Curvature Generation in Curved Spaces for Few-Shot Learning

Zhi Gao[1], Yuwei Wu[1]*, Yunde Jia[1], Mehrtash Harandi[2]
[1]Beijing Laboratory of Intelligent Information Technology
School of Computer Science, Beijing Institute of Technology, Beijing, China
[2]Department of Electrical and Computer Systems Eng., Monash University, and Data61, Australia
{gaozhi_2017,wuyuwei,jiayunde}@bit.edu.cn, mehrtash.harandi@monash.edu

Abstract

Few-shot learning describes the challenging problem of recognizing samples from unseen classes given very few labeled examples. In many cases, few-shot learning is cast as learning an embedding space that assigns test samples to their corresponding class prototypes. Previous methods assume that data of all few-shot learning tasks comply with a fixed geometrical structure, mostly a Euclidean structure. Questioning this assumption that is clearly difficult to hold in real-world scenarios and incurs distortions to data, we propose to learn a task-aware curved embedding space by making use of the hyperbolic geometry. As a result, task-specific embedding spaces where suitable curvatures are generated to match the characteristics of data are constructed, leading to more generic embedding spaces. We then leverage on intra-class and inter-class context information in the embedding space to generate class prototypes for discriminative classification. We conduct a comprehensive set of experiments on inductive and transductive few-shot learning, demonstrating the benefits of our proposed method over existing embedding methods.

1. Introduction

Few-shot learning (FSL) [2, 12, 57] aims to recognize samples from unseen classes, given very few labeled examples per class. In many cases, embedding methods [5, 44, 45, 53, 55] are the method of choice in addressing the FSL problem. The underlying idea is to learn an embedding space from seen classes, hoping that for new FSL tasks with unseen classes, class prototypes in the embedding space are informative enough to assist classification (*e.g.*, by assigning test samples to the closest prototypes).

Prior to our work, most embedding methods assume that data of all FSL tasks have the Euclidean structure, and use a Euclidean embedding space and corresponding Euclidean

*Corresponding author

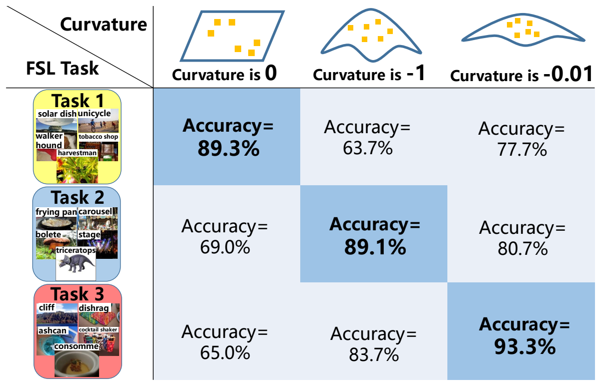

Figure 1. An example to show that a fixed curvature is not suitable for all FSL tasks. We evaluate the curvature as 0, -1, and -0.01 on three FSL tasks sampled from the Mini-ImageNet dataset [50]. In Task 1, we achieve good results when the curvature is 0 (*i.e.*, a Euclidean space), while the curvature 0 achieves inferior results in Task 2 and Task 3, whose suitable curvatures are -1 and -0.01, respectively. This shows that a fixed curvature is not suitable for all FSL tasks.

operations (*e.g.*, distance measure in vector spaces) for classification [3, 35, 38, 55]. However, such an assumption is clearly difficult to hold in real-world scenarios. Although some data (*e.g.*, remote sensing images [15]) is intrinsically Euclidean, non-Euclidean structures are also widely encountered in data (*e.g.*, face images are located in natural manifolds [59]). Thus, assuming the same geometrical structure among all FSL tasks may incur distortions to data and result in inferior performance. In this paper, we propose a curvature generation embedding method that learns a task-aware curved embedding space to match geometrical structures of data by using hyperbolic geometry.

Hyperbolic geometry that defines curved spaces can gracefully represent hierarchies in data and has been shown to be superior to Euclidean geometry for some problems in recent studies [33, 47, 30, 32, 36]. Curvature is a fundamental concept in hyperbolic geometry, representing the deviation of curved spaces from a flat space. Via changing the curvature, one can hope to better capture various geometrical structures of data [29]. A key observation in FSL is that *tasks can exhibit different geometrical structures*. In other words, while a space with a fixed and predetermined curvature might successfully capture the geometrical structure of data in an FSL task, it does not necessarily suit unseen and future tasks (an example is shown in Figure 1).

This motivates us to learn to identify suitable curvatures for FSL tasks. To this end, we need to address two challenges. **(1)** How to generate suitable curvatures using very few samples. In an FSL task, the geometrical structure may be complex and not uniform. When very few samples are given, it is challenging to generate an appropriate curvature that can suitably match the complex structure. **(2)** How to carry out discriminative classification in the task-aware curved embedding space. Sample features in FSL are usually extracted via a common feature extractor (*e.g.*, using a CNN). This makes the features agnostic to the generated curvatures, and hence naively combining them (*e.g.*, averaging) to form class prototypes may hurt the discriminative power of the embedding space.

To address the aforementioned challenges, we make use of the higher-order statistics of samples to identify curvatures for each class in an FSL task. This will result in describing a task with multiple geometries, where data in each class is separately characterized. Clearly, this design is more flexible as compared to using a single geometry everywhere, and may better match complex structures of data. We then benefit from intra-class and inter-class context information in the curved embedding space to produce class prototypes for discriminative classification. Our model is trained in a meta-learning framework that learns knowledge from some seen classes and applies it to new FSL tasks to generate both suitable curved spaces and discriminative prototypes using very few samples. We demonstrate the effectiveness of the proposed method on both inductive and transductive settings of FSL, and compare our method against state-of-the-art embedding methods. The code is available at `https://github.com/ZhiGaomcislab/CurvatureGeneration_FSL`.

In summary, our contributions are two-fold: **(1)** We propose a method to realize a task-aware curved embedding space for FSL. In doing so, our method generates task-specific curvatures, making the embedding space more generic. To the best of our knowledge, this is the first attempt to automatically generate curvatures for different tasks, having the ability to adapt a model to different geometrical structures of data. **(2)** To capture complex structures in data, we generate curvatures for each class and make use of intra-class and inter-class context information to generate class prototypes. This, as we will empirically show, leads to discriminative classification when very few samples are available.

2. Related Work

2.1. Embedding Methods for FSL

Learning an embedding space is an effective way to approach FSL. One can classify existing embedding methods into two groups based on the nature of the embedding, namely, task-agnostic embedding methods and task-specific embedding methods. Early pioneering works such as Prototypical Networks [44], Relation Networks [45], and Matching Networks [50] are task-agnostic embedding methods. They learn a common embedding space, and new samples are classified using the metric in the embedding space. Recent advances, however, suggest that a common and task-agnostic embedding space might not be discriminative enough. Thus, task-specific embedding methods emerge, which seek to adapt the embedding space to specific tasks by various means. This includes learning a task-specific transformation [13, 35, 55, 56], employing a task-specific loss function [21], generating task-specific class prototypes [12, 28, 37, 39, 54], realizing task-specific distances measures [3, 38], or learning task-specific subspaces [43], to name a few. Our work belongs to the group of task-specific embedding methods but goes beyond previous studies (*e.g.*, [28, 43, 55]) as we discard the assumption that data has the same geometrical structure in all FSL tasks. In essence, we propose a curvature generation embedding method that adapts the embedding space to geometrical structures by generating task-specific curvatures.

