Curvature Generation in Curved Spaces for Few-Shot Learning Supplementary Materials

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

1. Proof of Lemma 1

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*

$$w\otimes_c \exp_{\mathbf{0}}^c(\boldsymbol{x}) = \exp_{\mathbf{0}}^c(w\boldsymbol{x}), \tag{1}$$

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

Proof. We first denote

$$\boldsymbol{a} = \exp_{\mathbf{0}}^c(\boldsymbol{x}) = \tanh\left(\sqrt{c}\|\boldsymbol{x}\|\right)\frac{\boldsymbol{x}}{\sqrt{c}\|\boldsymbol{x}\|}, \tag{2}$$

and

$$\|\boldsymbol{a}\| = \frac{\tanh\left(\sqrt{c}\|\boldsymbol{x}\|\right)}{\sqrt{c}\|\boldsymbol{x}\|}\cdot\|\boldsymbol{x}\| = \frac{\tanh\left(\sqrt{c}\|\boldsymbol{x}\|\right)}{\sqrt{c}}. \tag{3}$$

Then, we can compute the left hand side of Eq. (1) as

$$\begin{aligned}
w\otimes_c\boldsymbol{a} =& \frac{1}{\sqrt{c}}\tanh\left(w\cdot\operatorname{artanh}(\sqrt{c}\|\boldsymbol{a}\|)\right)\frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}\\
=& \frac{1}{\sqrt{c}}\tanh\left(w\cdot\operatorname{artanh}(\sqrt{c}\frac{\tanh\left(\sqrt{c}\|\boldsymbol{x}\|\right)}{\sqrt{c}})\right)\\
&\cdot\tanh\left(\sqrt{c}\|\boldsymbol{x}\|\right)\frac{\boldsymbol{x}}{\sqrt{c}\|\boldsymbol{x}\|}\cdot\frac{\sqrt{c}}{\tanh\left(\sqrt{c}\|\boldsymbol{x}\|\right)}\\
=& \frac{1}{\sqrt{c}}\tanh\left(w\cdot\sqrt{c}\|\boldsymbol{x}\|\right)\cdot\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}\\
=& \tanh\left(\sqrt{c}\|w\boldsymbol{x}\|\right)\cdot\frac{w\boldsymbol{x}}{\sqrt{c}\|w\boldsymbol{x}\|}\\
=& \exp_{\mathbf{0}}^c(w\boldsymbol{x}).
\end{aligned} \tag{4}$$

Thus, we have

$$w\otimes_c\exp_{\mathbf{0}}^c(\boldsymbol{x}) = \exp_{\mathbf{0}}^c(w\boldsymbol{x}). \tag{5}$$

□

*Corresponding author

2. Poincaré ball

In this paper, we utilize the framework of Möbius gyrovector space to provide operations for the Poincaré ball. The Möbius gyrovector space provides the algebraic setting for hyperbolic geometry just as the vector space provides the algebraic setting for the Euclidean geometry. A rigorous theoretical and detailed mathematical background of the gyrovector space can be found in [13]. Here we briefly introduce several other commonly used operations in the Poincaré ball.

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 $\mathbb{D}_c^d$, given by

$$\exp_{\boldsymbol{x}}^c(\boldsymbol{v}) = \boldsymbol{x}\oplus_c\left(\tanh\left(\sqrt{c}\frac{\lambda_{\boldsymbol{x}}^c\|\boldsymbol{v}\|}{2}\right)\frac{\boldsymbol{v}}{\sqrt{c}\|\boldsymbol{v}\|}\right), \tag{6}$$

where $\lambda_{\boldsymbol{x}}^c = \frac{2}{1-c\|\boldsymbol{x}\|^2}$ is the conformal factor. The exponential map $\exp_{\mathbf{0}}^c$ in the tangent space $T_{\mathbf{0}}\mathbb{D}_c^d$ at the origin is defined as

$$\exp_{\mathbf{0}}^c(\boldsymbol{v}) = \tanh\left(\sqrt{c}\|\boldsymbol{v}\|\right)\frac{\boldsymbol{v}}{\sqrt{c}\|\boldsymbol{v}\|}. \tag{7}$$

