Pattern Recognition 158 (2025) 111042

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Hyperbolic prototype rectification for few-shot 3D point cloud classification

Yuan-Zhi Feng [a,1], Shing-Ho J. Lin [b,1], Xuan Tang [c], Mu-Yu Wang [a], Jian-Zhang Zheng [b], Zi-Yao He [a], Zi-Yi Pang [a], Jian Yang [d], Ming-Song Chen [a], Xian Wei [a,*]

[a] Software Engineering Institute, East China Normal University, 200062, Shanghai, China
[b] University of Chinese Academy of Sciences, 100049, Beijing, China
[c] School of Communication and Electronic Engineering, East China Normal University, 200062, Shanghai, China
[d] School of Geospatial Information, Information Engineering University, Zhengzhou, 450052, Henan, China

ARTICLE INFO

Dataset link: https://github.com/Jonathan-UCAS/HPR

Keywords:
Hyperbolic geometry
Few-shot learning
Point cloud classification
Prototype rectification
Feature enhancement

ABSTRACT

Few-shot point cloud classification is a challenging task in 3D computer vision and has received widespread attention from researchers. Most of the works on deep learning models rely heavily on Euclidean spatial metrics. However, point cloud objects often have complex non-Euclidean geometric structures, with underlying inter/intra-class hierarchical structures, which are difficult to capture by current Euclidean-based deep learning models. Moreover, due to the lack of training samples, many few-shot learning methods often suffer from the overfitting problem. Given the Hyperbolic metric of non-Euclidean geometry offering hierarchical structural prior, as we assume to be able to assist FSL task, we propose Hyperbolic Prototype Rectification (**HPR**) for few-shot point cloud classification, without requiring extra learnable parameter. Firstly, point clouds are embedded into hyperbolic space to better describe hierarchical similarity relationships in data. Secondly, the **HPR** utilizes hyperbolic spatial and distributional information to enhance the feature representation and improve the generalization capability, with more appropriate hyperbolic prototypes. The few-shot classification experiments and further ablation studies conducted on widely used point cloud datasets demonstrate the effectiveness of our method. On the real-world ScanObjectNN(-PB) datasets, the average classification accuracy outperforms the SOTA method by 2.08%(0.66%), respectively, indicating that the proposed **HPR** has great generalization capability and strong robustness against perturbed data. Our code is available at: https://github.com/Jonathan-UCAS/HPR.

1. Introduction

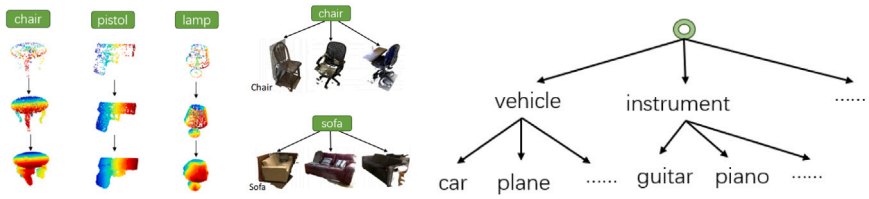
Few-shot 3D point cloud classification has received increasing attention in recent years. The community has witnessed the rapid development of 3D sensing technology, including 3D scanners, LiDARs and stereo reconstruction, as well as the wide application of classification task in robotics [1], autonomous driving [2], shape analysis [3], remote sensing [4,5], etc. However, the outstanding performance of deep learning-based methods [4] relies heavily on a large amount of labeled data. Despite the various ways to obtain 3D point cloud data, the process of obtaining annotated data is cumbersome and time-consuming. Deep learning models tend to be overfitting with limited generalization ability when encountering insufficient annotated samples. The sample scarcity issue highlights the importance of developing few-shot learning (FSL) [6] algorithms.

With previous success on 2D image [7,8], FSL can be introduced for 3D point cloud tasks. It aims to enhance the generalization ability of the deep learning models and enable the models to perform well even for novel classes with a few labeled samples only. Among different approaches, metric-based FSL methods have been widely used in existing studies because of their simplicity and effectiveness. Usually, the samples are firstly embedded into a feature space, then unseen query samples are classified by the nearest prototype obtained from the network. As a representative, ProtoNet [7] simply utilized the average of support samples of each class as prototypes and classified by Euclidean distance of query samples and prototypes. While being efficient, it is possibly biased due to the scarcity of support samples, thus unable to accurately represent the corresponding class. To address this issue, several works [8–10] have incorporated spatial relationships and distributional information of query samples or used multiple prototypes to obtain more accurate prototypes.

However, different from 2D images, 3D point cloud objects admit complex non-Euclidean structures, with behaviors that are dynamic,

* Corresponding author.
E-mail address: xwei@sei.ecnu.edu.cn (X. Wei).
[1] Equal Contribution.

https://doi.org/10.1016/j.patcog.2024.111042
Received 31 October 2023; Received in revised form 26 August 2024; Accepted 20 September 2024
Available online 26 September 2024



(a) Density (b) Angle (c) Semantic

Fig. 1. (a), (b) intra-class hierarchy, and (c) semantic inter-class hierarchy.

irregular, and unpredictable. In addition, 3D point cloud data generally contains position and shape information, with multiple levels of explicit or implicit hierarchy, including the part-whole hierarchy of objects [11,12], intra-class hierarchy in Fig. 1(a), (b), and class semantic hierarchy in Fig. 1(c). While there have been a series of recent works [13–15] for few-shot point cloud classification, they primarily focus on constructing models in Euclidean space, being unable to sufficiently obtain the geometric properties of point clouds, particularly the inter/intra-object hierarchical relationships within and between point clouds. Also, some works [14,15] require assistance from the image, increasing computational cost. Some of them [16] are also sensitive to noises.

FSL requires appropriate prototypes even with scarce samples, where spatial geometric priors and structural information become powerful assistance. The hierarchy of 3D point cloud is very useful structural information and has been studied in many approaches [11, 17]. Existing works on point cloud FSL did not consider such hierarchy and are mostly based on Euclidean space, which lacks such representation ability. Previous work [18] suggested that the hyperbolic space can embed tree-like hierarchical data with lower distortion than in Euclidean space. Based on this advantage in geometric priors, recent works [12,16] have explored the part-whole compositional structure in hyperbolic space.

Given that point cloud data exhibits a tree-like hierarchy, and the huge dependence on accurate data embedding for FSL as it faces sample scarcity, we assume that incorporating hyperbolic metrics could benefit the 3D point cloud FSL. It leads to a more accurate feature space, thus improving the representation ability for better performance.

In this work, we aim to ① validate the effectiveness of hyperbolic embeddings for 3D point cloud FSL via experiments, ② propose a framework using spatial relationship and distributional information in hyperbolic space for prototype rectification with comprehensive experiments to validate its effectiveness, and ③ discuss the future developments and applications. In summary, our main contributions are as follows:

- We firstly systematically construct metric learning in hyperbolic space for few-shot point cloud classification, considering the hierarchy structure of point clouds. Hyperbolic embeddings bring up to 1.66% accuracy improvement for 3D point cloud few-shot learning from our experiments.
- We proposed Hyperbolic Prototype Rectification (**HPR**) with two variants, the **HPR-D** based on spatial relationship, and the **HPR-WN** based on distributional information. They both improve the robustness of perturbations, obtaining more effective prototypes without requiring extra training parameters. Further geometrical analysis of our method is also performed.
- We outperform existing methods on public datasets. Specifically, for real-world performance tested on ScanObjectNN, we achieve state-of-the-art results among all methods. Several experiments verify the performance and robustness of **HPR**.

2. Related work

In this section, we briefly review the previous representative works on 3D point cloud classification, few-shot learning, and hyperbolic embedding.

2.1. Deep learning for 3D point cloud classification

3D point cloud classification is an important task with wide application scenarios. There are several feature embedding manners.

For ***projection-based methods*** onto image modal, MVCNN [19] is the early representative work using multi-view information. View-GCN [20] follows the same manner using graph convolution. Simple-View [21] proposes an efficient view projection module and was widely adopted in related tasks [14].

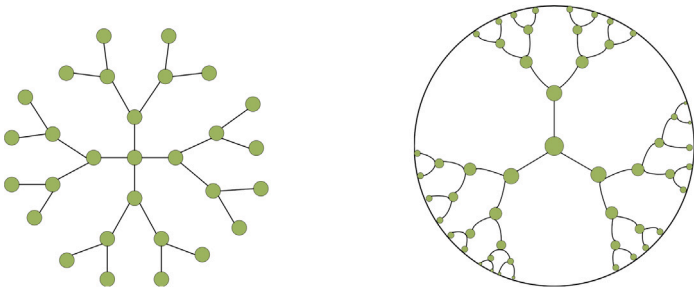
For ***point-based methods*** using raw points only, PointNet [22] extracts point-wise features with multi-layer perceptron (MLP) and uses max-pooling operations to aggregate global permutation invariant features. PointNet++ [23] considers local structures explicitly and introduces a hierarchical aggregation paradigm for point cloud by using local–global geometric information. PointCNN [24] uses $\mathcal{X}$-Conv layer replacing MLP layer to exploit certain canonical ordering of points, preserving equivariance. DGCNN [25] proposes EdgeConv, capturing local geometric information by constructing local dynamic graphs between base points and neighboring nodes in a specific region and extracting permutation invariant features. KPConv [26] proposes convolution on point cloud using similar ideas of image convolution. PointMLP [27] proposes a full MLP architecture with a geometric affine module. PCT [28] is a transformer-based method with the attention mechanism inspired by EdgeConv.

For comparison, projection-based methods benefit largely from image feature embedding, while point-based methods directly utilize the properties of points, incorporating more details. All the above successes are based on a large amount of labeled data, with cumbersome and time-consuming annotation of training data. To alleviate this problem, introducing the FSL algorithms that have attracted widespread attention in image tasks is considerable.

2.2. Few-shot learning for 2D image and 3D point cloud

2D image FSL. The meta-learning methods in Few-shot learning (FSL) methods are mostly designed for image tasks, and can be categorized [29] into model-based, optimization-based, and metric-based. ***Model-based*** methods aims to adapt to new tasks by changing the model's learnable parameters. ***Optimization-based*** methods regard the task as an optimization process. For example, MetaOptNet [30] incorporates a differentiable quadratic programming solver for a feature-relevant linear SVM predictor with better generalization for novel categories. ***Metric-based*** methods utilize the spatial relationship of features as a similarity measure, being more widely used due to its simplicity and effectiveness, in which our method is categorized. Therefore, the appropriate distance metric is crucial to its performance. We then review some representative metric-based methods.

Matching Network [31] adopts a bidirectional LSTM model to extract embeddings and uses cosine distance for classification. ProtoNet [7] treats the mean of support samples as prototypes that represent corresponding classes, and measures the similarity of features by Euclidean distance. However, due to the scarcity of support samples, the resulting prototypes usually deviate significantly from the ideal class centers. RelationNet [32] proposes a learnable metric module for similarity scores between prototypes and query samples.