2.2. Hyperbolic Geometry

Existing methods that exploit hyperbolic geometry can be divided into two categories. The first category establishes deep hyperbolic neural networks. Representative works include the hyperbolic multi-layer perceptrons [10], hyperbolic convolutional neural networks [42], hyperbolic graph convolutional neural networks [6, 29, 58, 7], and hyperbolic attention networks [14]. The second category focuses on learning hyperbolic embeddings. Hyperbolic embeddings have achieved superior performances in various natural language processing problems [8, 14, 47]. Very recently, several studies in computer vision show that hyperbolic embeddings can boost the performance of the model (*e.g.*, image classification [16, 30] and video search [32]). It is argued in both fields that the improvements are due to the fact that the hyperbolic geometry can well capture hierarchies in data [18, 34]. In our work, we also make use of the hyperbolic geometry to realize embedding spaces. In contrast to previous art that considers the curvature of the space fixed and tune it as a hyperparameter, we utilize a meta-learning framework to learn to automatically generate suitable curvatures. We will show that our method is able to quickly adapt a hyperbolic space to geometrical structures of data using few labeled samples.

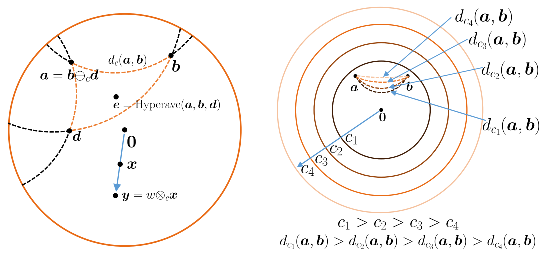
$d_c(\boldsymbol{a},\boldsymbol{b})$
$\boldsymbol{a}=\boldsymbol{b}\oplus_c\boldsymbol{d}$
$\boldsymbol{b}$
$\boldsymbol{e}=\text{Hyperave}(\boldsymbol{a},\boldsymbol{b},\boldsymbol{d})$
$\boldsymbol{d}$
$\mathbf{0}$
$\boldsymbol{x}$
$\boldsymbol{y}=w\otimes_c\boldsymbol{x}$
$d_{c_4}(\boldsymbol{a},\boldsymbol{b})$
$d_{c_3}(\boldsymbol{a},\boldsymbol{b})$
$d_{c_2}(\boldsymbol{a},\boldsymbol{b})$
$d_{c_1}(\boldsymbol{a},\boldsymbol{b})$
$\boldsymbol{a}$
$\boldsymbol{b}$
$\mathbf{0}$
c_1
c_2
c_3
c_4
$c_1 > c_2 > c_3 > c_4$
$d_{c_1}(\boldsymbol{a},\boldsymbol{b}) > d_{c_2}(\boldsymbol{a},\boldsymbol{b}) > d_{c_3}(\boldsymbol{a},\boldsymbol{b}) > d_{c_4}(\boldsymbol{a},\boldsymbol{b})$

Figure 2. Left: illustration of a 2-D Poincaré ball. Dash lines represent geodesics, the shortest curves connecting two vectors in the poincaré ball. Right: Poincaré balls with different curvatures. As the curvature ($-c$) increases, the distance between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ decreases.

3. Preliminaries of Hyperbolic Geometry

Hyperbolic space. A d-dimensional hyperbolic space is a smooth curved space with a negative curvature [19]. It has five isometric models: the Hyperboloid model, the Klein model, the Hemisphere model, the Poincaré ball model, and the Poincaré half-space model [33]. In this paper, we choose the Poincaré ball to model the embedding space, as shown in Figure 2, and utilize the framework of Möbius gyrovector space [49] to provide operations for the Poincaré ball. A Poincaré ball is defined as $\mathbb{D}_c^d = \{\boldsymbol{x} \in \mathbb{R}^d, c\|\boldsymbol{x}\| < 1\}$, where $\|\cdot\|$ is the Euclidean norm. The parameter $c > 0$ is essential in our work and determines the curvature of $\mathbb{D}_c^d$. Note that, this definition of $\mathbb{D}_c^d$ covers the Euclidean space and hyperbolic spaces with different curvatures. If $c \to 0$, then $\mathbb{D}_c^d$ becomes the d-dimensional Euclidean space $\mathbb{R}^d$; if $c > 0$, then $\mathbb{D}_c^d$ is an open ball with the curvature being $-c$. For a point $\boldsymbol{x} \in \mathbb{D}_c^d$ on the Poincaré ball, the tangent space, denoted by $T_{\boldsymbol{x}}\mathbb{D}_c^d$, is a Euclidean space, containing all vectors tangent to $\mathbb{D}_c^d$ at $\boldsymbol{x}$.

Möbius addition. The addition of two vectors $\boldsymbol{x}, \boldsymbol{u} \in \mathbb{D}_c^d$ is defined by the Möbius addition,

$$\boldsymbol{x}\oplus_c\boldsymbol{u} = \frac{(1+2c\langle\boldsymbol{x},\boldsymbol{u}\rangle + c\|\boldsymbol{u}\|^2)\boldsymbol{x} + (1-c\|\boldsymbol{x}\|^2)\boldsymbol{u}}{1+2c\langle\boldsymbol{x},\boldsymbol{u}\rangle + c^2\|\boldsymbol{x}\|^2\|\boldsymbol{u}\|^2}, \quad (1)$$

where $\langle\cdot\rangle$ denotes the Euclidean inner product.

Möbius scalar multiplication. The scalar multiplication of a vector $\boldsymbol{x} \in \mathbb{D}_c^d$ by a scalar $w \in \mathbb{R}$ is defined by the Möbius scalar multiplication,

$$w\otimes_c\boldsymbol{x} = \frac{1}{\sqrt{c}}\tanh\left(w \cdot \operatorname{arctanh}(\sqrt{c}\|\boldsymbol{x}\|)\right)\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}. \quad (2)$$

Hyperbolic averaging. Given a set $\boldsymbol{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_m\}$, $\boldsymbol{x}_i \in \mathbb{D}_c^d$ and $i \in [1, m]$, we use the Einstein midpoint [48] to compute the mean of $\boldsymbol{X}$. Specifically, we first project vectors from the Poincaré ball $\mathbb{D}_c^d$ to the Klein model $\mathbb{K}_c^d$, then we compute the mean on the Klein model, and finally we project the mean back to the Poincaré ball, given by

$$\overline{\boldsymbol{x}} = \text{Hyperave}_{\boldsymbol{x}_i\in\boldsymbol{X}}(\boldsymbol{x}_i) = \begin{cases} \boldsymbol{u}_i = \dfrac{2\boldsymbol{x}_i}{1+c\|\boldsymbol{x}_i\|^2} \\ \overline{\boldsymbol{u}} = \dfrac{\sum_{i=1}^m \gamma_i\boldsymbol{u}_i}{\sum_{i=1}^m \gamma_i} \\ \overline{\boldsymbol{x}} = \dfrac{\overline{\boldsymbol{u}}}{1+\sqrt{1-c\|\overline{\boldsymbol{u}}\|^2}} \end{cases}, \quad (3)$$

where $\boldsymbol{u}_i \in \mathbb{K}_c^d$, $\overline{\boldsymbol{u}}$ is the mean on $\mathbb{K}_c^d$, $\overline{\boldsymbol{x}}$ is the mean on $\mathbb{D}_c^d$, and $\gamma_i = \frac{1}{\sqrt{1-c\|\boldsymbol{u}_i\|^2}}$ is the Lorentz factor.

Distance measure. The distance between two vectors $\boldsymbol{x}, \boldsymbol{u} \in \mathbb{D}_c^d$ is

$$d_c(\boldsymbol{x}, \boldsymbol{u}) = \frac{2}{\sqrt{c}}\operatorname{arctanh}(\sqrt{c}\|-\boldsymbol{x}\oplus_c\boldsymbol{u}\|). \quad (4)$$

When $c \to 0$, $d_c(\boldsymbol{x}, \boldsymbol{u})$ becomes proportional to the Euclidean distance, that is, $\lim_{c\to 0} d_c(\boldsymbol{x}, \boldsymbol{u}) = 2\|\boldsymbol{x} - \boldsymbol{u}\|$.