Logarithmic map. The logarithmic map $\log_{\boldsymbol{x}}^c$ maps a vector $\boldsymbol{u}$ from the Poincaré ball $\mathbb{D}_c^d$ to the tangent space $T_{\boldsymbol{x}}\mathbb{D}_c^d$, given by

$$\log_{\boldsymbol{x}}^c(\boldsymbol{u}) = \frac{2}{\sqrt{c}\lambda_{\boldsymbol{x}}^c}\operatorname{artanh}(\sqrt{c}\|-\boldsymbol{x}\oplus_c\boldsymbol{u}\|)\frac{-\boldsymbol{x}\oplus_c\boldsymbol{u}}{\|-\boldsymbol{x}\oplus_c\boldsymbol{u}\|}. \tag{8}$$

The exponential map $\log_{\mathbf{0}}^c$ in the tangent space $T_{\mathbf{0}}\mathbb{D}_c^d$ at the origin is defined as

$$\log_{\mathbf{0}}^c(\boldsymbol{u}) = \operatorname{artanh}(\sqrt{c}\|\boldsymbol{u}\|)\frac{\boldsymbol{u}}{\sqrt{c}\|\boldsymbol{u}\|}. \tag{9}$$

Möbius matrix multiplication. The matrix multiplication of a vector $\boldsymbol{x} \in \mathbb{D}_c^d$ by a matrix $\boldsymbol{M} \in \mathbb{R}^{d'\times d}$ is defined by

the Möbius matrix-vector multiplication. If $\boldsymbol{M}\boldsymbol{x} \neq \mathbf{0}$, we have

$$\boldsymbol{M}\otimes_c\boldsymbol{x} = \frac{1}{\sqrt{c}}\tanh\Big(\frac{\|\boldsymbol{M}\boldsymbol{x}\|}{\boldsymbol{x}}\mathrm{artanh}(\sqrt{c}\|\boldsymbol{x}\|)\Big)\frac{\boldsymbol{M}\boldsymbol{x}}{\|\boldsymbol{M}\boldsymbol{x}\|}, \tag{10}$$

and $\boldsymbol{M}\otimes_c\boldsymbol{x} = \mathbf{0}$, if $\boldsymbol{M}\boldsymbol{x} = \mathbf{0}$.

Parallel transport. The parallel transport $P^c_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v})$ transforms a vector $\boldsymbol{v}$ from one tangent space $T_{\boldsymbol{x}}\mathbb{D}^d_c$ to another tangent space $T_{\boldsymbol{y}}\mathbb{D}^d_c$,

$$P^c_{\boldsymbol{x}\to\boldsymbol{y}}(\boldsymbol{v}) = \log^c_{\boldsymbol{y}}\big(\boldsymbol{y}\oplus_c\exp^c_{\boldsymbol{x}}(\boldsymbol{v})\big). \tag{11}$$

If the tangent space $T_{\boldsymbol{x}}\mathbb{D}^d_c$ is at the origin of the Poincaré ball, *i.e.*, $\boldsymbol{x} = \mathbf{0}$, the parallel transport $P^c_{\mathbf{0}\to\boldsymbol{y}}(\boldsymbol{v})$ is

$$P^c_{\mathbf{0}\to\boldsymbol{y}}(\boldsymbol{v}) = \log^c_{\boldsymbol{y}}(\boldsymbol{y}\oplus_c\exp^c_{\mathbf{0}}(\boldsymbol{v})) = \frac{\lambda^c_{\mathbf{0}}}{\lambda^c_{\boldsymbol{y}}}\boldsymbol{v}. \tag{12}$$

3. Implementation Details

Datasets. We conducted experiments on four popular datasets, namely Mini-ImageNet [14], tiered-ImageNet [10], CUB dataset [15], and CIFAR-FS dataset [1]. The Mini-ImageNet dataset contains 100 classes from the ImageNet dataset, and each class has 600 images. We splited the 100 classes into 64, 16, and 20 for training, validation, and testing, respectively. The tiered-ImageNet dataset has 779165 images from 608 classes totally, where 351, 97, and 160 classes were used for training, validation, and testing, respectively. Images of the Mini-ImageNet and tiered-ImageNet datasets were resized into 84×84 pixels. The CUB dataset is a fine-grained image dataset that contains 200 bird classes and 11788 images totally. Following the protocol of [8, 16], we utilized 100, 50, and 50 classes for training, validation, and testing, respectively. We cropped bird regions from images with given bounding boxes before training. CIFAR-FS is a FSL dataset derived from CIFAR-100 [3]. It contains 100 classes, where 64, 16, and 20 classes are used for training, validation, and testing, respectively. Each class has 600 images with size of 32×32.