(a). Embeddings in Euclidean space. (b). Embeddings in Hyperbolic space.

Fig. 2. The comparison of embeddings in (a) Euclidean space, and (b) Hyperbolic space. To emphasize, all black lines are with the same hyperbolic distance in (b).

To alleviate this problem, Prototype Rectification (PR) [9] uses query samples with pseudo labels to rectify prototypes and proposes two bias elimination methods to rectify the offset between the support set and query set. The prototype rectification method is semi-supervised few-shot learning, using only the support and query set. Yang et al. (DisCalib) [8,10] further incorporates distributional information for new sample generation using a large amount of unlabeled data and for semi-supervised few-shot learning. Additionally, CPN [33] learns CNN and prototypes jointly with discriminative & generative loss for open set recognition.

The above methods focus on image classification, with unsatisfying performance when directly applied to 3D point clouds, as they do not consider the unstructured property. Therefore, 3D Point Cloud FSL is introduced.

3D point cloud FSL. Recently, researchers also considered few-shot learning for more challenging 3D point clouds. CIA [13] firstly studies few-shot point cloud classification systematically and proposes the Cross-Instance Adaption module to alleviate the high intra-class variance and insignificant inter-class differences problems. ViewNet [15] further introduces a 2D projection-based network and view pooling block for more descriptive and distinguishing features. Yang et al. (CrossMod) [14] fuses features from raw point cloud data and its projection onto depth images to obtain more appropriate and robust features. The sqMA module [14] takes support-query as input as an improvement to prototype-query of CIF module [13]. Besides, ViewNet [15] and CrossMod [14] are projection-assisted, which may introduce computational burden for additional operations among: (1) 2D-plane projection, (2) extra parameters for additional image backbone, and (3) modal fusion stage. Also, these methods are performed in Euclidean space. GPr-Net [16] introduces hyperbolic metrics and utilizes intrinsic geometric descriptors with lightweight architecture, but is very sensitive to noises and other disruptions.

2.3. Hierarchical and hyperbolic embeddings

Regarding point cloud hierarchy. Hierarchy is an important and commonly existing multi-level structural information in 3D objects. PointNet++ [23] is the early hierarchy-aware method. PointGLR [11] further concentrates on the part-whole hierarchy within objects, with hypersphere mapping at different scale levels. HGNet [17] constructs local geometry structures for hierarchical geometry.

Hyperbolic embedding. More systematically, the hyperbolic space is especially proposed for hierarchical embedding, with a comparison to Euclidean embedding provided in Fig. 2. Most of them use Poincaré disk [34] with assistance of Klein model [18]. Nickel et al. [34] firstly introduces the Poincaré disk model to learn an embedding with latent hierarchical structures. Khrulkov et al. [18] applies the Klein model for averaging operation. It is also proved that the Poincaré disk has a larger representation capacity and better generalization ability than in Euclidean space and is detailed in [35] on tree-structure with combinatorial construction. HyperVAEs [36,37] thoroughly formulates distributional information in hyperbolic space. Recently, it has been applied in knowledge graph [38], and temporal link prediction [39]. DHA-Net [40] combines hyperbolic metric with DGCNN for object reconstruction.

For few-shot learning, it demonstrates promising potential as offering spatial geometric prior, without extra modules. HyperProtoNet [18] performs few-shot image classification in hyperbolic space, benefiting from latent hierarchical structures between images. Similarly, HyperZSL [41] is for zero-shot recognition, also preserving the hierarchical structure of semantic classes in low dimensions. Recently, there has been increasing attention paid to 3D point clouds. HyCoRe [12] effectively leverages the properties of hyperbolic space to extract and embed the compositional structure of point clouds, preserving the part-whole hierarchy. GPr-Net [16] as illustrated above for few-shot learning, also applied hyperbolic embedding and geometric descriptors.

Inspired by the above works, we perform FSL for point cloud classification in hyperbolic space to preserve the hierarchical relationship between point cloud objects that benefit FSL, while not introducing extra parameters to be trained or additional modules. By combining existing well-performed FSL methods in prototype rectification to improve the robustness of previous works, leveraging the unique advantages of hyperbolic space, and systematically incorporating spatial and distributional information, we proposed **HPR** in this paper.

3. Preliminaries

In this section, we present preliminaries of few-shot learning in Section 3.1, and Hyperbolic space in Section 3.2, both establishing foundations of our **HPR**.

3.1. Problem setting of few-shot learning for classification

We follow the same FSL setting as 2D image few-shot classification [30] and utilize meta-learning [29] procedure to deal with the FSL problem, the process is shown in Fig. 3.

Given a point cloud dataset with data-label pairs $D = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, M\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ represents the feature vector of a point cloud object and its label $y_i \in C$ as the set of class labels. The set C is divided into basic (train) classes C_b for meta-training and novel (test) classes C_n for meta-testing, where $C_b \cap C_n = \emptyset$ and $C_b \cup C_n = C$. The objective is to initially train the model on samples from the basic classes and perform well when tasked with new samples from the novel classes.

Meta-learning. It is usually divided into meta-training and meta-testing, formulated as N-way K-shot (N class, each class with K samples in support set) [6], where each task for meta-testing consists of the labeled support set and the unlabeled query set. Generally, each task has a small number of labeled samples to measure the generalization ability of the model. We have: (1) Support set S =

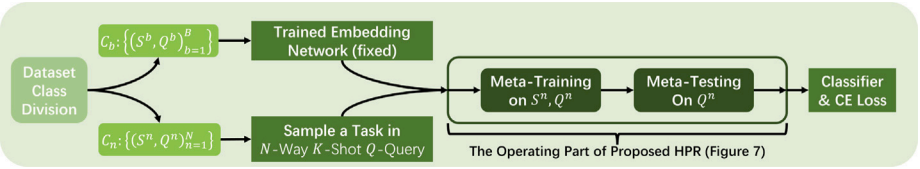


Fig. 3. The Meta-Learning Process. Our proposed **HPR** operates in the framed part.

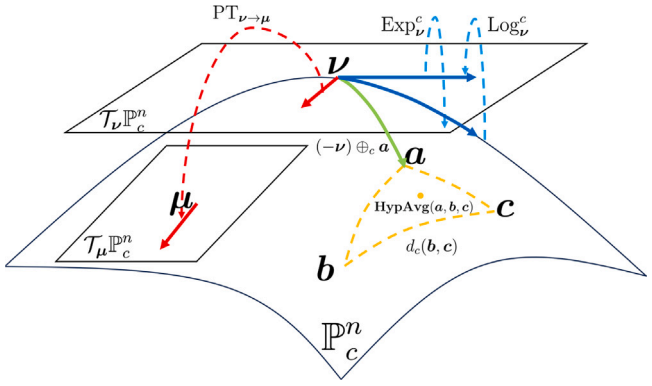


Fig. 4. Basic operators and their usages in hyperbolic space, including ① Möbius Addition (subtraction form), ② Hyperbolic Distance and Average, ③Exponential & Logarithmic Mapping, and ④ Parallel Transport. $\boldsymbol{\mu} \in \mathbb{P}_c^n$, $\mathcal{T}_{\boldsymbol{\mu}}\mathbb{P}_c^n$ is tangent space of $\mathbb{P}_c^n$ at $\boldsymbol{\mu}$, same as others.

$\{(\boldsymbol{x}_{i_s}, y_{i_s})\}_{i=1\,s=1}^{N\quad K}$, with N classes each contains K samples, and (2) Query set $Q = \{(\boldsymbol{x}_{i_q}, y_{i_q})\}_{i=1\,q=K+1}^{N\quad K+Q}$, with same N classes each contain Q samples. We construct the input as $\mathcal{T}_b = \{(S^b, Q^b)\}_{b=1}^B$ for meta-training, with labels of Q_b for supervised learning to optimize the objective function:

$$(\theta^*, \phi^*) = \arg\min_{\theta,\phi} \mathcal{L}_\phi(\mathcal{T}_b; \theta) \tag{1}$$

where $\mathcal{T}_b$ is from the basic classes given labels, with parameter ϕ of the backbone network, and θ of the classifier. $\mathcal{L}_\phi(\mathcal{T}_b; \theta)$ is the cross-entropy loss:

$$\mathcal{L}_\phi(\mathcal{T}_b; \theta) = \mathbb{E}_{\mathcal{T}_b}\left[-\log p(\hat{y} = c|\boldsymbol{x}; \theta)\right],$$
$$p(y = c \mid \boldsymbol{x}; \theta) = \mathbf{Softmax}(D_\theta(F_\phi(\boldsymbol{x}))) \tag{2}$$

where $\boldsymbol{x}$ is the feature, $\hat{y}$ is the predicted label, F_ϕ is embedding network and D_θ is classifier. After meta-training, similarly, we construct novel (test) tasks set as $\mathcal{T}_n = \{(S^n, Q^n)\}_{n=1}^N$ sampled from the novel classes for meta-testing. We fix ϕ of F_ϕ, fine-tuning for θ of D_θ on S_n and testing on Q_n.

3.2. Mathematical foundations of hyperbolic geometry

Hyperbolic space is a non-Euclidean Riemannian space, possessing constant sectional curvature of -1. There are several isomorphic models of Hyperbolic space, such as the Poincaré disk model and Klein model, etc. In this paper, we adapt the Poincaré disk model.

3.2.1. Poincaré disk model and basic operators

Poincaré disk model $(\mathbb{P}_c^n, g^{\mathbb{P}})$ can be defined as $\mathbb{P}_c^n = \{x \in \mathbb{R}^n : c\|x\|^2 < 1, c \geq 0\}$, with the Riemannian metric $g^{\mathbb{P}}(x) = \lambda_x^2 g^{\mathbb{E}}$, where $\lambda_x = \frac{2}{1-c\|x\|^2}$ is conformal factor, $g^{\mathbb{E}} = \mathbb{I}_n$ is Euclidean metric tensor, and c is the curvature parameter of Poincaré disk. Hyperbolic space differs from the vector space defined in Euclidean space, and many operations in Hyperbolic space can be extended with the assistance of Möbius vector space. Here, following [18], for $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{P}_c^n$, we briefly introduce operations available on Poincaré disk. For a clearer understanding, we visualize the operators in Fig. 4.

Möbius addition. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{P}_c^n$, the Möbius addition $\oplus_c$ is defined as:

$$\boldsymbol{x} \oplus_c \boldsymbol{y} := \frac{(1 + 2c\langle \boldsymbol{x}, \boldsymbol{y}\rangle + c\|\boldsymbol{y}\|^2)\boldsymbol{x} + (1 - c\|\boldsymbol{x}\|^2)\boldsymbol{y}}{1 + 2c\langle \boldsymbol{x}, \boldsymbol{y}\rangle + c^2\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2} \tag{3}$$

where $\langle\cdot,\cdot\rangle$, $\|\cdot\|$ denotes Euclidean inner product and norm, respectively.