Exponential map. The exponential map $\exp_{\boldsymbol{x}}^c$ maps a vector $\boldsymbol{v}$ from the tangent space $T_{\boldsymbol{x}}\mathbb{D}_c^d$ to the Poincaré ball $\mathbb{D}_c^d$. In this work, we use a neural network to attain features in the tangent space $T_{\mathbf{0}}\mathbb{D}_c^d$ at the origin $\mathbf{0}$ in a Poincaré ball and then use $\exp_{\mathbf{0}}^c$ to obtain embeddings in the Poincaré ball,

$$\exp_{\mathbf{0}}^c(\boldsymbol{v}) = \tanh\left(\sqrt{c}\|\boldsymbol{v}\|\right)\frac{\boldsymbol{v}}{\sqrt{c}\|\boldsymbol{v}\|}. \quad (5)$$

4. Curvature Generation Embedding Method

4.1. Framework

An FSL task (*i.e.*, an episode) is denoted as a k-shot n-way classification problem that contains a support set and a query set. A support set $\mathcal{S} = \{\boldsymbol{I}_{s,i}, y_{s,i}\}_{i=1}^{kn}$ consists of k labeled samples $\boldsymbol{I}_{s,i}$ for each of n classes, $y_{s,i}$ is the label, and $\mathcal{C}_j$ contains samples of the j-th class. A query set $\mathcal{Q} = \{\boldsymbol{I}_{q,i}\}_{i=1}^{en}$ consists of e test samples in each class, and their ground truth are $\{y_{q,i}\}_{i=1}^{en}$. FSL methods usually sample k-shot n-way FSL tasks from some seen classes with sufficient labeled samples to train the model, and their performance is evaluated on FSL tasks with unseen classes.

This paper proposes an embedding method that employs the Poincaré ball to model a task-aware curved embedding space (see Figure 3 for a conceptual illustration). Our method contains a feature extractor $f_\theta(\cdot)$, a class-curvature generator (CCG) $g_\phi(\cdot)$, and a hyperbolic aggregation network (HAN) $h_\psi(\cdot)$, with θ, ϕ, and ψ denoting their parameters. Given an FSL task, our method first extracts features of samples, $f_\theta(\boldsymbol{I}_{s,i}), f_\theta(\boldsymbol{I}_{q,i}) \in \mathbb{R}^d$. Then, our method generates a curvature for each class using the class-curvature generator (*i.e.*, $c_j = g_\phi(\boldsymbol{D}_j)$) and aggregates samples into discriminative class prototypes for the adapted embedding space via the hyperbolic aggregation network as $\boldsymbol{p}_j = h_\psi(\boldsymbol{D}_j) \in \mathbb{D}_{c_j}^d$. Here, $j \in [1, n]$, and c_j and $\boldsymbol{p}_j$ show the curvature and prototype of the j-th class. $\boldsymbol{D}_j$ is

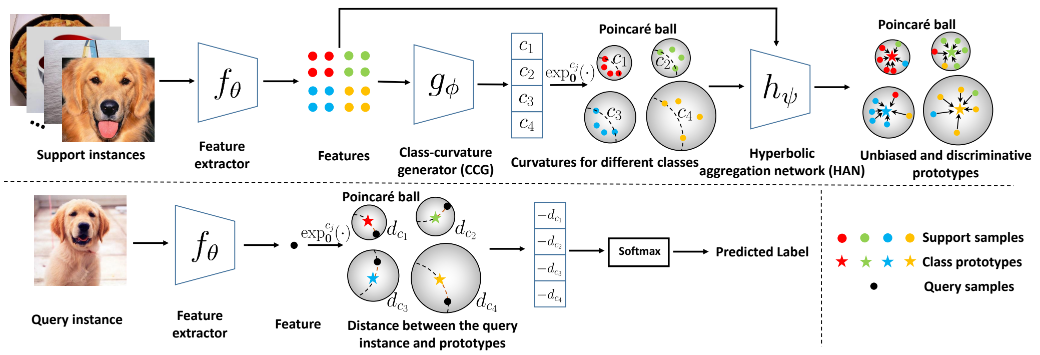

Figure 3. A conceptual diagram of our method. Given an FSL task, we generate a curvature and a discriminative prototype for each class. We then map query samples to the embedding space and calculate distances between them and class prototypes for classification.

a set containing in-class and out-of-class samples for the j-th class and will be defined shortly. Finally, we map query samples $f_\theta(\boldsymbol{I}_{q,i})$ to the embedding space via $\exp_{\mathbf{0}}^{c_j}$ and compute distances between them and class prototypes using the class curvatures for classification.

Suppose $\widehat{y}_{q,i}$ is the prediction of our method for a query sample $\boldsymbol{I}_{q,i}$, we train our model by minimizing the following objective,

$$\mathcal{L}(\theta,\phi,\psi) = -\sum_t \frac{1}{|\mathcal{Q}_t|} \sum_{\boldsymbol{I}_{q,i}\in\mathcal{Q}_t} \log p(\widehat{y}_{q,i} = y_{q,i}|\boldsymbol{I}_{q,i}), \tag{6}$$

where $\mathcal{Q}_t$ is the query set of the t-th FSL task sampled from seen classes, and $p(\widehat{y}_{q,i} = y_{q,i}|\boldsymbol{I}_{q,i})$ is the probability that $\boldsymbol{I}_{q,i}$ belongs to its ground truth class. In this paper, we formulate $p(\hat{y}_{q,i} = j|\boldsymbol{I}_{q,i})$ as

$$\begin{aligned} p(y_{q,i} = j|\boldsymbol{I}_{q,i}) &= \frac{\exp\Big(-d_{c_j}\Big(\exp_{\mathbf{0}}^{c_j}\big(f_\theta(\boldsymbol{I}_{q,i})\big), \boldsymbol{p}_j\Big)\Big)}{\sum_{j'}\exp\Big(-d_{c_{j'}}\Big(\exp_{\mathbf{0}}^{c_{j'}}\big(f_\theta(\boldsymbol{I}_{q,i})\big), \boldsymbol{p}_{j'}\Big)\Big)} \\ &= \frac{\exp\Big(-d_{g_\phi(\boldsymbol{D}_j)}\Big(\exp_{\mathbf{0}}^{g_\phi(\boldsymbol{D}_j)}\big(f_\theta(\boldsymbol{I}_{q,i})\big), h_\psi(\boldsymbol{D}_j)\Big)\Big)}{\sum_{j'}\exp\Big(-d_{g_\phi(\boldsymbol{D}_{j'})}\Big(\exp_{\mathbf{0}}^{g_\phi(\boldsymbol{D}_{j'})}\big(f_\theta(\boldsymbol{I}_{q,i})\big), h_\psi(\boldsymbol{D}_{j'})\Big)\Big)}. \end{aligned} \tag{7}$$

$\boldsymbol{D}_j = [\boldsymbol{X}_j, \boldsymbol{Z}_j]$ contains an in-class set $\boldsymbol{X}_j$ and an out-of-class set $\boldsymbol{Z}_j$. Note that, the proposed method can be applied to both the inductive and transductive settings of FSL. In the inductive setting, as the query set is not available, we choose $\boldsymbol{X}_j = \{f_\theta(\boldsymbol{I}_{s,i})|\boldsymbol{I}_{s,i}\in\mathcal{C}_j\}$ and $\boldsymbol{Z}_j = \{f_\theta(\boldsymbol{I}_{s,i})|\boldsymbol{I}_{s,i}\in\mathcal{S}\backslash\mathcal{C}_j\}$, where $|\boldsymbol{X}_j| = k$ and $|\boldsymbol{Z}_j| = k(n-1)$. In the transductive setting, both the support and query sets are available, and thus we set $\boldsymbol{X}_j = \{f_\theta(\boldsymbol{I}_{s,i})|\boldsymbol{I}_{s,i}\in\mathcal{C}_j\}$, $\boldsymbol{Z}_j = \{f_\theta(\boldsymbol{I}_{q,i})|\boldsymbol{I}_{q,i}\in\mathcal{Q}\}\cup\{f_\theta(\boldsymbol{I}_{s,i})|\boldsymbol{I}_{s,i}\in\mathcal{S}\backslash\mathcal{C}_j\}$, and $|\boldsymbol{Z}_j| = k(n-1)+en$. In following sections, we will detail out the class-curvature generator $g_\phi(\cdot)$ and the hyperbolic aggregation network $h_\psi(\cdot)$.

