks: a review of methods and applications (2022)
41. Zhou, Y., Sharpee, T.O.: Hyperbolic geometry of gene expression. Iscience **24**(3), 102225 (2021)