Möbius multiplication. Given the matrix $\mathbf{W} \in \mathbb{R}^{n'\times n}$ and $\boldsymbol{x} \in \mathbb{P}_c^n$, the Möbius multiplication $\otimes_c$ calculates multiplication in tangent space, formulated as:

$$\mathbf{W} \otimes_c \boldsymbol{x} := \frac{1}{\sqrt{c}} \tanh\left(\frac{\|\mathbf{W}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \tanh^{-1}\left(\sqrt{c}\|\boldsymbol{x}\|\right)\right)\frac{\mathbf{W}\boldsymbol{x}}{\|\mathbf{W}\boldsymbol{x}\|} \tag{4}$$

Hyperbolic distance. For $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{P}_c^n$, the hyperbolic distance is obtained as:

$$d_c(\boldsymbol{x}, \boldsymbol{y}) := \frac{2}{\sqrt{c}} \tanh^{-1}\left(\sqrt{c}\left\|-\boldsymbol{x} \oplus_c \boldsymbol{y}\right\|\right) \tag{5}$$

Exponential & logarithmic map. It establishes a bijection between tangent space at a point and Riemannian manifold. The exponential mapping takes a tangent vector $\boldsymbol{w} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{P}_c^n$ to the manifold $\mathbb{P}_c^n$, i.e., $\mathrm{Exp}_{\boldsymbol{x}} : \mathcal{T}_{\boldsymbol{x}}\mathbb{P}_c^n \to \mathbb{P}_c^n$, as:

$$\mathrm{Exp}_{\boldsymbol{x}}^c(w) := \boldsymbol{x} \oplus_c \left(\tanh\left(\sqrt{c}\frac{\lambda_{\boldsymbol{x}}^c\|\boldsymbol{w}\|}{2}\right)\frac{\boldsymbol{w}}{\sqrt{c}\|\boldsymbol{w}\|}\right),$$
$$\mathrm{Exp}_{\mathbf{0}}^c(w) := \frac{\tanh\left(\sqrt{c}\|\boldsymbol{w}\|\right)}{\sqrt{c}\|\boldsymbol{w}\|}\boldsymbol{w} \tag{6}$$

where $\mathcal{T}_{\boldsymbol{x}}\mathbb{P}_c^n \cong \mathbb{R}^n$ is the tangent space of the manifold $\mathbb{P}_c^n$ at $\boldsymbol{x}$. Therefore, $\mathrm{Exp}_{\boldsymbol{x}}^c(\boldsymbol{w}) = \boldsymbol{z}, \gamma(\mathbf{0}) = \boldsymbol{x}, \gamma(1) = \boldsymbol{z}, \gamma'(0) = w$. $\mathrm{Exp}_{\mathbf{0}}^c(w)$ is easier to compute and more commonly used as it maps from $\mathbf{0}$. Logarithmic mapping is the inverse of the exponential map, i.e.: $\mathrm{Log}_{\boldsymbol{x}} : \mathbb{P}_c^n \to \mathcal{T}_{\boldsymbol{x}}\mathbb{P}_c^n$, as:

$$\mathrm{Log}_{\boldsymbol{x}}^c(\boldsymbol{u}) = \frac{2}{\sqrt{c}\lambda_{\boldsymbol{x}}^c}\frac{\tanh^{-1}\left(\sqrt{c}\left\|-\boldsymbol{x} \oplus_c \boldsymbol{u}\right\|\right)}{\left\|-\boldsymbol{x} \oplus_c \boldsymbol{u}\right\|}(-\boldsymbol{x} \oplus_c \boldsymbol{u}),$$
$$\mathrm{Log}_{\mathbf{0}}^c(\boldsymbol{u}) = \frac{2}{\sqrt{c}\lambda_{\mathbf{0}}^c}\frac{\tanh^{-1}\left(\sqrt{c}\|\boldsymbol{u}\|\right)}{\|\boldsymbol{u}\|}\boldsymbol{u} \tag{7}$$

Averaging operation. We first give the averaging operation in the Klein model:

$$\mathbf{HypAvg\text{-}K}(\boldsymbol{x}_{1\mathbb{K}}, \ldots, \boldsymbol{x}_{n\mathbb{K}}) = \sum_{i=1}^n \xi_i \boldsymbol{x}_{i\mathbb{K}} / \sum_{i=1}^n \xi_{i\mathbb{K}}, \tag{8}$$

where $\boldsymbol{x}_{i\mathbb{K}}$ is the coordinate of $\boldsymbol{x}_i$ in Klein model, $\xi_i = \frac{1}{\sqrt{1-c\|\boldsymbol{x}_{i\mathbb{K}}\|^2}}$ is the Lorentz factor. Owing to the isomorphism of Poincaré and Klein model, the coordinate transform could be formed as:

$$f_{\mathbb{K}\to\mathbb{P}}(\boldsymbol{x}_{\mathbb{K}}) = \frac{\boldsymbol{x}_{\mathbb{K}}}{1 + \sqrt{1 - c\left\|\boldsymbol{x}_{\mathbb{K}}\right\|^2}}, \qquad f_{\mathbb{P}\to\mathbb{K}}(\boldsymbol{x}_{\mathbb{P}}) = \frac{2\boldsymbol{x}_{\mathbb{P}}}{1 + c\left\|\boldsymbol{x}_{\mathbb{P}}\right\|^2}, \tag{9}$$

where $\boldsymbol{x}_{\mathbb{P}}$ and $\boldsymbol{x}_{\mathbb{K}}$ are the coordinates in the Poincaré disk model and Klein model, respectively. Therefore, we first map the points from the Poincaré disk model to the Klein model and compute the average in the Klein model, then map it back to the Poincaré disk model, formulated as:

$$\mathbf{HypAvg\text{-}P}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = f_{\mathbb{K}\to\mathbb{P}}\left(\mathbf{HypAvg\text{-}K}\left(f_{\mathbb{P}\to\mathbb{K}}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)\right)\right). \tag{10}$$

Table 1
Comparisons between operations of DGCNN in Euclidean and Hyperbolic space, $\boldsymbol{x} \in \mathbb{R}^m, \boldsymbol{z} = \mathrm{Exp}_{\mathbf{0}}^c(\boldsymbol{x}) \in \mathbb{P}_c^m$. AGG(·) is aggregation function, here is Max-Pooling.

Module	Euclidean	Hyperbolic
Linear (Euc.)/Mobius (Hyp.)	$\mathbf{W}\boldsymbol{x} + \mathbf{b}$	$\mathbf{W} \otimes_c \boldsymbol{x} \oplus_c \mathbf{b}$
Activation σ	$\sigma^{\mathbb{E}}(\cdot) = \mathrm{LeakyReLU}(\cdot)$	$\sigma^{\mathbb{P}}(\cdot) = \mathrm{Exp}_{\mathbf{0}}^c(\mathrm{LeakyReLU}(\mathrm{Log}_{\mathbf{0}}^c(\cdot)))$
Edge Function	$h_\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \overline{h}_\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j - \boldsymbol{x}_i)$	$h_\Theta(\boldsymbol{x}_i, \boldsymbol{x}_j) = \overline{h}_\Theta(\boldsymbol{z}_i, \boldsymbol{z}_j \oplus_c (-\boldsymbol{z}_i))$
EdgeConv	$\sigma^{\mathbb{E}}(\mathrm{AGG}(\mathrm{Linear}(\boldsymbol{x})))$	$\sigma^{\mathbb{P}}(\mathrm{AGG}(\mathrm{Mobius}(\boldsymbol{z})))$

Parallel transport. The parallel transport mapping from a vector $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{v}}\mathbb{P}_c^n$ to another tangent space $\mathcal{T}_{\boldsymbol{\mu}}\mathbb{P}_c^n$ is given by:

$$\mathrm{PT}_{\boldsymbol{v}\to\boldsymbol{\mu}}^c(\boldsymbol{v}) = \mathrm{Log}_{\boldsymbol{\mu}}^c(\boldsymbol{\mu} \oplus_c \mathrm{Exp}_{\boldsymbol{v}}^c(\boldsymbol{v})), \qquad \mathrm{PT}_{\mathbf{0}\to\boldsymbol{\mu}}^c(\boldsymbol{v}) = \frac{\lambda_{\mathbf{0}}^c}{\lambda_{\boldsymbol{\mu}}^c}\boldsymbol{v} \tag{11}$$

3.2.2. Hyperbolicity for quantified analysis on hierarchy

In addition to the qualitative analysis, a quantitative analysis of the hierarchical relationships between point cloud objects can also be conducted using the relative hyperbolicity δ_{rel} to compute the similarity between the dataset and Hyperbolic space. We refer to [18] for the calculation of hyperbolicity.

For metric space $\mathcal{X}$ be an arbitrary metric space (e.g.: Hyperbolic space) equipped with distance metric $d_c(\cdot,\cdot)$, given $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{X}$, the Gromov product is defined as: $(\boldsymbol{y}, \boldsymbol{z})_{\boldsymbol{x}} = \frac{1}{2}\left(d_c(\boldsymbol{x}, \boldsymbol{y}) + d_c(\boldsymbol{x}, \boldsymbol{z}) - d_c(\boldsymbol{y}, \boldsymbol{z})\right)$. Given a set of points and fixed $\boldsymbol{x}$, we firstly compute pairwise Gromov products A, then obtain δ as:

$$\delta = \max\{(A \otimes A) - A\} \tag{12}$$

where $\otimes$ denotes the min–max matrix product. To better validate the hierarchical assumption of the dataset, the relative hyperbolicity δ_{rel} is given as:

$$\delta_{rel}(\mathcal{X}) = \frac{2\delta(\mathcal{X})}{\mathrm{diam}(\mathcal{X})}. \tag{13}$$

where $\mathrm{diam}(\mathcal{X})$ is the set diameter (maximal pairwise distance) for scaling.

4. Method

In this section, we first present the hyperbolic metric learning paradigm in Section 4.1 and formally present the proposed **HPR** in Section 4.2. We present an enhanced metric learning with hyperbolic distance, namely Prototype Rectification with Hyperbolic Distance (**HPR-D**), in Section 4.2.1. Finally, we further consider the data distribution in hyperbolic space, i.e., Hyperbolic Wrapped Normal (HWN) distribution introduced in Section 4.2.2, and propose a novel metric learning method, namely Prototype Rectification with HWN Distribution (**HPR-WN**), in Section 4.2.3.

4.1. Hyperbolic metric learning for FSL

Before presenting the hyperbolic metric learning for FSL, we first introduce the backbone network for feature embedding, as in Section 4.1.1.