Architecture. In this paper, we used the a 4-layer convolutional network or a 12-layer residual network as the backbone, *i.e.*, the feature extractor $f_\theta(\cdot)$ in our method. The 4-layer convolutional network (ConvNet) [6, 12, 14] and the 12-layer residual network (ResNet12) are commonly used backbones for FSL. For ConvNet, its channels in the 4 convolutional layers are $(64, 64, 64, 64)$. For ResNet12, note that, in previous works, there exists two ResNet12 backbones. The two ResNet12 backbones both have four residual blocks, while the numbers of convolutional channels in the four blocks are different. In some works, such as [7, 8, 9], they used a smaller ResNet12 backbone (ResNet12), and their numbers of convolutional channels in the four blocks are $(64, 128, 256, 512)$. In some works, such as [5, 11, 16], they used a bigger ResNet12 backbone (BigResNet12), and their numbers of convolutional channels in the four blocks are $(64, 160, 320, 640)$. For fair and comprehensive comparisons with existing methods, we evaluated our method in both the two ResNet12 backbones. In our paper, we denote the smaller ResNet12 backbone as ResNet12, and denote the bigger ResNet12 backbone as BigResNet12.

Training. Trianing and evaluating our model requires one GeForce GTX 2080Ti GPU with 11GB. We first pre-trained the ResNet12 and BigResNet12 backbones using the cross-entropy loss on the training set, while we did not pre-train the ConvNet backbone. For the Mini-ImageNet and tiered-ImageNet datasets, the pre-trained ResNet12[1] and BigResNet12[2] backbones were downloaded from the code webpages of works [2, 16]. For the CIFAR-FS dataset, we used the Adam optimizer over 100 epoches to pre-train the BigResNet12 backbone. We set the learning rate was as 0.001 in the pre-training stage, decayed the learning rate per 50 epoches, and the dacay rate was set as 0.1. Then, we removed the last FC layer and the softmax layer of the pre-trained model, and the rest layers were used as the feature extractor $f_\theta(\cdot)$ in our method. Finally, we carried out meta-training to train our feature extractor, class-curvature generator (CCG), and hyperbolic aggregation network (HAN) together over 20000 episodes. We used the Adam optimizer in the meta-training stage, where the learning rate for the CCG and HAN was 0.001. We decayed the learning rate per 8000 episodes in the meta-training stage, and the dacay rate was 0.1. We set the value of the weight decay as 0.0005 for our method, except for the ConvNet backbone. The value of the weight decay was 0 for the ConvNet backbone.

In our method, we have three hyperparameters: ϱ in CCG, r in CCG, and m in HAN. We set $\varrho = 128$ and $r = 5$. We set $m = 4$ and $m = 20$ for 1-shot and 5-shot inductive tasks, respectively. In transductive setting, we set $m = 60$.

4. Additional Experimental Results

4.1. Intra-class and inter-class context information

Here we evaluated the effectiveness of the used intra-class and inter-class context information in our hyperbolic aggregation network (HAN). Experiments were conducted on the Mini-ImageNet dataset using the BigResNet12 backbone. We show the performance of removing the intra-class context information, removing the inter-class context information, and removing both of them. Concretely, removing the intra-class context information means not computing weights for samples in the in-class set $\boldsymbol{X}_j$ and use hyperbolic averaging of features as the preliminary prototypes,

[1] https://github.com/Sha-Lab/FEAT

[2] https://github.com/cyvius96/few-shot-meta-baseline

Table 1. Evaluation of the intra-class and inter-class context information.

Setting	Method	1-shot	5-shot
Inductive	w/o intra inter	59.47 ± 0.20	80.41 ± 0.14
	w/o intra	65.81 ± 0.20	81.01 ± 0.14
	w/o inter	59.47 ± 0.20	80.87 ± 0.14
	GAE (Ours)	$\mathbf{67.02 \pm 0.20}$	$\mathbf{82.32 \pm 0.14}$
Transductive	w/o intra inter	59.94 ± 0.21	80.67 ± 0.14
	w/o intra	63.69 ± 0.20	81.04 ± 0.14
	w/o inter	59.94 ± 0.21	80.96 ± 0.15
	GAE (Ours)	$\mathbf{71.79 \pm 0.23}$	$\mathbf{83.00 \pm 0.17}$

denoted as 'w/o intra'. Removing the inter-class context information means not computing weights for or aggregating samples in the out-of-class set $\boldsymbol{Z}_j$ and using the preliminary prototypes as the final prototypes, denoted as 'w/o inter'. For 1-shot tasks, there is only 1 sample in $\boldsymbol{X}_j$, and the weight assigned for this sample is always equal to one. Thus, 'w/o inter' of the 1-shot tasks has the same performance with 'w/o intra inter'. Results are shown in Table 1.