4.2. Class-curvature Generator (CCG)

In the class-curvature generator, the curvature c_j for the j-th class is generated based on the second-order statistics of $\boldsymbol{D}_j = [\boldsymbol{X}_j, \boldsymbol{Z}_j]$, which has an ability to capture expressive correlations between features of given samples [11, 26]. Specifically, we denote the second-order statistics of $\boldsymbol{D}_j$ as $\boldsymbol{b}_j\in\mathbb{R}^\varrho$ (ϱ is a hyperparameter of the class-curvature generator), and its l-th element b_j^l can be computed via the factorized bilinear model [24, 46]:

$$b_j^l = \sum_{\boldsymbol{x}_j^i\in\boldsymbol{X}_j}\sum_{\boldsymbol{z}_j^{i'}\in\boldsymbol{Z}_j} \mathbb{1}^\top(\boldsymbol{U}_l^\top\boldsymbol{x}_j^i \circ \boldsymbol{V}_l^\top\boldsymbol{z}_j^{i'}), \tag{8}$$

where $\boldsymbol{x}_j^i\in\mathbb{R}^d$, $\boldsymbol{z}_j^{i'}\in\mathbb{R}^d$, $\boldsymbol{U}_l\in\mathbb{R}^{d\times r}$, $\boldsymbol{V}_l\in\mathbb{R}^{d\times r}$, and $\mathbb{1}\in\mathbb{R}^r$ is a vector whose elements are all '1's. r is the other hyperparameter of the class-curvature generator denoting the rank of $\boldsymbol{U}_l$ and $\boldsymbol{V}_l$, and $\circ$ denotes the Hadamard product. In this case, we can directly compute $\boldsymbol{b}_j$ as

$$\boldsymbol{b}_j = \sum_{\boldsymbol{x}_j^i\in\boldsymbol{X}_j}\sum_{\boldsymbol{z}_j^{i'}\in\boldsymbol{Z}_j} \boldsymbol{O}(\boldsymbol{U}^\top\boldsymbol{x}_j^i \circ \boldsymbol{V}^\top\boldsymbol{z}_j^{i'}), \tag{9}$$

where $\boldsymbol{U} = [\boldsymbol{U}_1,\cdots,\boldsymbol{U}_\varrho]\in\mathbb{R}^{d\times r\varrho}$, $\boldsymbol{V} = [\boldsymbol{V}_1,\cdots,\boldsymbol{V}_\varrho]\in\mathbb{R}^{d\times r\varrho}$, and $\boldsymbol{O}\in\mathbb{R}^{\varrho\times r\varrho}$ is a fixed binary matrix with elements in row l, columns $((l-1)\times r+1)$ to (lr) being "1" for $l\in[1,\varrho]$.

We empirically observed that arbitrary curvatures may cause numerical instabilities in the training process. Besides, some methods have shown that setting c in the range of $[0,1]$ is suitable in many cases [16, 32]. Thus, we use a multi-layer perceptron (MLP) network MLP_1 and a sigmoid function to generate c_j:

$$c_j = \mathrm{sigmoid}\big(\mathrm{MLP}_1(\boldsymbol{b}_j)\big), \tag{10}$$

where the parameter of MLP_1 is $\boldsymbol{W}_{f1}$, and the sigmoid function is a regularization manner towards efficient training. All in all, parameters of the class-curvature generator are $\phi = \{\boldsymbol{U},\boldsymbol{V},\boldsymbol{W}_{f1}\}$.

4.3. Hyperbolic Aggregation Network (HAN)

We consider context information of the task-aware curved embedding space to generate weights for samples

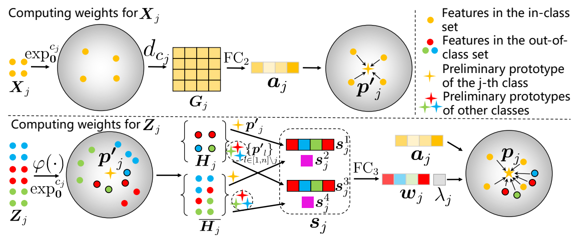
Computing weights for X_j
$\exp_0^{c_j}$
X_j
d_{c_j}
G_j
FC$_2$
a_j
p'_j
Features in the in-class set
Features in the out-of-class set
Preliminary prototype of the j-th class
Preliminary prototypes of other classes
Computing weights for Z_j
$\varphi(\cdot)$
$\exp_0^{c_j}$
Z_j
p'_j
H_j
$\overline{H_j}$
$\{p'_l\}_{l\in[1,n]\setminus j}$
s_j^1
s_j^2
s_j^3
s_j^4
s_j
FC$_3$
a_j
w_j
λ_j
p_j

Figure 4. The illustration of the HAN. We first compute weights a_j for samples in the in-class set X_j, and aggregate them into a preliminary prototype p'_j. We then collect H_j and $\overline{H_j}$ from the out-of-class set Z_j, and compute weights w_j and trade-off λ_j for H_j and X_i to obtain a prototype p_j.

in $D_j = [X_j, Z_j]$ and aggregate them into discriminative prototypes p_j, where the context information is modeled as intra-class and inter-class distances in the embedding space. We first compute weights for samples in X_j. An intra-class distance matrix G_j is calculated using the curvature c_j of the j-th class,

$$G_j^{ii'} = d_{c_j}\left(\exp_{\mathbf{0}}^{c_j}(x_j^i), \exp_{\mathbf{0}}^{c_j}(x_j^{i'})\right), \tag{11}$$

where $x_j^i, x_j^{i'} \in X_j$, and $G_j^{ii'}$ is the element at the i-th raw and i'-th column of the matrix $G_j \in \mathbb{R}^{k\times k}$. We use an MLP network MLP$_2$ to compute the weights $a_j \in \mathbb{R}^k$ for X_j,

$$a_j = \mathrm{MLP}_2(G_j), \tag{12}$$

where the parameter of MLP$_2$ is W_{f2}. Based on a_j, we compute a preliminary prototype $p'_j \in \mathbb{D}_{c_j}^d$ by

$$p'_j = \mathrm{Hyperave}_{x_j^i \in X_j}\left(a_j^i \otimes_{c_j} \exp_{\mathbf{0}}^{c_j}(x_j^i)\right), \tag{13}$$

where a_j^i is the i-th element of a_j. Preliminary prototypes of all classes are denoted as $P = [p'_1, \cdots, p'_n]$. Eq. (13) has a complex formulation with the hyperbolic averaging operation, the exponential map, and the Möbius scalar multiplication. Based on Lemma 1, we can rewrite Eq. (13) into a simpler formulation:

$$p'_j = \mathrm{Hyperave}_{x_j^i \in X_j}\left(\exp_{\mathbf{0}}^{c_j}(a_j^i x_j^i)\right). \tag{14}$$

Lemma 1. *In a Poincaré ball $\mathbb{D}_c^d$, the exponential map and the scalar multiplication in the tangent space $T_{\mathbf{0}}\mathbb{D}_c^d$ at the origin satisfy the commutative law, that is*

$$w \otimes_c \exp_{\mathbf{0}}^c(x) = \exp_{\mathbf{0}}^c(wx), \tag{15}$$

where $w \in \mathbb{R}$ is a scalar and $x \in T_{\mathbf{0}}\mathbb{D}_c^d$.