4.1.1. Feature embedding network

For feature embedding, selecting a backbone network involves multi-aspect considerations. DGCNN [25] is a widely adapted backbone for point cloud analysis. We utilize DGCNN for (1) stronger point-wise connection on graph compared with other backbones, and (2) fairer comparison as other methods (See Section 5.3) also utilize DGCNN as backbone. Same as [14], we only retain the max pooling for the last few layers of the original DGCNN and obtain the final output with dimension $m = 1024$, denoted as 'DGCNN†'. We utilized DGCNN† as the base backbone for comparisons on different baselines.

Furthermore, another concern is, that features are extracted in Euclidean metric for DGCNN. To operate in hyperbolic space for hierarchical feature embedding and for being more adapted in the following **HPR** module, inspired by DHA-Net [40], we utilize the hyperbolic dynamic graph convolution. We replace the main operations in the following:

- ***k*NN-Hyp** uses hyperbolic distance to search nearest neighbor
- **EdgeConv-Hyp** uses hyperbolic metric based edge function
- **MLP-Hyp** replaces the linear transform with the Möbius transform (adaptation of linear transform in hyperbolic space) and replaces activation with a hyperbolic activation function.

Table 1 compares different operations in Euclidean and Hyperbolic space. Fig. 5 describe the architecture of DGCNN†, DGCNN-Hyp(-Full). We did not employ a fully hyperbolic metric-based module, as described in DGCNN-Hyp-Full. For the EdgeConv-Hyp and MLP-Hyp layers in hyperbolic space, the computational burden increases as the layer gets propagated with the continually expanding receptive field of the hyperbolic dynamic graph. For 4-layer DGCNN-Hyp, parameters become difficult to update from the 2nd layer onwards.

In this way, we consider a trade-off and adopt the DGCNN-Hyp. We use kNN based on Euclidean distance for obtaining neighborhood as an approximation to kNN-Hyp based on hyperbolic distance. We retain the first layer of EdgeConv-Hyp and MLP-Hyp, and apply Logarithmic mapping $\mathrm{Log}_{\mathbf{0}}^c(\cdot)$ to map it back into Euclidean space. For the onward layers, it still operates in Euclidean space, the same as DGCNN†. We present their architectures in Fig. 5.

4.1.2. Hyperbolic metric learning for FSL

We adopt the hyperbolic metric for feature embedding to capture hierarchical feature similarity. We use the Poincaré disk model $\mathbb{P}_c^n$ of hyperbolic space as defined in Section 3.2.

Firstly, for an N-way K-shot FSL task (see Section 3.1), we are to obtain m-dimensional ($m = 1024$ in this paper) hyperbolic point cloud features for further operations. We denote Euclidean features as $\{\boldsymbol{x}_{i_s}\}_{i=1\,s=1}^{N\;K}$, $\boldsymbol{x}_{i_s} \in \mathbb{R}^m$ and Hyperbolic features as $\{\boldsymbol{z}_{i_s}\}_{i=1\,s=1}^{N\;K}$, $\boldsymbol{z}_{i_s} \in \mathbb{P}_c^m$, both indicating sth support feature of ith class.

Given features $\boldsymbol{x}$ obtained from a feature embedding network, we use exponential mapping $\mathrm{Exp}_{\mathbf{0}}^c : \mathbb{R}^m \to \mathbb{P}_c^m$ in Eq. (6) to map features from Euclidean space to Hyperbolic space, i.e., $\boldsymbol{z} = \mathrm{Exp}^c(\boldsymbol{x}) \in \mathbb{P}_c^m$. After hyperbolic feature $\boldsymbol{z}_k, \boldsymbol{z}_l$ are obtained, the hyperbolic distance between is $d_c(\boldsymbol{z}_k, \boldsymbol{z}_l)$ from Eq. (5).

With the hyperbolic metric from Section 3.2, we incorporate metric-based FSL methods in the following section, formally introducing **HPR**.

4.2. Hyperbolic Prototype Rectification (*HPR*)

In this section, we mainly introduce our **HPR**, branched into hyperbolic distance based **HPR-D** and HWN distribution based **HPR-WN**. We first briefly introduce the original ProtoNet and rectification process, then lead to **HPR-D** in Section 4.2.1. Furthermore, we illustrate the HWN distribution and its sampling strategy in Section 4.2.2, and then propose **HPR-WN** in Section 4.2.3.

ProtoNet [7] obtains appropriate class prototypes using **average** of class features on Euclidean metric, based on a trained feature embedding with a classifier on basic classes. By the nearest prototype

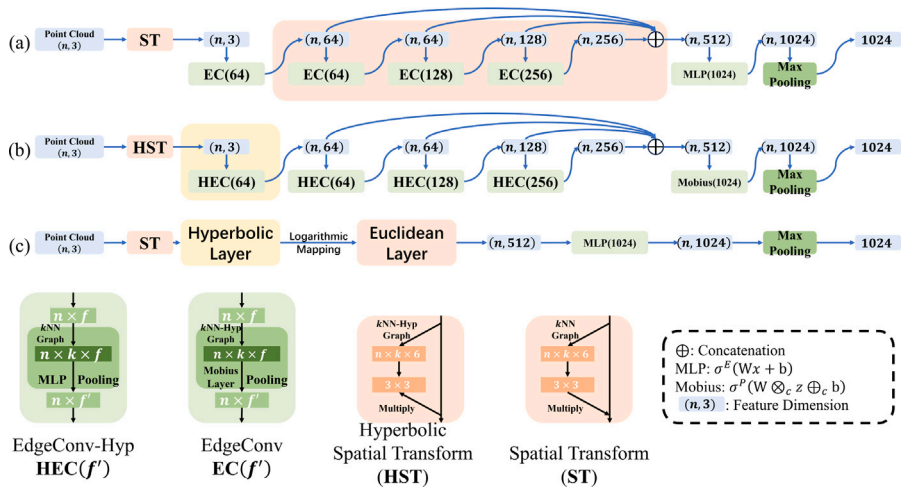


Fig. 5. Architectures of Feature Embedding Networks. (a) DGCNN† modified from [25], (b) DGCNN-Hyp-Full being a fully hyperbolic module, (c) DGCNN-Hyp balancing between Euclidean and Hyperbolic modules.

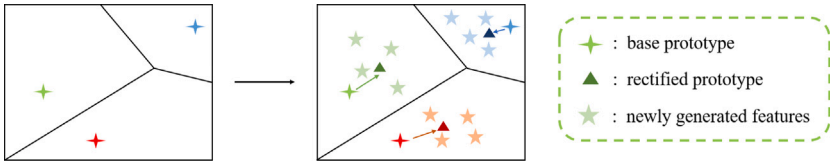


Fig. 6. Prototype rectification.

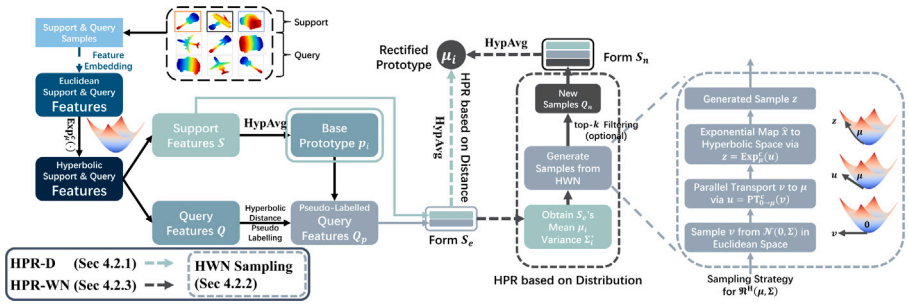


Fig. 7. The overview of proposed **HPR**. Feature embedding network, DGCNN† or DGCNN-Hyp as shown in Fig. 5, to obtain features. **HPR** module, in branched dotted arrows, for ① **HPR-D** for hyperbolic distance based prototype rectification, and ② **HPR-WN** for hyperbolic wrapped normal distribution based prototype rectification.

matching, it obtains the **Softmax**(·) output for classification as follows:

$$p(y=i \mid \boldsymbol{x}_q) = \textbf{Softmax}(\boldsymbol{x}_q, \boldsymbol{p}) = \frac{\exp(-d(\boldsymbol{x}_q, \boldsymbol{p}_i))}{\sum_{i=1}^{n} \exp(-d(\boldsymbol{x}_q, \boldsymbol{p}_i))} \tag{14}$$

where $\boldsymbol{x}_q$ is the query sample feature and $d(\cdot,\cdot)$ is the Euclidean distance.

PR [9] uses pseudo-labeled query samples to expand and enhance the features of the support set, being more robust against outlier or noise samples, thus avoiding overfitting and reducing biases. DisCalib [10] further includes distributional information. The prototype rectification process is shown in Fig. 6.

Inspired by these works, considering the potential hierarchical relationships among point cloud objects, we propose **HPR-D** and **HPR-WN** to rectify prototypes in hyperbolic space in the following (see Fig. 7).

4.2.1. Prototype rectification with hyperbolic distance (HPR-D)

The proposed **HPR-D** conducts pseudo-labeling on query samples based on nearest prototype matching, using hyperbolic distance as spatial relationship.

Hyperbolic prototype forming. Given hyperbolic features $\boldsymbol{z}_{i_j}$, hyperbolic class prototypes are firstly obtained via averaging operation for calculating the mean of support samples:

$$\boldsymbol{P}_i = \textbf{HypAvg-P}(\boldsymbol{z}_{i_j} \in S^i) \in \mathbb{R}^m, \qquad |S^i| = n_i \tag{15}$$

where $\boldsymbol{z}_{i_j}$ is the hyperbolic feature of the jth support sample in the support set S^i of class i.

Pseudo-labeling and enhanced-set forming. Then, we utilize the **Softmax**(·) output to obtain the pseudo-labels of query features:

$$p(y=i \mid \boldsymbol{z}_q) = \textbf{Softmax}(\boldsymbol{z}_q, \boldsymbol{P}) = \frac{\exp(-d_c(\boldsymbol{z}_q, \boldsymbol{P}_i))}{\sum_{i=1}^{n} \exp(-d_c(\boldsymbol{z}_q, \boldsymbol{P}_i))} \tag{16}$$

where $\boldsymbol{z}_q$ is the hyperbolic query sample feature, $d_c(\cdot,\cdot)$ is the hyperbolic distance defined in Eq. (5). Then we use the pseudo-labeled query sample set Q_p to expand the support set S and obtain the enhanced support set $S_e = S \cup Q_p$.

HPR based on hyperbolic distance. Hence, the new prototype of the features of class i in S_e is calculated as hyperbolic average of enhanced support, i.e., $\boldsymbol{P}'_i = \textbf{HypAvg-P}(\boldsymbol{z}'_{i_j} \in S^i_e)$, where $\boldsymbol{z}'_{i_j}$ is a hyperbolic feature

in S_e of class i, $|S_e| = \mathbf{n}_i$. Finally, we apply rectified prototypes $\boldsymbol{P}_i'$ to Eq. (16) for classification.