We can find that, both the intra-class and inter-class context information have influence in the hyperbolic aggregation network. For example, without the intra-class context information, 'w/o intra' achieves 81.01% and 81.04% in the 5-shot inductive and transductive settings, respectively, 1.31% and 1.96% lower than GAE. Without the inter-class context information, 'w/o intra' achieves 80.87% and 80.96% in the 5-shot inductive and transductive settings, respectively, 1.45% and 2.04% lower than GAE. When neither the intra-class context information nor the inter-class context information is considered, 'w/o intra inter' achieves 80.41 and 80.67 in the 5-shot inductive and transductive settings, 1.91% and 2.33% lower than GAE. This experimental results show that both the intra-class and inter-class context information play important roles in the aggregation samples into discriminative prototypes.

4.2. More ablations

In this section, we added more experiments to show the performance of manully setting c. Experiments are conducted on the Mini-ImageNet dataset using the BigResNet12 backbone. Compared with ablation study in the manuscript that manully set c as 1, 0.1, 0.01, and 0.001, we further set the c as 10, 5, 2, 0.0001, 0.00001, and 0.000001, denoted by 'w/o CCG HAN, $c = 10/5/2/0.0001/0.00001/0.000001$' and 'w/o CCG, $c = 10/5/2/0.0001/0.00001/0.000001$'. We conducted an ablation study that learns two fully-connected layers to generate task-specific embeddings and prototypes to further show the effectiveness of our CCG and HAN, denoted by 'completely learned'. We conducted an ablation that replaces hyperbolic geometry with spherical geometry and generates positive curvatures to show the effectiveness of hyperbolic geometry with negative curvatures, denoted by 'spheircal'.

Table 2. Ablation experiments on the Mini-ImageNet dataset.

Setting	Method	1-shot	5-shot
Inductive	ProtoNet	58.34 ± 0.20	78.49 ± 0.14
	w/o CCG HAN, $c = 10$	56.72 ± 0.21	79.21 ± 0.14
	w/o CCG HAN, $c = 5$	56.49 ± 0.21	79.25 ± 0.14
	w/o CCG HAN, $c = 2$	55.81 ± 0.21	79.17 ± 0.14
	w/o CCG HAN, $c = 1$	59.05 ± 0.21	78.34 ± 0.22
	w/o CCG HAN, $c = 0.1$	51.38 ± 0.29	56.20 ± 0.34
	w/o CCG HAN, $c = 0.01$	50.92 ± 0.22	75.66 ± 0.16
	w/o CCG HAN, $c = 0.001$	58.68 ± 0.21	76.94 ± 0.14
	w/o CCG HAN, $c = 0.0001$	58.70 ± 0.20	77.30 ± 0.14
	w/o CCG HAN, $c = 0.00001$	58.79 ± 0.20	79.37 ± 0.14
	w/o CCG HAN, $c = 0.000001$	58.58 ± 0.21	79.29 ± 0.14
	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, $c = 10$	61.31 ± 0.20	75.13 ± 0.18
	w/o CCG, $c = 5$	61.20 ± 0.20	74.47 ± 0.18
	w/o CCG, $c = 2$	60.51 ± 0.20	73.80 ± 0.19
	w/o CCG, $c = 1$	62.60 ± 0.20	79.25 ± 0.14
	w/o CCG, $c = 0.1$	57.65 ± 0.21	59.52 ± 0.30
	w/o CCG, $c = 0.01$	64.17 ± 0.21	76.49 ± 0.16
	w/o CCG, $c = 0.001$	59.12 ± 0.21	77.87 ± 0.16
	w/o CCG, $c = 0.0001$	63.12 ± 0.21	80.33 ± 0.14
	w/o CCG, $c = 0.00001$	62.78 ± 0.20	79.65 ± 0.15
	w/o CCG, $c = 0.000001$	62.40 ± 0.20	80.57 ± 0.14
	completely learned	59.97 ± 0.25	77.77 ± 0.20
	spheircal	63.56 ± 0.21	81.00 ± 0.15
	GAE (Ours)	$\mathbf{67.02 \pm 0.20}$	$\mathbf{82.32 \pm 0.14}$
Transductive	w/o CCG HAN, $c = 10$	56.49 ± 0.21	79.18 ± 0.14
	w/o CCG HAN, $c = 5$	57.09 ± 0.21	79.19 ± 0.14
	w/o CCG HAN, $c = 2$	57.14 ± 0.21	79.21 ± 0.14
	w/o CCG HAN, $c = 1$	59.14 ± 0.22	79.82 ± 0.21
	w/o CCG HAN, $c = 0.1$	51.04 ± 0.29	56.82 ± 0.33
	w/o CCG HAN, $c = 0.01$	51.06 ± 0.22	75.73 ± 0.16
	w/o CCG HAN, $c = 0.001$	58.74 ± 0.21	78.18 ± 0.14
	w/o CCG HAN, $c = 0.0001$	58.92 ± 0.20	78.15 ± 0.14
	w/o CCG HAN, $c = 0.00001$	58.53 ± 0.21	80.49 ± 0.14
	w/o CCG HAN, $c = 0.000001$	57.88 ± 0.21	80.57 ± 0.14
	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, $c = 10$	62.87 ± 0.20	79.65 ± 0.18
	w/o CCG, $c = 5$	62.92 ± 0.20	79.73 ± 0.18
	w/o CCG, $c = 2$	62.86 ± 0.20	79.49 ± 0.19
	w/o CCG, $c = 1$	62.80 ± 0.20	81.02 ± 0.14
	w/o CCG, $c = 0.1$	51.69 ± 0.24	65.61 ± 0.25
	w/o CCG, $c = 0.01$	62.50 ± 0.21	77.59 ± 0.16
	w/o CCG, $c = 0.001$	69.66 ± 0.22	80.91 ± 0.14
	w/o CCG, $c = 0.0001$	69.50 ± 0.22	80.72 ± 0.14
	w/o CCG, $c = 0.00001$	62.71 ± 0.20	79.61 ± 0.15
	w/o CCG, $c = 0.000001$	62.81 ± 0.20	79.86 ± 0.15
	model capacity	68.25 ± 0.21	81.08 ± 0.16
	GAE (Ours)	$\mathbf{71.79 \pm 0.23}$	$\mathbf{83.00 \pm 0.17}$