Proof. The proof is in the supplementary materials. □

Next, we compute weights for samples in Z_j. This is done by only considering the m closest samples to p'_j in Z_j. To this end, we first map samples in Z_j to task-specific embeddings via a projection $\varphi(\cdot)$ parameterized by W_φ and the exponential map $\exp_{\mathbf{0}}^{c_j}$. We then collect the m samples closest to p'_j into a set $H_j = \{\exp_{\mathbf{0}}^{c_j}(\varphi(z_j^i)) | \exp_{\mathbf{0}}^{c_j}(\varphi(z_j^i)) \in N_m(p'_j), z_j^i \in Z_j\}$, where $|H_j| = m$, $N_m(p'_j)$ contains m closest embeddings to p'_j, and the i-th embedding in H_j is $h_j^i \in \mathbb{D}_{c_j}^d$. The remaining samples constitute a set $\overline{H_j} = \{\exp_{\mathbf{0}}^{c_j}(\varphi(z_j^i)) | \exp_{\mathbf{0}}^{c_j}(\varphi(z_j^i)) \notin N_m(p'_j), z_j^i \in Z_j\}$, where $|\overline{H_j}| = |Z_j| - m$, with the i-th embedding in $\overline{H_j}$ being $\overline{h_j^i} \in \mathbb{D}_{c_j}^d$. We use another MLP network MLP$_3$ (parameterized by W_{f3}) to compute weights for the collected m samples in Z_j. The input (denoted by dis_j) to MLP$_3$ is composed of inter-class distances according to: (1) distances between p'_j and embeddings in H_j, denoted by dis_j^1; (2) the mean of distances between p'_j and embeddings in $\overline{H_j}$, denoted by dis_j^2; (3) the mean of distances between other preliminary prototypes $P \backslash p'_j$ and embeddings in H_j, denoted by dis_j^3; (4) the mean of distances between other preliminary prototypes $P \backslash p'_j$ and embeddings in $\overline{H_j}$, denoted by dis_j^4,

$$\begin{aligned}
dis_j^1 =& [d_{c_j}(p'_j, h_j^1), \cdots, d_{c_j}(p'_j, h_j^m)] \in \mathbb{R}^m, \\
dis_j^2 =& \frac{1}{|Z_j| - m} \sum_{i=1}^{|Z_j|-m} d_{c_j}(p'_j, \overline{h_j^i}) \in \mathbb{R}, \\
dis_j^3 =& [\frac{1}{n-1} \sum_{l\in([1,n]\setminus j)} d_{c_l}(p'_l, h_j^1), \cdots, \\
& \frac{1}{n-1} \sum_{l\in([1,n]\setminus j)} d_{c_l}(p'_l, h_j^m)] \in \mathbb{R}^m, \\
dis_j^4 =& \frac{1}{(n-1)\times(|Z_j| - m)} \sum_{l\in([1,n]\setminus j)} \sum_{i=1}^{|Z_j|-m} d_{c_l}(p'_l, \overline{h_j^i}) \in \mathbb{R}, \\
dis_j =& [dis_j^1, dis_j^2, dis_j^3, dis_j^4].
\end{aligned} \tag{16}$$

We feed dis_j to MLP$_3$ to obtain

$$w_j, \lambda_j = \mathrm{MLP}_3(dis_j), \tag{17}$$

$w_j \in \mathbb{R}^m$ is the weight for the collected m samples in Z_j and λ_j is the trade-off between X_j and Z_j. The prototype $p_j \in \mathbb{D}_{c_j}^d$ will be computed using a_j, w_j, and λ_j as

$$\begin{cases}
T_j =& \left\{\exp_{\mathbf{0}}^{c_j}(\lambda_j a_j^i x_j^i) | x_j^i \in X_j\right\} \\
& \cup \left\{(1-\lambda_j) w_j^i \otimes_{c_j} h_j^i | h_j^i \in H_j\right\} \\
p_j =& \mathrm{Hyperave}_{t_j^i \in T_j}(t_j^i)
\end{cases}. \tag{18}$$

In (18), w_j^i is the i-th element of w_j. Overall, the framework of the hyperbolic aggregation network is shown in Figure 4, and parameters are $\psi = [W_{f2}, W_\varphi, W_{f3}]$. The training process of our method is shown in Algorithm 1.

Algorithm 1 Training process of the proposed method.

```
Input: Seen classes. Initial feature extractor $f_\theta$, class-curvature generator $g_\phi$, and hyperbolic aggregation network $h_\psi$.
Output: Updated $f_\theta$, $g_\phi$, and $h_\psi$.
1: while $t <$ MaxIteration do
2:     Sample a $k$-shot $n$-way FSL task from seen classes with a support set $\mathcal{S}_t$ and a query set $\mathcal{Q}_t$.
3:     Extract features $f_\theta(\boldsymbol{I}_{s,i})$ and $f_\theta(\boldsymbol{I}_{q,i})$ for $\boldsymbol{I}_{s,i} \in \mathcal{S}_t, \boldsymbol{I}_{q,i} \in \mathcal{Q}_t$.
4:     Construct data sets $\boldsymbol{D}_j = [\boldsymbol{X}_j, \boldsymbol{Z}_j]$ for each class.
5:     Generate curvatures $c_j$ for each class by the class-curvature generator $g_\phi$ via Eq. (9) and Eq. (10).
6:     Generate class prototypes $\boldsymbol{p}_j$ by the hyperbolic aggregation network $h_\psi$. 1. compute weights for samples in $\boldsymbol{X}_j$ via Eq. (11), Eq. (12), Eq. (14). 2. compute weights for samples in $\boldsymbol{Z}_j$ via Eq. (16), Eq. (17). 3. aggregate instances with the weights via Eq. (18).
7:     Classify samples in $\mathcal{Q}_t$ via Eq. (7).
8:     Update $f_\theta$, $g_\phi$, and $h_\psi$ via minimizing Eq. (6).
9: end while
10: Return $f_\theta$, $g_\phi$, and $h_\psi$.
```

5. Experiments

5.1. Experimental Settings

Datasets. We conducted experiments on four popular datasets: mini-ImageNet [50], tiered-ImageNet [40], CUB [51], and CIFAR-FS [4]. We used the standard protocol to process and divide data for training, validation, and testing. Details are in the supplementary materials.

Backbones. For fair and comprehensive comparisons with existing methods, three backbones were used: a 4-layer convolutional network (ConvNet) [22, 45, 50], a 12-layer residual network (ResNet12) [35, 38, 39], and a bigger 12-layer residual network (we denote it as BigResNet12) [20, 43, 55]. ResNet12 and BigResNet12 both have four residual blocks, while the differences is that the numbers of convolutional channels in the four blocks are $(64, 128, 256, 512)$ for ResNet12 and $(64, 160, 320, 640)$ for BigResNet12. Details are in the supplementary materials.

Training Details. Following [35, 55], we pre-trained ResNet12 and BigResNet12 on the training set. Then, we carried out meta-training to learn our method over 200 epochs, and each epoch has 100 episodes. The validation set was only used to select the model after the meta-training stage, and performance was reported as the mean accuracy on the test set with the 95% confidence interval. Other details are in the supplementary materials.