The above process considers the latent hierarchical relationships of point cloud objects. However, similar to the PR [9], this process does not generate new samples, and the pseudo-labeled query sample set is imbalanced in the number of classes. To solve these problems, we further utilize the hyperbolic wrapped normal distribution to generate novel samples that obey the hierarchical distributional information in the following.

4.2.2. Hyperbolic Wrapped Normal (HWN) distribution

Assuming class samples as a whole follow a normal distribution in feature space is a common strategy [10]. However, normal distributions cannot be directly utilized in hyperbolic space. There are many types of normal distributions in Hyperbolic space [36], including *Riemannian Normal* (HRN), *Wrapped Normal* (HWN) distribution, etc.

Rationale for choosing HWN. HRN requires a normalizing coefficient, while HWN utilizes exponential mapping from tangent space, defined as transporting points from normal distribution into hyperbolic space:

$$\mathcal{N}^{\mathrm{W}}_{\mathbb{P}^d_c}(\boldsymbol{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{d\nu^{\mathrm{W}}(\boldsymbol{z} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})}{d\mathcal{M}(\boldsymbol{z})} = \mathcal{N}\left(\mathrm{PT}^{-1}_{\mathbf{0}\to\boldsymbol{\mu}}\left(\mathrm{Log}_{\boldsymbol{\mu}}(\boldsymbol{z})\right) \mid \mathbf{0}, \boldsymbol{\Sigma}\right) \times \left(\frac{\sqrt{c}d_c(\boldsymbol{\mu}, \boldsymbol{z})}{\sinh\left(\sqrt{c}d_c(\boldsymbol{\mu}, \boldsymbol{z})\right)}\right)^{d-1} \tag{17}$$

Our rationale for choosing HWN includes ① easier to obtain without normalizing coefficient and ② being coherent with our feature embedding process as it is also exponential mapping based. Sampling could be conducted once the mean and covariance of the distribution are obtained, which is detailed below.

Sampling from HWN. We introduce the sampling strategy of Hyperbolic Wrapped Normal (HWN) distribution [36] prior to **HPR-WN** method in Section 4.2.3. Given distribution $\mathcal{N}^{\mathrm{W}}_{\mathbb{P}^d_c}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i^*)$, sampling in HWN involves more complicated but straightforward strategies than in Euclidean space. Based on the derivation of HWN from Eq. (17) and operators from Section 3.2, we could sample from the normal distribution in Euclidean space, then transport onto Hyperbolic space, as: ① sample $\boldsymbol{v} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_i^*)$ in the tangent space $\mathcal{T}_{\mathbf{0}}\mathbb{P}^m_c$, ② move $\boldsymbol{v}$ to $\boldsymbol{u} = \mathrm{PT}^c_{\mathbf{0}\to\boldsymbol{\mu}_i}(\boldsymbol{v}) \in \mathcal{T}_{\boldsymbol{\mu}_i}\mathbb{P}^m_c$ by parallel transport in Eq. (11), and ③ map $\boldsymbol{u}$ to $\boldsymbol{z} = \mathrm{Exp}^c_{\boldsymbol{\mu}_i}(\boldsymbol{u}) \in \mathbb{P}^m_c$ by exponential mapping in Eq. (6).

4.2.3. Prototype rectification with HWN distribution (*HPR-WN*)

Based on the HWN, we introduce **HPR-WN** in the following, as in Algorithm 1. We firstly obtain class prototype $\boldsymbol{P}_i$ from Eq. (15), and query samples to be classified using Eq. (16). We then rectify the hyperbolic prototype $\boldsymbol{P}_i$ using the distribution information of the query samples.

Algorithm 1 The **HPR-WN** for an N-way-K-shot-M-query task

Input: Hyperbolic Support set features $S = (\boldsymbol{z}_{i_s}, y_{i_s})^{N\ \ K}_{i=1\ s=1}$,
Hyperbolic Query set features $Q = (\boldsymbol{z}_{i_q}, y_{i_q})^{N\ \ K+M}_{i=1\ q=K+1}$
Output: New prototypes $\boldsymbol{P}_i''$

```
for (z_i, y_i) ∈ S, Q do
    Form pseudo-query set Q_p using Eq. (16) and expand S as S_e = S ∪ Q_p.
    Obtain mean μ_i and var Σ_i^* for class y_i based on Eqs. (18) and (20).
    Sample new features from the HWN of Eq. (21) and form Q_n.
    Obtain new support set as S_n = S_e ∪ Q_n.
    Obtain HypAvg-P(·) of Eq. (10) for each class i of S_n as P_i''.
end for
return P_i''
```

We assume that the features of each class are subject to HWN and the similar classes admit similar means and variances [10]. Therefore, to alleviate the bias of prototypes by insufficient samples, we consider using the distribution of query samples to generate sufficient samples to expand the support set.

Mean and covariance of enhanced-set. Same as Section 4.2.1, we firstly obtain the enhanced support set $S_e = S \cup Q_p$. To obtain the HWN distribution from S_e, we are to obtain $(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, the mean and covariance for each class i. The hyperbolic average of the features of class i is calculated as:

$$\boldsymbol{\mu}_i = \mathbf{HypAvg\text{-}P}(\boldsymbol{z}'_{i_j} \in S_e^i), \qquad |S_e^i| = \mathbf{n}_i \tag{18}$$

where $\boldsymbol{z}'_{i_j}$ is the hyperbolic features in S_e^i of class i and $\boldsymbol{\mu}_i$ is taken as the mean of the HWN.

Given the covariance matrix $\boldsymbol{\Sigma}$ is invariant in transforming between Euclidean and Hyperbolic space [36], the covariance matrix $\boldsymbol{\Sigma}_i$ of class i can be directly taken from Euclidean space:

$$\boldsymbol{\Sigma}_i = \frac{1}{\mathbf{n}_i - 1}\sum_{k=1}^{\mathbf{n}_i}(\boldsymbol{x}_k - \boldsymbol{\mu}'_i)(\boldsymbol{x}_k - \boldsymbol{\mu}'_i)^\top \in \mathbb{R}^{m\times m} \tag{19}$$

where $\boldsymbol{x}_k$ is the original Euclidean feature, recall $\boldsymbol{z}_k = \mathrm{Exp}^c_{\mathbf{0}}(\boldsymbol{x}_k)$, $\boldsymbol{\mu}'_i = \frac{1}{\mathbf{n}_i}\sum_{k=1}^{\mathbf{n}_i}\boldsymbol{x}_k$ is the average of the ith class in Euclidean space.

Trading-off between efficiency and performance. There are possible options. ① *Diagonalization* reduces the parameter number from m^2 to m, still preserving anisotropic property and variance of same dimension features, meanwhile reducing around half of sampling time. ② *Isotropic* further reduces the parameter number to 1, but performs worse as it deviates more from complete $\boldsymbol{\Sigma}$, also isotropic process involves a similar amount of time as diagonalization. Therefore, we employ diagonalization $\mathbf{diag}(\cdot)$ on each class i in S_e:

$$\boldsymbol{\Sigma}_i^* = \mathbf{diag}(\boldsymbol{\Sigma}_i) \in \mathbb{R}^m \tag{20}$$

Sample generation and new-set forming. After obtaining the mean and the covariance of each class, we can generate features from HWN for class i as:

$$\mathfrak{N}_i^{\mathrm{H}} = \{\boldsymbol{z} \mid \boldsymbol{z} \sim \mathcal{N}^{\mathrm{W}}_{\mathbb{P}^m_c}(\cdot \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i^*)\} \tag{21}$$

Then more diverse labeled samples can be generated for class i by sampling from the distribution $\mathfrak{N}_i^{\mathrm{H}}$. During actual implementation, we rely on the sampling process shown in Section 4.2.2, and generate the same number of samples for each class to avoid class imbalance issues.

In general, the classification based on class prototypes is subject to misclassification. Intuitively, to reduce the impact of misclassified samples on the results, we introduce the top-$\boldsymbol{k}$ setting. We perform the filtering strategy, using $\boldsymbol{k}$-NN based on the hyperbolic distance to prototypes for each class to selectively choose the generated features, that are assumed to have a higher probability of belonging to the class, thereby enhancing subsequent classification performance. We denote it as 'top-$\boldsymbol{k}$ **HPR-WN**'.

The generated pseudo-labeled samples of each class form Q_n, which is used to expand each class sample in the support set S_e to obtain a new support set $S_n = S_e \cup Q_n = S \cup Q_p \cup Q_n$. In this way, we can alleviate the lack of support samples and enrich the diversity of the samples.

Prototype rectification with new-set samples. Once S_n is available, the prototype $\boldsymbol{P}'_i$ of class i in S_n can be calculated as $\boldsymbol{P}''_i = \mathbf{HypAvg\text{-}P}(\boldsymbol{z}''_j \in S_n^i)$, $|S_n^i| = \mathfrak{n}_i$, where $\boldsymbol{z}''_{i_j}$ is the feature of the jth sample in S_n^i of class i. Then $\boldsymbol{P}''_i$ is used as the rectified class prototype, and final classification can be performed by applying $\boldsymbol{P}''_i$ to Eq. (16).

5. Experiments

In this section, we compare the proposed methods, i.e., **HPR-D** and **HPR-WN**, with the representative FSL methods [7,13–15,30,32], including state-of-the-art methods, on four datasets. Extensive ablation studies evaluated the effectiveness and generalization ability of **HPR**.

Table 2
Dataset description for our experiments.

Dataset	Source of data	# Class	# Object
ModelNet40 [42]	Sampling from CAD model	40	12 308
ModelNet40-C [43]	Perturbed in density	40	12 308
ScanObjectNN [44]	Real-World indoor scanning	15	2902
ScanObjectNN-PB [44] (PB_T50_RS)	Perturbed in translation, rotation, and scaling	15	14 510

5.1. Datasets

Description of dataset. We describe the dataset for our experiments statistically in Table 2. ModelNet40(-C) [42,43] are sampled from CAD models, which are more regular. ScanObjectNN(-PB) [44] are obtained from real-world scanning, from which our model's performance in the real world could be tested. ModelNet40-C [43] and ScanObjectNN-PB [44] are perturbed intentionally from the original dataset to demonstrate the robustness of our model. For reproducing real-world existing perturbations, we employed the density corruption of ModelNet40-C [43], and ScanObjectNN-PB [44] containing up to 50% of random translation, as well as perturbation in rotation and scaling.

Preprocessing of dataset. For few-shot classification, it is common to split the dataset classwisely for training and testing, respectively [15]. We split ModelNet40(-C) [42,43] into 4 groups, each with 10 classes; and ScanObjectNN(-PB) [44] into 3 groups, each with 5 classes. We select one group as the testing set and the rest as the training set for each dataset. For sampling, 1024 points are uniformly sampled and normalized to the unit sphere.

5.2. Implementation details

Hardware & software. Experiments are performed on a single Nvidia GeForce V100 GPU, with a CUDA version of 11.1, Pytorch version of 1.10.0, Python version of 3.6.13, and the system being CentOS 7.