In the 'model capacity' experiment, we set the backbone as $(64, 150, 320, 610)$ to keep the number of our whole parameters consistent with the backbone. We used the 'model capacity' experiment to show whether our improvement is from the model capacity. Results are shown in Table 2. More experiments further demonstrate that producing appropriate curvatures leads to better performance than manually setting a fixed and unitary curvature for various FSL tasks.

4.3. Efficiency

We provided the time and memory cost of our method on both the 1-shot 5-way and 5-shot 5-way tasks. Experiments were conducted on the mini-ImageNet dataset, where the BigResNet12 backbone was used. We used official codes of compared methods. The performance was measured using an Inter(R) Core(TM) i7-7820X 3.6GHz CPU, GeForce

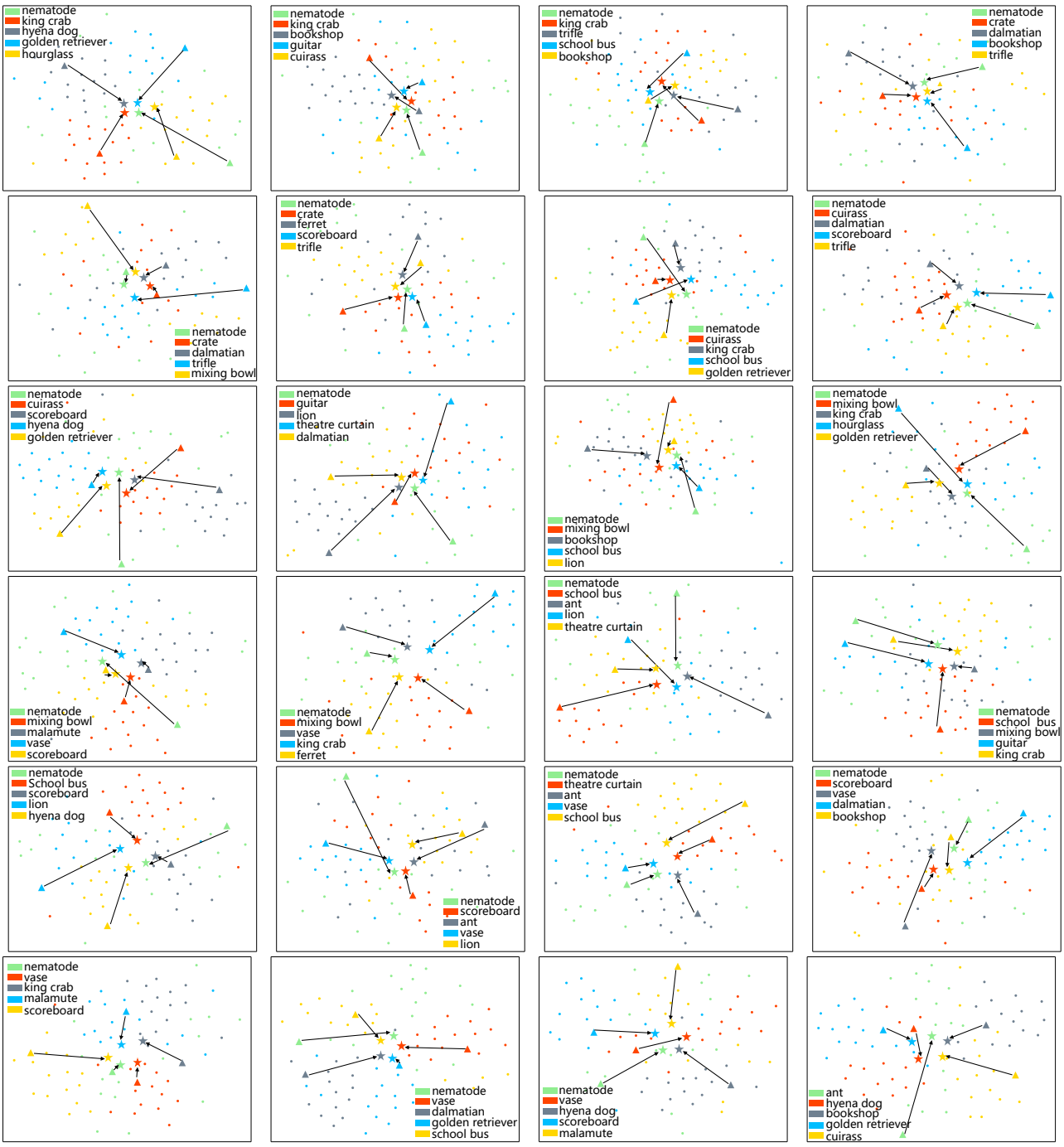

Figure 1. Visualization of data distributions and class prototypes on 24 FSL tasks of the min-ImangeNet dataset. Triangles represent support samples, dots denote query samples, and stars represent generated class prototypes.

Table 3. Training and test time (seconds) on the mini-ImageNet dataset.

Method	**1-shot training time**	**1-shot test time**	**5-shot training time**	**5-shot test time**
FEAT [16]	3240	787	4166	955
DSN [11]	5769	943	7561	1247
Ours	4445	610	5507	775

Table 4. Memory cost (MB) on the mini-ImageNet dataset.

Method	1-shot Memory cost	5-shot Memory cost
FEAT [16]	4749	5527
DSN [11]	8267	9681
Ours	4785	5633

GTX 2080Ti GPU and 32GB RAM. The training and test time of all methods was measured over the same number of episodes. Results are shown in Table 3. Results show that, our method requires the least test time. In other words, our method takes the least time to adapt the embedding space to a