5.2. Main Results

We compare our method with state-of-the-art FSL methods in the inductive and transductive settings on the mini-ImageNet, tiered-ImageNet, CUB, and CIFAR-FS datasets, where 1-shot/5-shot 5-way classification were implemented

Setting	Method	Backbone	1-shot	5-shot
Inductive	MatchingNet [50]	ConvNet	43.56 ± 0.84	55.31 ± 0.73
	ProtoNet [44]	ConvNet	49.42 ± 0.78	68.20 ± 0.66
	RelationNet [45]	ConvNet	50.44 ± 0.82	65.32 ± 0.70
	MMN [5]	ConvNet	53.37 ± 0.48	66.97 ± 0.35
	DSN [43]	ConvNet	51.78 ± 0.96	68.99 ± 0.69
	Afrasiyabi *et al.* [1]	ConvNet	53.14 ± 1.06	71.45 ± 0.72
	FEAT [55]	ConvNet	55.15	71.61
	Ours	ConvNet	$\mathbf{55.53 \pm 0.20}$	$\mathbf{72.12 \pm 0.16}$
	ProtoNet [44]	ResNet12	56.52 ± 0.45	74.28 ± 0.20
	TADAM [35]	ResNet12	58.50 ± 0.30	76.70 ± 0.38
	TapNet [56]	ResNet12	61.65 ± 0.15	76.36 ± 0.10
	DC [25]	ResNet12	62.53 ± 0.19	78.95 ± 0.13
	ECMSFMT [39]	ResNet12	59.00	77.46
	Ours	ResNet12	$\mathbf{63.56 \pm 0.20}$	$\mathbf{79.13 \pm 0.14}$
	MetaOptNet [20]	BigResNet12	62.64 ± 0.61	78.63 ± 0.46
	Net-Cosine [27]	BigResNet12	61.72 ± 0.81	81.79 ± 0.55
	DSN [43]	BigResNet12	62.64 ± 0.66	78.83 ± 0.45
	Ours	BigResNet12	$\mathbf{67.02 \pm 0.20}$	$\mathbf{82.32 \pm 0.14}$
Transductive	TPN [31]	ConvNet	55.51	69.86
	TEAM [38]	ConvNet	56.57	72.04
	DSN [43]	ConvNet	55.88 ± 0.90	70.50 ± 0.68
	FEAT [55]	ConvNet	57.04	72.89
	Ours	ConvNet	$\mathbf{58.29 \pm 0.22}$	$\mathbf{73.93 \pm 0.16}$
	TEAM [38]	ResNet12	60.07	75.90
	Ours	ResNet12	$\mathbf{66.73 \pm 0.22}$	$\mathbf{80.57 \pm 0.14}$
	DSN [43]	BigResNet12	64.60 ± 0.72	79.51 ± 0.50
	Ours	BigResNet12	$\mathbf{71.79 \pm 0.23}$	$\mathbf{83.00 \pm 0.17}$

Table 1. Accuracy (%) comparisons with the state-of-the-art few-shot classification results on the mini-ImageNet dataset.

Setting	Method	1-shot	5-shot
Inductive	MAML [9]	51.67 ± 1.81	70.30 ± 1.75
	ProtoNet [44]	53.51 ± 0.89	72.69 ± 0.74
	RelationNet [45]	54.48 ± 0.93	71.32 ± 0.78
	MetaOptNet [20]	65.99 ± 0.72	81.56 ± 0.63
	CTM [23]	68.41 ± 0.39	84.28 ± 1.73
	SimpleShot [52]	69.09 ± 0.22	84.58 ± 0.16
	TapNet [56]	63.08 ± 0.15	80.26 ± 0.12
	DSN [43]	66.22 ± 0.75	82.79 ± 0.48
	FEAT [55]	70.80 ± 0.23	84.79 ± 0.16
	Ours	$\mathbf{71.66 \pm 0.23}$	$\mathbf{85.50 \pm 0.15}$
Transductive	TPN [31]	59.91 ± 0.94	73.30 ± 0.75
	DSN [43]	67.39 ± 0.82	82.85 ± 0.56
	Ours	$\mathbf{77.19 \pm 0.24}$	$\mathbf{86.18 \pm 0.15}$

Table 2. Accuracy (%) comparisons with the state-of-the-art few-shot classification results on the tiered-ImageNet dataset.

and query sets had 15 samples per class. For fair comparisons, we used ConvNet, ResNet12, and BigResNet12 on the mini-ImageNet dataset. We used BigResNet12 on tiered-ImageNet and CIFAR-FS, and ConvNet on the CUB dataset. Results on the four datasets are shown in Table 1, Table 2, Table 3, and Table 4, respectively.

On the mini-ImageNet dataset, our method achieves the state-of-the-art performance no matter which backbone is used, especially in the transductive setting. TADAM [35], TEAM [38], DSN [43], Afrasiyabi *et al.* [1], Net-Cosine [27], BD-CSPN [28], and FEAT [55] are state-of-the-art embedding methods while using a fixed and unitary Euclidean geometry. Compared with them, our method has better performance, demonstrating that adapting curvatures to geometrical structures of data can lead to better embeddings. On the tiered-ImageNet, CUB, and CIFAR-FS datasets, our method again achieves the state-of-the-art performance. For example, in the transductive setting of

Setting	Method	1-shot	5-shot
Inductive	MAML [9]	55.92 ± 0.95	72.09 ± 0.76
	ProtoNet [44]	51.31 ± 0.91	70.77 ± 0.69
	RelationNet [45]	62.45 ± 0.98	76.11 ± 0.69
	MatchNet [50]	61.16 ± 0.89	72.86 ± 0.70
	Afrasiyabi *et al.* [1]	63.30 ± 0.94	81.35 ± 0.67
	TEAM [38]	69.39	82.78
	FEAT [55]	68.87 ± 0.22	82.90 ± 0.15
	Ours	$\mathbf{74.66 + 0.21}$	$\mathbf{88.37 \pm 0.12}$
Transductive	EPNet [41]	65.94 ± 0.93	78.80 ± 0.64
	TEAM [38]	75.71	86.04
	BD-CSPN [28]	75.10	87.25
	Ours	$\mathbf{76.69 \pm 0.21}$	$\mathbf{89.30 \pm 0.12}$

Table 3. Accuracy (%) comparisons with the state-of-the-art few-shot classification results on the CUB dataset.

Setting	Method	1-shot	5-shot
Inductive	ProtoNets [44]	72.2 ± 0.7	83.5 ± 0.5
	MetaOpt-RR [20]	72.6 ± 0.7	84.3 ± 0.5
	MetaOpt-SVM [20]	72.0 ± 0.7	84.2 ± 0.5
	DSN [43]	72.3 ± 0.8	85.1 ± 0.6
	Ours	$\mathbf{73.0 \pm 0.7}$	$\mathbf{85.8 \pm 0.5}$
Transductive	DSN [43]	75.6 ± 0.9	86.2 ± 0.6
	Ours	$\mathbf{76.8 \pm 0.7}$	$\mathbf{86.4 \pm 0.5}$

Table 4. Accuracy (%) comparisons with the state-of-the-art few-shot classification results on the CIFAR-FS dataset.

Setting	Method	1-shot	5-shot
Inductive	ProtoNet	58.34 ± 0.20	78.49 ± 0.14
	w/o CCG HAN, $c = 1$	59.05 ± 0.21	78.34 ± 0.22
	w/o CCG HAN, $c = 0.01$	50.92 ± 0.22	75.66 ± 0.16
	w/o HAN, single c	58.97 ± 0.20	80.19 ± 0.14
	w/o HAN, class-level c	59.47 ± 0.20	80.41 ± 0.14
	w/o CCG, $c = 1$	62.60 ± 0.20	79.25 ± 0.14
	w/o CCG, $c = 0.01$	64.17 ± 0.21	76.49 ± 0.16
	Ours	$\mathbf{67.02 \pm 0.20}$	$\mathbf{82.32 \pm 0.14}$
Transductive	w/o CCG HAN, $c = 1$	59.14 ± 0.22	79.82 ± 0.21
	w/o CCG HAN, $c = 0.01$	51.06 ± 0.22	75.73 ± 0.16
	w/o HAN, single c	59.16 ± 0.20	80.29 ± 0.14
	w/o HAN, class-level c	59.94 ± 0.21	80.67 ± 0.14
	w/o CCG, $c = 1$	62.80 ± 0.20	81.02 ± 0.14
	w/o CCG, $c = 0.01$	62.50 ± 0.21	77.59 ± 0.16
	model capacity	68.25 ± 0.21	81.08 ± 0.16
	Ours	$\mathbf{71.79 \pm 0.23}$	$\mathbf{83.00 \pm 0.17}$

Table 5. Ablation experiments on the mini-ImageNet dataset.

tiered-ImageNet, we achieve 77.19% and 86.18% on the 1-shot and 5-shot tasks, 9.80% and 3.33% higher than existing methods. In the inductive setting of CUB, our method achieves 74.66% and 88.37%, 5.27% and 5.47% higher than existing methods.