FSL training details. The experiments are based on the standard task-based FSL setting in Section 3.1. For more fair comparison, following [13], we set 100 epochs, for each epoch: ① *Meta-training.* contains 400 tasks randomly sampled from the basic classes. Simultaneously, we apply random rotating and jittering to augment samples. ② *Meta-testing.* We randomly sampled 700 tasks from the novel classes. Specifically, for experiments on ModelNet40-C, we utilize original ModelNet40 samples for meta-training and density-corrupted samples for meta-testing to demonstrate robustness.

Hyperparameter settings. The input batch size is set to 32. All experiments adopt the Adam optimizer with an initial learning rate (lr) of 1e−3 and $\gamma = 0.8$ as the lr coefficient, that learning rate decreases every 5 epochs. Selecting proper curvature c allows us to balance between Hyperbolic and Euclidean geometries. For **HPR(-D/WN)**, following [18], we set curvature c to 0.01 for 5-way 1-shot tasks and 0.005 for 5-way 5-shot tasks. We set the number of generated samples from HWN for each class as 40. This is validated in our experiments in Fig. 11(a)(b).

Evaluation. The accuracy presented in all results on accuracy is the average over meta-testing with 95% confidence intervals, with results presented as 'result±std.', the standard from multiple experiments.

5.3. Results and analysis

General results. Tables 3 and 4 are results of few-shot learning experiments on 4 datasets with 5-way-K-shot-10-query, $K = 1, 5$ settings. 'DGCNN*' denotes the improved DGCNN in [14] modifying the average/max pooling layers, note that we do not include the image backbone for a fairer comparison. DGCNN† and DGCNN-Hyp are mentioned in Section 4.1.1 and in Fig. 5.

Dgcnn† based. On the ModelNet40 and ModelNet40-C datasets, the average accuracy of our methods compared with SOTA CrossMod [14] is about 1.48% lower under 1-shot setting and about 0.37% higher under 5-shot setting. On the ScanObjectNN and ScanObjectNN-PB, the average accuracy of our methods compared with SOTA ViewNet is about 0.50% and 0.29% higher under the 1-shot setting and about 3.66% and 1.02% higher under the 5-shot setting.

DGCNN-Hyp based. Compared with DGCNN† within our method, DGCNN-Hyp outperforms half of the results in Tables 3 and 4, and brings up to 0.28% improvement, which is a bit limited. Considering that operations on hyperbolic space involve vast exponential calculation for the feature embedding network, we continue using DGCNN† as the feature embedding network in the following.

Analysis. Our methods heavily rely on the feature embedding capability of the backbone network, whereas CrossMod [14] takes into account the relationship between support and query features (mutual attention) after the backbone network, and CIA [13] also applied cross-instance module, benefiting effective features. ViewNet [15] completely rely on view-projected images, which fails to consider unique properties in 3D point cloud. Therefore, under the 1-shot setting, our methods perform worse on ModelNet40 and ModelNet40-C datasets. However, as the number of support set samples increases, features extracted by the embedded network become more accurate. In such a more accurate feature space, our methods bring greater improvements by obtaining a more suitable prototype, outperforming existing methods without any extra module.

For the ScanObject-NN and ScanObjectNN-PB datasets, results presented tend to be more sensitive to noise, leading to biases on prototypes, reflected in a larger accuracy gap to that obtained from ModelNet40(-C). In this case, the proposed methods are capable of finding more accurate class prototypes, effectively improving classification accuracy, and thus outperforming baselines.

In general, we achieve SOTA results on point-based methods on 4 datasets. Including the comparison of the view-based method, we achieve SOTA results on real-world ScanObjectNN(-PB) datasets and demonstrate high robustness against perturbations. Further experiments are conducted below.

5.4. Ablation study

This section presents ablation studies on the effectiveness of our methods.

Regarding hyperbolic embedding and prototype rectification. Table 5 presents detailed comparisons of **HPR** variants, including those in Euclidean space. The 'PR' and 'PR-N' denote the prototype rectification based on Euclidean distance and normal distribution, respectively. For a fair comparison, PR only uses intra-class bias elimination [9] in relevant experiments. The HyperProtoNet outperforms ProtoNet by around 0.07%∼1.66%. The average accuracy of our **HPR-D** and **HPR-WN** is about 0.61% and 0.19% higher than the corresponding PR and PR-N, respectively. In visualization, Fig. 8(a),(b) shows the t-SNE visualization between ProtoNet and HyperProtoNet, indicating the higher discrimination of Hyperbolic embedding, providing a more precise metric space for few-shot point cloud classification tasks with the hierarchical prior and improves performance. Fig. 9(a), (b) shows the t-SNE visualization before and after prototype rectification. It explicitly demonstrates the effectiveness of rectification, further enlarging discrimination, while being more robust to the entire class.

Table 3
Few-shot classification accuracy with 95% confidence intervals on ModelNet40 and ModelNet40-C. **Bold** denotes optimal result. Within our methods, underline denotes optimal, and underwave denotes suboptimal.

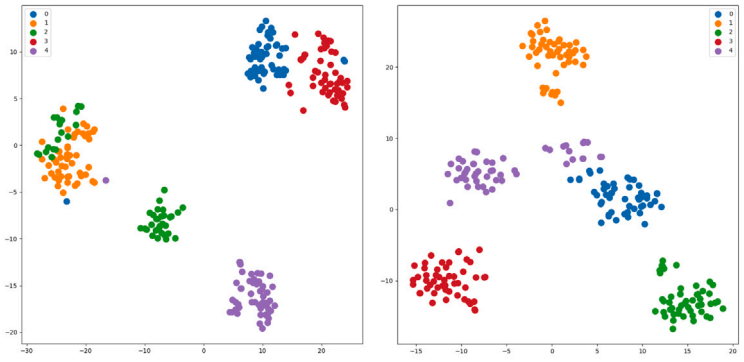
Method	Year	Backbone	ModelNet40		ModelNet40-C	
			5way-1shot	5way-5shot	5way-1shot	5way-5shot
ProtoNet [7]	2017	DGCNN	79.46 ± 0.76	88.65 ± 0.54	77.69 ± 0.77	86.81 ± 0.58
RelationNet [32]	2018	DGCNN	77.46 ± 0.80	85.11 ± 0.61	–	–
MetaOptNet [30]	2019	DGCNN	75.77 ± 0.83	86.44 ± 0.62	73.34 ± 0.85	85.15 ± 0.63
CIA [13]	2022	DGCNN	83.46 ± 0.83	89.15 ± 0.50	80.64 ± 0.86	88.23 ± 0.54
CrossMod [14]	2023	DGCNN*	**83.89 ± 0.75**	90.64 ± 0.52	**81.87 ± 0.78**	89.51 ± 0.54
ViewNet [15]	2023	ViewNet	82.68 ± 0.80	89.64 ± 0.55	81.05 ± 0.78	88.75 ± 0.49
HPR-D (ours)	/	DGCNN†	83.34 ± 0.78	90.64 ± 0.52	79.37 ± 0.79	**90.24 ± 0.50**
		DGCNN-Hyp	83.40 ± 0.77	**90.66 ± 0.52**	79.35 ± 0.80	90.20 ± 0.49
HPR-WN (ours)	/	DGCNN†	83.03 ± 0.75	90.34 ± 0.50	80.00 ± 0.74	90.00 ± 0.51
		DGCNN-Hyp	83.10 ± 0.76	90.38 ± 0.51	80.08 ± 0.73	89.92 ± 0.50

Table 4
Few-shot classification accuracy with 95% confidence intervals on ScanObjectNN and ScanObjectNN-PB. **Bold** denotes optimal result. Within our methods, underline denotes optimal, and underwave denotes suboptimal.

Method	Year	Backbone	ScanObjectNN		ScanObjectNN-PB	
			5way-1shot	5way-5shot	5way-1shot	5way-5shot
ProtoNet [7]	2017	DGCNN	60.46 ± 0.67	70.20 ± 0.52	59.29 ± 0.65	67.68 ± 0.47
RelationNet [32]	2018	DGCNN	54.23 ± 0.63	66.72 ± 0.50	–	–
MetaOptNet [30]	2019	DGCNN	61.12 ± 0.66	67.73 ± 0.45	57.15 ± 0.63	65.56 ± 0.50
CIA [13]	2022	DGCNN	62.17 ± 0.68	71.31 ± 0.45	57.02 ± 0.68	67.37 ± 0.49
CrossMod [14]	2023	DGCNN*	64.69 ± 0.64	74.60 ± 0.43	60.25 ± 0.63	71.00 ± 0.47
ViewNet [15]	2023	ViewNet	66.48 ± 0.60	74.77 ± 0.45	–	–
HPR-D (ours)	/	DGCNN†	66.04 ± 0.61	78.43 ± 0.46	**60.54 ± 0.64**	71.34 ± 0.48
		DGCNN-Hyp	66.12 ± 0.60	**78.49 ± 0.46**	60.50 ± 0.65	71.42 ± 0.48
HPR-WN (ours)	/	DGCNN†	**66.98 ± 0.68**	78.00 ± 0.45	60.15 ± 0.64	72.02 ± 0.49
		DGCNN-Hyp	66.70 ± 0.68	78.14 ± 0.45	60.09 ± 0.63	**72.05 ± 0.49**

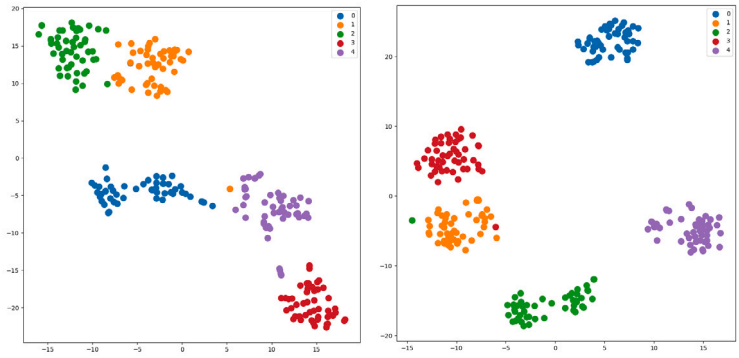
Table 5
Ablation study of HPR on ModelNet40 and ScanObjectNN datasets with different variants, comparing ① metric space, ② spatial relationship/distributional information, and ③ sampling strategy. **Bold** denotes optimal results, and underline denotes suboptimal results.