new task. We also compare the memory consumption with state-of-the-art methods, as shown in Table 4. We can find that, our method requires similar memory consumption with FEAT and less memory consumption than DSN. In fact, our CCG and HAN are not large. Here we provide our largest parameter number (using BigResNet12). CCG, HAN, and the backbone have 8.20×10^5, 7567, and 1.24×10^7 parameters, respectively. Compared with backbone, our CCG and HAN are indeed small. Considering the accuracy improvement, we think the memory and computation costs are acceptable.

4.4. Prototypes

In this section, we visualized more generated class prototypes, as shown in Figure 1. Experiments were conducted on the 1-shot 5-way tasks of the Mini-ImageNet dataset, where the ConvNet backbone was used. Concretely, we computed all pairs of distances between embeddings, and sent the distances to the MDS dimensionality reduction method [4] to reduce embeddings to 2-D vectors. Experimental results show that our method can generate discriminative prototypes even outlier support instances are given.

References

[1] 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.

[2] Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *ArXiv*, abs/2003.04390, 2020.

[3] Established In. *Canadian Institute for Advanced Research*. Alphascript Publishing, 2010.

[4] J. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29:1–27, 1964.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] A. Ungar. A gyrovector space approach to hyperbolic geometry. In *A Gyrovector Space Approach to Hyperbolic Geometry*, 2009.

[14] 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.

[15] 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.

[16] 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.