5.3. Ablation Study

We conducted ablation experiments on the mini-ImageNet dataset to evaluate our class-curvature generator (CCG) and hyperbolic aggregation network (HAN). We first removed the CCG and HAN, manually set c as 1, or 0.01, and computed the class prototypes by averaging samples, denoted by 'w/o CCG HAN, c=1/0.01'. Then, we added the CCG to the model. We evaluated generating a single curvature for all classes, denoted by 'w/o HAN, single c', and evaluated generating curvatures for each class, denoted by 'w/o HAN, class-level c'. Finally, we added the HAN while removing the CCG, and c was set manually as 1, or 0.01, denoted by 'w/o CCG, c=1/0.01'. Besides, we removed 9×10^5 parameters from BigResNet12, to keep the number of our whole parameters consistent with the backbone, denoted by 'model capacity'. Our target is to show whether our improvement is from the model capacity. Results are shown in Table 5. Comparing 'w/o CCG HAN, c=1/0.01' with 'w/o HAN' and comparing 'w/o CCG, c=1/0.01' with 'Ours', we can find that generating appropriate curvatures leads to better performance than manually setting a fixed and unitary curvature for various FSL tasks. Besides, multiple class-level curvatures has better performance than a single curvature for all classes, better matching complex data structures. Comparing 'w/o CCG HAN, c=1/0.01' with 'w/o CCG, c=1/0.01', the HAN can lead to discriminative class prototypes even 1 samples are available. The performance of 'model capacity' still higher than compared methods and other settings in Table 5, showing that our improvement is not from the model capacity, but the adaptive curvatures.

Setting	Method	1-shot
Inductive	$m = 1$	65.03 ± 0.20
	$m = 2$	65.82 ± 0.20
	$m = 3$	66.45 ± 0.20
	$m = 4$	$\mathbf{67.02 \pm 0.20}$
Transductive	$m = 15$	65.88 ± 0.22
	$m = 30$	67.98 ± 0.22
	$m = 45$	69.62 ± 0.22
	$m = 60$	$\mathbf{71.79 \pm 0.22}$
	$m = 75$	71.11 ± 0.22

Table 6. Evaluation of m on the mini-ImageNet dataset.

We evaluated the number m of the collected samples in the HAN on the mini-ImageNet dataset, as shown in Table 6. In the inductive setting, $|\boldsymbol{Z}_j| = 4$, and we measured m from 1 to 4. With the increase of m, we achieve better performance, and the best performance 67.02% is achieved when $m = 4$. The reason is since only support samples are available in $\boldsymbol{Z}_j$, a large m provides more information to generate prototypes. In the transductive setting, $|\boldsymbol{Z}_j| = 79$, and we measured m in $[15, 30, 45, 60, 75]$. The performance first increases, and then the performance tends to stabilize around 71%. In beginning, a large m provides more information, while in the stable stage, the information is sufficient to obtain good prototypes.

5.4. Effectiveness of Generated Curvatures

In this section, we evaluated the manner of generated curvatures. We replaced second-order statistics of features in CCG with concatenation and mean of features, and we evaluated generating curvatures using a hand-designed curvature estimation method [16]. After the CCG generates c for each class, we made some disturbance on the generated c and evaluated its performance. If the performance de-

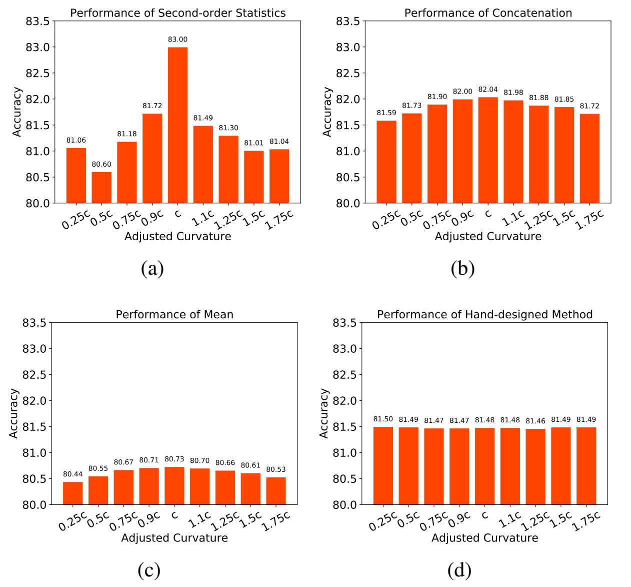

Figure 5. Evaluation of multiplying the generated c by 0.25, 0.5, 0.75, 0.9, 1.1, 1.25, 1.5, and 1.75. In Figure 5(a), 5(b), 5(c), the CCG generates c based on second-order statistics, concatenation, and mean of samples, respectively. In Figure 5(d), c is generated by a hand-designed curvature estimation method [16].

creases, it means that the original curvatures is appropriate. If the performance does not change much or even increases, it means that the original curvature is not good. Specifically, we disturbed the generated c via multiplying it by 0.25, 0.5, 0.75, 0.9, 1.1, 1.25, 1.5, and 1.75, and measured averaging accuracies over 10000 FSL tasks. We conducted 5-shot 5-way experiments on the mini-ImageNet dataset, and results are shown in Figure 5. We can find that, using second-order statistics can generate appropriate curvatures. When using the concatenation or mean of features, although the original c achieves the best performance, the performance difference between the disturbed c and the original c is small, showing the original c is not good enough. Finally, disturbed c of using the curvature estimation method achieves almost the same performance with its original c. The curvature estimation method usually requires a lot samples to avoid deviation in estimation [16], while an FSL task only provides very few samples. Thus, it cannot generate appropriate c. In contrast, CCG exploits meta-learning to learn knowledge from seen classes, which helps to generate appropriate curvatures for unseen classes given few samples.

5.5. Class Prototypes

We visualized the generated class prototypes on the 1-shot 5-way tasks of the mini-ImageNet dataset, and we used the MDS method [17] to reduce embeddings to 2-D vectors, as shown in Figure 6. We observe that sometimes the support sample may be an outlier (*e.g.*, the support sample of the miniature class in the left figure, and the bar class in the right figure). In this case, directly using the support samples as prototypes will result in bad performance. In contrast, our method can generate discriminative class prototypes based on the intra-class and inter-class context information of the adaptive curvatures, pushing the outlier support sample into its cluster. Thus, our method can improve the performance by a large margin. Note that, the tasks depicted in Figure 6 are hand-picked to show that our method can generate discriminative prototypes in extremely challenging scenarios, where support samples include outliers. Thus, the bad performance of directly using support samples in the figure is not common across all tasks.

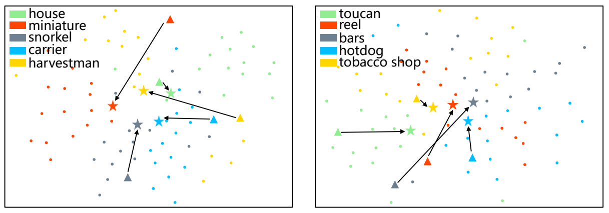

Figure 6. Visualization of data distributions and class prototypes on four FSL tasks of the min-ImangeNet dataset. Triangles represent support samples, dots denote query samples, and stars represent generated class prototypes. Sometimes the support sample may be an outlier (*e.g.*, the support sample of the miniature class in the left figure, and the bar class in the right figure). Directly using the triangles as prototypes will result in a bad performance. In contrast, our method can generate more discriminative prototypes.