Method	Backbone	ModelNet40		ScanObjectNN	
		5way-1shot	5way-5shot	5way-1shot	5way-5shot
ProtoNet [7]	DGCNN†	80.65 ± 0.74	90.37 ± 0.50	64.10 ± 0.57	76.24 ± 0.44
HyperProtoNet [41]	DGCNN†	80.72 ± 0.72	90.47 ± 0.48	65.31 ± 0.59	77.90 ± 0.45
PR [9]	DGCNN†	82.77 ± 0.79	90.62 ± 0.54	65.34 ± 0.64	77.29 ± 0.44
HPR-D (ours)	DGCNN†	**83.34 ± 0.78**	**90.64 ± 0.52**	66.04 ± 0.61	**78.43 ± 0.46**
PR-N (ours)	DGCNN†	83.03 ± 0.77	90.62 ± 0.54	65.36 ± 0.64	77.27 ± 0.43
HPR-WN (ours)	DGCNN†	83.03 ± 0.75	90.34 ± 0.50	**66.98 ± 0.68**	78.00 ± 0.45
HPR-WN top-k (ours)	DGCNN†	83.01 ± 0.75	90.30 ± 0.50	66.81 ± 0.70	78.00 ± 0.47



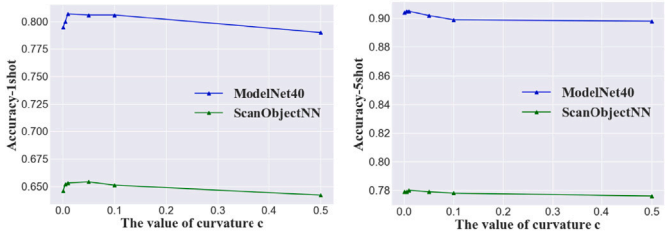
(a). Euclidean. (b). Hyperbolic.

Fig. 8. The t-SNE visualization of (a), (b): the feature embeddings for different spaces. All visualizations use the ModelNet40 dataset, with 50 query samples for each of 5 classes, i.e., 50 × 5 samples in total.



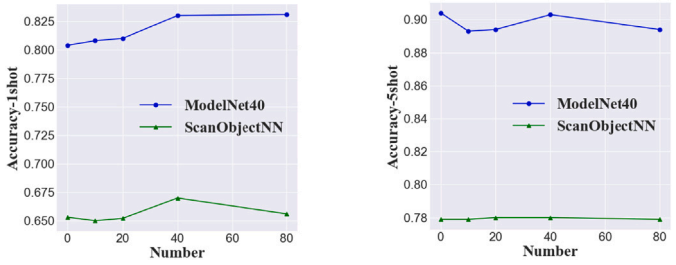
(a). Before Rectify. (b). After Rectify.

Fig. 9. The t-SNE visualization of and (a), (b): before and after prototype rectification. All visualizations use the ModelNet40 dataset, with 50 query samples for each of 5 classes, i.e., 50 × 5 samples in total.



(a). Curvature c (1 shot). (b). Curvature c (5 shot).

Fig. 10. (a), (b): The effect of the curvature parameter. Accuracy on ModelNet40 and ScanObjectNN for (a) 1-shot setting and (b) 5-shot setting when increasing the curvature (horizontal axis) of hyperbolic space.



(a). Number (1 shot). (b). Number (5 shot).

Fig. 11. (a), (b): The effect of the number of generated features. Accuracy on ModelNet40 and ScanObjectNN for (a) 1-shot setting and (b) 5-shot setting when increasing the number (horizontal axis) of generated features.

Regarding curvature of the model. Reviewing the definition of the Poincaré disk in Section 3.2, curvature controls the size of the Poincaré disk, smaller curvature leads to a larger size of the Poincaré disk. A proper curvature could balance between Euclidean and Hyperbolic geometry. In comparison to the 1-shot setting, the 5-shot setting has a larger sample size, requiring a larger capacity of disk for embedding, hence a smaller curvature would be more effective. Fig. 10(a)(b) shows the effect of curvature of Poincaré disk. For the performance on two datasets, as curvature increases, they all experience growth in performance firstly, followed by a steady decrease. As different Ks lead to different sample sizes, selecting proper c could better embed different sizes of data. We set $c = 0.01$ for $K = 1$, and $c = 0.005$ for $K = 5$ for the results presented in previous experiments.

Regarding HPR-WN. Table 5 shows the comparison results between **HPR** and other baseline methods. Compared with ProtoNet and HyperProtoNet, the proposed methods achieve around 0.17%∼2.88% improvement on ModelNet40 and ScanObjectNN under all settings. The improvement brought by the **HPR-WN** is greatly reduced under the 5-shot setting, which indicates that the prototypes are already accurate with more support samples. Overall, our methods achieve a satisfying rectification effect on class prototypes with improved generalization ability. Fig. 11(a)(b) shows the effect of the number of features generated by HWN. With increasing sampling number $\boldsymbol{k}$, the classification accuracy initially rises before stabilizing, indicating the benefit of appropriate $\boldsymbol{k}$ for balance between performance and efficiency. Relatively high accuracy and low computation cost are obtained when $\boldsymbol{k} = 40$, being utilized for **HPR-WN** setting. Furthermore, for the top-$\boldsymbol{k}$ **HPR-WN** setting, following the same consideration and validated by experiments, we sample 70 samples and obtain 40 samples.

As is shown in Table 5, **HPR-WN** does not bring full improvement among **HPR-D**. We assume that the sampling process introduces bias against the original samples, resulting in generated samples deviating from the original samples. Table 6 presents our computation on δ-hyperbolicity [18] (a metric on the degree of hierarchy) using different data. It confirmed our assumption, as new samples show lower hyperbolicity.

Additionally, compared with **HPR-WN** without top-$\boldsymbol{k}$, the assistance of top-$\boldsymbol{k}$ setting does not yield any better results. This could be because the learned features already have high intra-class variance and low inter-class variance. Selecting the top-$\boldsymbol{k}$ features based on distance does not effectively improve the accuracy of pseudo labels of generated features nor brings effective improvements in the results. This suggests that relying solely on distance-based criteria for class belonging is inaccurate.

6. Discussion

6.1. Shortcomings & limitations

During the research, there are still some insufficiencies and many areas worth further exploration: ① The calculation in hyperbolic space is more costly than in Euclidean space. Although numerous simplifications are used in our work, some calculation can still be further optimized. Also, exploring a better hyperbolic feature embedding module (beyond DGCNN), while combining with strategies like support-query attention, could further improve the results. ② The **HPR-WN** generated samples does not fully conform point cloud prior distribution, and its top-$\boldsymbol{k}$ setting does not obtain the expected effect. Further

Table 6
Comparison of relative hyperbolicity δ_{rel} on different data, 'HWN generated' indicates samples generated from HWN distribution. All are obtained from an average of 40 batches (batch size is 32). $\delta_{rel} \to 0$ indicate a stronger hyperbolicity of the data.

Data	HWN generated	ModelNet40	ModelNet40-C	ScanObjectNN	ScanObjectNN-PB
δ_{rel}	0.10	0.24	0.23	0.22	0.26

investigation on distributional rectification strategy in hyperbolic space holds promise. ③ More visualization beyond presented t-SNE on the hierarchical structure could benefit the description of the hierarchical nature of point clouds. ④ While our methods achieve satisfying results in the mentioned benchmarks, we still admit the possible biases exist in these currently widely used datasets due to limited class number. It could be validated more comprehensively with datasets of larger class number, which could also further validate the generalization ability.

Overall, although model performance and efficiency could be further improved, we do not observe any weakness that could affect our result by a significant level. Experiments are validated multiple times for one result with deviation presented.

6.2. Future perspectives

Our work demonstrated the effectiveness of hyperbolic space for point cloud embedding and few-shot learning. Besides the further advancement in our approach, it could be combined with other methods, like the existing ones we compared. Also, its potential can be further explored in more complicated tasks in open-world learning (OWL) [45], including zero-shot learning (ZSL), class-incremental learning (CIL) and open-set recognition (OSR).

7. Conclusion

Deep learning for few-shot point cloud classification faces challenges in overfitting, generalization, and robustness. In this paper, we utilize hyperbolic space as a hierarchical structural geometric prior for hierarchical feature representation and propose a novel **HPR** method for few-shot point cloud classification combined with metric-based few-shot learning. Compared with existing methods, our **HPR** requires no other operation than exponential mapping and uses features directly extracted from points. The experiments verify that hyperbolic space provides a more accurate metric space few-shot point cloud classification. Also, extensive experiments demonstrate that the proposed **HPR** achieves satisfying results, exhibiting higher robustness for noises and perturbations with state-of-the-art results. It also states the potential for point-based methods against projection-based methods.

During the systematic attempt to use hyperbolic metrics for 3D point cloud FSL, we experienced several limitations. Our method puts the main focus on post-feature embedding and lacks consideration of the feature embedding process. Even though a variant of backbone network operating in hyperbolic space is introduced, it is a trade-off against computational burden. Designing efficient backbone networks better suited for hyperbolic space is a future direction, benefiting tasks beyond FSL, into open-world learning tasks. Besides, the distribution-based rectification did not obtain the expected results due to the complex statistical and geometrical aspects of the data. It is worth further investigation and method development.

CRediT authorship contribution statement

Yuan-Zhi Feng: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Shing-Ho J. Lin:** Writing – review & editing, Writing – original draft, Resources, Project administration, Investigation, Funding acquisition, Formal analysis, Software. **Xuan Tang:** Supervision, Resources, Funding acquisition, Conceptualization. **Mu-Yu Wang:** Software, Formal analysis, Data curation. **Jian-Zhang Zheng:** Writing – review & editing, Resources, Formal analysis. **Zi-Yao He:** Investigation, Data curation. **Zi-Yi Pang:** Investigation. **Jian Yang:** Resources, Funding acquisition. **Ming-Song Chen:** Resources. **Xian Wei:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Code for reproduction is publicaly available at: https://github.com/Jonathan-UCAS/HPR.

Acknowledgments

The work was supported by the National Natural Science Foundation of China under No. 42130112 and No. 42371479, and the Beijing Natural Science Foundation No. QY23187.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.patcog.2024.111042.