6. Conclusion

In this paper, we have found that assuming data of all FSL tasks has the same geometrical structure may distort structures in some tasks and thus result in a poor generalization capability. To solve this issue, we have proposed a curvature generation embedding method that adapts a task-aware curved embedding space to data structures by generating curvatures for each class. Although few samples are given in FSL, using second-order statistics can capture expressive representations of them, which helps to generate suitable curvatures. Compared with an embedding space using a single curvature everywhere, curvatures for different classes can better match complex data. By considering the intra-class and inter-class context information of the adapted geometry, our method can generate discriminative class prototypes in the embedding space for few-shot classification. Extensive experimental results confirm that our embedding space can adapt well to new FSL tasks and achieves state-of-the-art performance.

Acknowledgements. This work was supported in part by the Natural Science Foundation of China (NSFC) under Grants No. 62072041 and No. 61773062.

References

[1] Arman Afrasiyabi, Jean-Franccois Lalonde, and Christian Gagné. Associative alignment for few-shot image classification. In *European Conference on Computer Vision (ECCV)*, pages 18–35, 2020.

[2] Sungyong Baik, Seokil Hong, and Kyoung Mu Lee. Learning to forget for meta-learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2379–2387, 2020.

[3] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14493–14502, 2020.

[4] Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations (ICLR)*, 2019.

[5] Qi Cai, Yingwei Pan, Ting Yao, Chenggang Yan, and Tao Mei. Memory matching networks for one-shot image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4080–4088, 2018.

[6] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Neural Information Processing Systems (NeurIPS)*, pages 4868–4879, 2019.

[7] Jindou Dai, Yuwei Wu, Zhi Gao, and Yunde Jia. A hyperbolic-to-hyperbolic graph convolutional network. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 154–163, June 2021.

[8] Bhuwan Dhingra, Christopher Shallue, Mohammad Norouzi, Andrew Dai, and George Dahl. Embedding text in hyperbolic spaces. In *Workshop on Graph-Based Methods for Natural Language Processing*, pages 59–69, 2018.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135, 2017.

[10] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Neural Information Processing Systems (NeurIPS)*, pages 5345–5355, 2018.

[11] Zhi Gao, Yuwei Wu, Xiaoxun Zhang, Jindou Dai, Yunde Jia, and Mehrtash Harandi. Revisiting bilinear pooling: A coding perspective. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 3954–3961, 2020.

[12] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4367–4375, 2018.

[13] Jiechao Guan, Zhiwu Lu, Tao Xiang, and Ji-Rong Wen. Few-shot learning as domain adaptation: Algorithm and analysis. *arXiv preprint:2002.02050*, 2020.

[14] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. Hyperbolic attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[15] W. He. Application of euclidean norm in multi-temporal remote sensing image change detection. In *International Congress on Image and Signal Processing*, pages 2111–2115, 2010.

[16] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6418–6428, 2020.

[17] J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, pages 1–27, 1964.

[18] M. Law, Renjie Liao, J. Snell, and R. Zemel. Lorentzian distance learning for hyperbolic representations. In *International Conference on Machine Learning (ICML)*, pages 3672–3681, 2019.

[19] John M. Lee. Riemannian manifolds: An introduction to curvature. 1997.

[20] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10657–10665, 2019.

[21] Aoxue Li, Weiran Huang, Xu Lan, Jiashi Feng, Zhenguo Li, and Liwei Wang. Boosting few-shot learning with adaptive margin loss. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12576–12584, 2020.

[22] Aoxue Li, Tiange Luo, Tao Xiang, Weiran Huang, and Liwei Wang. Few-shot learning with global class representations. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9715–9724, 2019.

[23] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–10, 2019.

[24] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. Factorized bilinear models for image recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2079–2087, 2017.

[25] Yann Lifchitz, Yannis Avrithis, Sylvaine Picard, and Andrei Bursuc. Dense classification and implanting for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9258–9267, 2019.

[26] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1449–1457, 2015.

[27] Bin Liu, Y. Cao, Yutong Lin, Q. Li, Zheng Zhang, Mingsheng Long, and H. Hu. Negative margin matters: Understanding margin in few-shot classification. In *European Conference on Computer Vision (ECCV)*, pages 438–455, 2020.

[28] Jinlu Liu, Liang Song, and Yongqiang Qin. Prototype rectification for few-shot learning. In *European Conference on Computer Vision (ECCV)*, pages 741–756, 2020.

[29] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. In *Neural Information Processing Systems (NeurIPS)*, pages 8230–8241, 2019.

[30] Shaoteng Liu, Jingjing Chen, Liangming Pan, Chong-Wah Ngo, Tat-Seng Chua, and Yu-Gang Jiang. Hyperbolic visual embedding learning for zero-shot recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9273–9281, 2020.

[31] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2019.

[32] Teng Long, Pascal Mettes, Heng Tao Shen, and Cees GM Snoek. Searching for actions on the hyperbole. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1141–1150, 2020.

[33] Brice Loustau. Hyperbolic geometry. *arXiv: Differential Geometry*, 2020.

[34] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Neural Information Processing Systems (NeurIPS)*, pages 6338–6347. 2017.

[35] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Neural Information Processing Systems (NeurIPS)*, pages 721–731, 2018.

[36] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. Hyperbolic deep neural networks: A survey. *ArXiv*, abs/2101.04562, 2021.

[37] Hang Qi, M. Brown, and D. Lowe. Low-shot learning with imprinted weights. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5822–5830, 2018.

[38] Limeng Qiao, Yemin Shi, Jia Li, Yaowei Wang, Tiejun Huang, and Yonghong Tian. Transductive episodic-wise adaptive metric for few-shot learning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3603–3612, 2019.

[39] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 331–339, 2019.

[40] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations (ICLR)*, 2018.

[41] Pau Rodríguez, Issam Laradji, Alexandre Drouin, and Alexandre Lacoste. Embedding propagation: Smoother manifold for few-shot classification. In *European Conference on Computer Vision (ECCV)*, pages 121–138, 2020.

[42] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. In *International Conference on Learning Representations (ICLR)*, 2021.

[43] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4136–4145, 2020.

[44] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Neural Information Processing Systems (NeurIPS)*, pages 4077–4087, 2017.

[45] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1199–1208, 2018.

[46] J. Tenenbaum and W. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12:1247–1283, 2000.

[47] Alexandru Tifrea, Gary Becigneul, and Octavian-Eugen Ganea. Poincare glove: Hyperbolic word embeddings. In *International Conference on Learning Representations (ICLR)*, 2018.

[48] A. Ungar. Analytic hyperbolic geometry: Mathematical foundations and applications. 2005.

[49] A. Ungar. A gyrovector space approach to hyperbolic geometry. 2009.

[50] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Neural Information Processing Systems (NeurIPS)*, pages 3630–3638, 2016.

[51] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[52] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019.

[53] Yu-Xiong Wang, Ross B. Girshick, M. Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7278–7286, 2018.

[54] Chen Xing, Negar Rostamzadeh, Boris Oreshkin, and Pedro OO Pinheiro. Adaptive cross-modal few-shot learning. In *Neural Information Processing Systems (NeurIPS)*, pages 4848–4858, 2019.

[55] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8808–8817, 2020.

[56] Sung Whan Yoon, Jun Seo, and J. Moon. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *International Conference on Machine Learning (ICML)*, pages 7115–7123, 2019.

[57] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12203–12213, 2020.

[58] Yiding Zhang, Xiao Wang, Xunqiang Jiang, Chuan Shi, and Yanfang Ye. Hyperbolic graph attention network. *arXiv preprint arXiv:1912.03046*, 2019.

[59] Jun-Yan Zhu, Philipp Krähenbühl, E. Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision (ECCV)*, pages 597–613, 2016.