References

[1] L. Zou, Z. Huang, N. Gu, G. Wang, Learning geometric consistency and discrepancy for category-level 6D object pose estimation from point clouds, Pattern Recognit. 145 (2024) 109896.
[2] R. Qian, X. Lai, X. Li, 3D object detection for autonomous driving: A survey, Pattern Recognit. 130 (2022) 108796.
[3] C. Zhao, J. Yang, X. Xiong, A. Zhu, Z. Cao, X. Li, Rotation invariant point cloud analysis: Where local geometry meets global topology, Pattern Recognit. 127 (2022) 108626.
[4] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun, Deep learning for 3D point clouds: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 43 (12) (2021) 4338–4364.
[5] Y. Zhu, R. Mottaghi, E. Kolve, J.J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in: ICRA, 2017, pp. 3357–3364.
[6] Y. Song, T. Wang, P. Cai, S.K. Mondal, J.P. Sahoo, A comprehensive survey of few-shot learning: Evolution, applications, challenges, and opportunities, ACM Comput. Surv. 55 (13s) (2023).
[7] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: NeurIPS, Vol. 30, 2017.
[8] S. Yang, S. Wu, T. Liu, M. Xu, Bridging the gap between few-shot and many-shot learning via distribution calibration, IEEE Trans. Pattern Anal. Mach. Intell. 44 (12) (2022) 9830–9843.
[9] J. Liu, L. Song, Y. Qin, Prototype rectification for few-shot learning, in: ECCV, Vol. 12346, 2020, pp. 741–756.
[10] S. Yang, L. Liu, M. Xu, Free lunch for few-shot learning: Distribution calibration, in: ICLR, 2021.
[11] Y. Rao, J. Lu, J. Zhou, Global-local bidirectional reasoning for unsupervised representation learning of 3D point clouds, in: CVPR, 2020.
[12] A. Montanaro, D. Valsesia, E. Magli, Rethinking the compositionality of point clouds through regularization in the hyperbolic space, in: NeurIPS, Vol. 35, 2022.
[13] C. Ye, H. Zhu, Y. Liao, Y. Zhang, T. Chen, J. Fan, What makes for effective few-shot point cloud classification? in: WACV, 2022, pp. 267–276.
[14] M. Yang, J. Chen, S. Velipasalar, Cross-modality feature fusion network for few-shot 3D point cloud classification, in: WACV, 2023, pp. 653–662.
[15] J. Chen, M. Yang, S. Velipasalar, ViewNet: A novel projection-based backbone with view pooling for few-shot point cloud classification, in: CVPR, 2023, pp. 17652–17660.

[16] T. Anvekar, D. Bazazian, GPr-net: Geometric prototypical network for point cloud few-shot learning, in: CVPR Workshops, 2023, pp. 4179–4188.
[17] T. Yao, Y. Li, Y. Pan, T. Mei, HGNet: Learning hierarchical geometry from points, edges, and surfaces, in: CVPR, 2023, pp. 21846–21855.
[18] V. Khrulkov, L. Mirvakhabova, E. Ustinova, I. Oseledets, V. Lempitsky, Hyperbolic image embeddings, in: CVPR, 2020, pp. 6418–6428.
[19] H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3D shape recognition, in: ICCV, 2015, pp. 945–953.
[20] X. Wei, R. Yu, J. Sun, View-GCN: View-based graph convolutional network for 3D shape analysis, in: CVPR, 2020, pp. 1847–1856.
[21] A. Goyal, H. Law, B. Liu, A. Newell, J. Deng, Revisiting point cloud shape classification with a simple and effective baseline, in: ICML, Vol. 139, 2021, pp. 3809–3820.
[22] R.Q. Charles, H. Su, M. Kaichun, L.J. Guibas, PointNet: Deep learning on point sets for 3D classification and segmentation, in: CVPR, 2017, pp. 77–85.
[23] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: Deep hierarchical feature learning on point sets in a metric space, in: NeurIPS, Vol. 30, 2017.
[24] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, B. Chen, PointCNN: Convolution on X-Transformed points, in: NeurIPS, Vol. 31, 2018.
[25] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph CNN for learning on point clouds, ACM Trans. Graph. 38 (5) (2019).
[26] H. Thomas, C.R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, L.J. Guibas, KPConv: Flexible and deformable convolution for point clouds, in: ICCV, 2019.
[27] X. Ma, C. Qin, H. You, H. Ran, Y. Fu, Rethinking network design and local geometry in point cloud: A simple residual MLP framework, in: ICLR, 2022.
[28] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R.R. Martin, S.-M. Hu, Pct: Point cloud transformer, Comput. Vis. Media 7 (2021) 187–199.
[29] Y. Tian, X. Zhao, W. Huang, Meta-learning approaches for learning-to-learn in deep learning: A survey, Neurocomputing 494 (2022) 203–223.
[30] K. Lee, S. Maji, A. Ravichandran, S. Soatto, Meta-learning with differentiable convex optimization, in: CVPR, 2019, pp. 10649–10657.
[31] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, in: NeurIPS, Vol. 29, 2016.
[32] F. Sung, Y. Yang, L. Zhang, T. Xiang, P.H. Torr, T.M. Hospedales, Learning to compare: Relation network for few-shot learning, in: CVPR, 2018, pp. 1199–1208.
[33] H.-M. Yang, X.-Y. Zhang, F. Yin, Q. Yang, C.-L. Liu, Convolutional prototype network for open set recognition, IEEE Trans. Pattern Anal. Mach. Intell. 44 (5) (2022) 2358–2370.
[34] M. Nickel, D. Kiela, Poincaré embeddings for learning hierarchical representations, in: NeurIPS, Vol. 30, 2017.
[35] F. Sala, C. De Sa, A. Gu, C. Re, Representation tradeoffs for hyperbolic embeddings, in: ICML, Vol. 80, 2018, pp. 4460–4469.
[36] E. Mathieu, C. Le Lan, C.J. Maddison, R. Tomioka, Y.W. Teh, Continuous hierarchical representations with Poincaré variational auto-encoders, in: NeurIPS, Vol. 32, 2019.
[37] Y. Nagano, S. Yamaguchi, Y. Fujita, M. Koyama, A wrapped normal distribution on hyperbolic space for gradient-based learning, in: ICML, Vol. 97, 2019, pp. 4693–4702.
[38] Y. Wang, H. Wang, W. Lu, Y. Yan, HyGGE: hyperbolic graph attention network for reasoning over knowledge graphs, Inform. Sci. 630 (2023) 190–205.
[39] Q. Bai, C. Nie, H. Zhang, D. Zhao, X. Yuan, Hgwavenet: A hyperbolic graph neural network for temporal link prediction, in: ACM WWW, 2023, pp. 523–532.
[40] Z. Leng, S.-C. Wu, M. Saleh, A. Montanaro, H. Yu, Y. Wang, N. Navab, X. Liang, F. Tombari, Dynamic hyperbolic attention network for fine hand-object reconstruction, in: CVPR, 2023, pp. 14894–14904.
[41] S. Liu, J. Chen, L. Pan, C.-W. Ngo, T.-S. Chua, Y.-G. Jiang, Hyperbolic visual embedding learning for zero-shot recognition, in: CVPR, 2020, pp. 9273–9281.
[42] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3D shapenets: A deep representation for volumetric shapes, in: CVPR, 2015, pp. 1912–1920.
[43] J. Sun, Q. Zhang, B. Kailkhura, Z. Yu, C. Xiao, Z.M. Mao, Benchmarking robustness of 3d point cloud recognition against common corruptions, in: ICLR 2022 Workshop on Socially Responsible Machine Learning, SRML, 2022.
[44] M.A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, S.-K. Yeung, Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data, in: ICCV, 2019, pp. 1588–1597.
[45] F. Zhu, S. Ma, Z. Cheng, X.-Y. Zhang, Z. Zhang, C.-L. Liu, Open-world machine learning: A review and new outlooks, 2024, arXiv preprint arXiv:2403.01759.

Yuan-Zhi Feng received the B.S. degree in Automation from Henan University, Kaifeng, China, in 2021, and M.E. degree in electronic information at the School of Electronic, Electrical, and Communication Engineering, University of Chinese Academy of Sciences (UCAS), Beijing, China. He is currently a visiting researcher at Software Engineering Institute, East China Normal University (ECNU). His research interests include hyperbolic geometry, few-shot learning and point cloud classification.

Shing-Ho J. Lin is currently an undergraduate student at School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS), Beijing, China. He was visiting student at Hong Kong University of Science and Technology (HKUST). He participated in joint research with the Department of EE, Tsinghua University. His research interest include Geometric Deep Learning and Optimization. He is student member of IEEE, CAAI.

Xuan Tang is currently an Associate Professor in the School of Communication Electronic Engineering at the East China Normal University, Shanghai, China. She received her B.Eng. and Ph.D. degree in Electronic and Communications Engineering from Northumbria University, Newcastle, UK in 2008 and 2013, respectively. She was a Postdoctoral Researcher at the Department of EE, Tsinghua University from Oct. 2012 to Jul. 2014. Her research interests includes Optical Wireless Communication Systems and RF Communication Technologies. She has led or participated in almost 20 projects. She has published over 60 papers and was invited-speaker of over 10 international conferences. She serves as a reviewer for several high impact journals. She is member of IEEE.

Mu-Yu Wang received the B.S. degree in Software Engineering from Northeastern University, Shenyang, China, in 2016, and the M.S. degree in Software Engineering from the School of Software, Liaoning Technical University, Huludao, China, in 2024. He's currently pursuing for Ph.D. degree at Software Engineering Institute, East China Normal University (ECNU). His research interests include computer vision, laser point cloud and Transformer.

Jian-Zhang Zheng received the B.S. degree in electronic science and technology from Xi'an Jiaotong University, Xi'an, China, in 2018, and the M.E. degree in control engineering from University of Chinese Academy of Sciences (UCAS), Beijing, China in 2021. He is currently working as a research assistant in Fujian Institute of Research on the Structure of Matter, Chinese Academy of Sciences. His research interests include geometric deep learning and machine learning.

Zi-Yao He received B.Eng. degree in Software Engineering from Anhui University, Hefei, China in 2024. He is currently M.S student at Software Engineering Institute, East China Normal University (ECNU). His research interests include point cloud computing and hyperbolic space. He has won the second prize of Robocup China Robot World Cup and the third prize of the National Finals of Robocom Robot Developer Competition.

Zi-Yi Pang is an undergraduate student at the Software Engineering Institute, East China Normal University (ECNU) from 2023. His research interests include LiDAR point cloud and computer vision.

Jian Yang is a lecturer at Information Engineering University, Zhengzhou, China. He received B.S. and M.E. degrees in information and communication engineering from the National University of Defense Technology, Changsha, China, in 2007 and 2009. Then, he went to Germany for doctoral study and received his doctoral degree in cartography and geographic information engineering from the Technical University of Munich, Munich, Germany, in 2016. His research interests include knowledge discovery in crowd sourced geo-spatial data, machine maps and their application in autonomous driving.

Ming-Song Chen received his undergraduate and master's degrees in Computer Science and Technology from Nanjing University in 2003 and 2006, respectively, and his doctoral degree in Computer Science from the University of Florida in 2010. He joined the Software Engineering Institute, East China Normal University(ECNU) in 2010 and obtained the title of professor in 2015. His research interests includes trusted artificial intelligence, IoT technology and automation of information physics fusion system design.

Xian Wei received the Ph.D. degree in Computer Engineering from the Technical University of Munich, Munich, Germany. Currently, he is a research professor at Software Engineering Institute (SEI), East China Normal University(ECNU), Shanghai, China. His research interests focus on sparse coding, deep learning, geometric optimization, and time series analysis. The applications include multi-sensor fusion for intelligent car, robotic vision, data sequence or images modeling, synthesis, recognition and semantics. He has authored over 100 publications in refereed journals and conference proceedings. He is a senior member of IEEE and CCF.