Supplementary Material: Kernel Methods in Hyperbolic Spaces

Pengfei Fang[1,3], Mehrtash Harandi[2,3], Lars Petersson[3]
[1]The Australian National University, [2]Monash University, [3]DATA61-CSIRO
Pengfei.Fang@anu.edu.au, mehrtash.harandi@monash.edu, Lars.Petersson@data61.csiro.au

1. Proof of Length Equivalence

Here, we prove the Theorem 1 from § 4 of the main paper. The proof of this theorem follows several steps. We first formally define the curve length and intrinsic metric.

Definition 1. *(Curve Length) The length of a curve γ is the supremum of $L(\gamma; \{t_i\}_{i=0}^n)$ over all possible partitions $\{t_i\}_{i=0}^n$, where $0 = t_0 < t_1 < \ldots < t_{n-1} < t_n = 1$ and $L(\gamma; \{t_i\}_{i=0}^n) = \sum_{i=1}^n d(\gamma(t_{i-1}), \gamma(t_i))$.*

Definition 2. *(Intrinsic Metric) The intrinsic metric $\hat{\delta}(x, y)$ on $\mathcal{M}$ is defined as the infimum of the lengths of all paths from x to y.*

Theorem 1. *([4]) If the intrinsic metrics induced by two metrics d_1 and d_2 are identical to a scale ζ, then the length of any given curve is the same under both metrics up to ζ.*

Theorem 2. *([4]) if $d_1(x, y)$ and $d_2(x, y)$ are two metrics defined on a space $\mathcal{M}$ such that*

$$\lim_{d_1(x,y)\to 0} \frac{d_2(x, y)}{d_1(x, y)} = 1, \tag{1}$$

uniformly (with respect to x and y), then the length of any given curve is the same under both metrics. Consequently, the intrinsic metrics induced by d_1 and d_2 are identical.

Therefore, we need to study the the behavior of

$$\lim_{d_c(\boldsymbol{z}_i,\boldsymbol{z}_j)\to \mathbf{0}} \frac{\tilde{\lambda}_c \, d_e(\boldsymbol{z}_i, \boldsymbol{z}_j)}{d_c(\boldsymbol{z}_i, \boldsymbol{z}_j)}, \tag{2}$$

to prove our theorem. This is also equivalent to study

$$\lim_{\gamma\to \mathbf{0}} \frac{\tilde{\lambda}_c \, d_e(\boldsymbol{z}_i, \boldsymbol{z}_j)}{d_c(\boldsymbol{z}_i, \boldsymbol{z}_j)}, \tag{3}$$

where $\boldsymbol{z}_j = \boldsymbol{z}_i \oplus_c \gamma$.

Proof. We first prove $f_{\mathbb{D}}(\boldsymbol{z}_i \oplus_c \gamma) = f_{\mathbb{D}}(\boldsymbol{z}_i) + f_{\mathbb{D}}(\gamma)$ for $\gamma \to \mathbf{0}$.

$$\begin{aligned}
\boldsymbol{z}_i \oplus_c \gamma &= \frac{(1 + 2c\langle \boldsymbol{z}_i, \gamma\rangle + c\|\gamma\|^2)\boldsymbol{z}_i + (1 - c\|\boldsymbol{z}_i\|^2)\gamma}{1 + 2c\langle \boldsymbol{z}_i, \gamma\rangle + c^2\|\boldsymbol{z}_i\|^2\|\gamma\|^2} \\
&\approx \frac{(1 + 2c\langle \boldsymbol{z}_i, \gamma\rangle)\boldsymbol{z}_i + (1 - c\|\boldsymbol{z}_i\|^2)\gamma}{1 + 2c\langle \boldsymbol{z}_i, \gamma\rangle} \\
&\approx \boldsymbol{z}_i + \frac{1 - c\|\boldsymbol{z}_i\|^2}{1 + 2c\langle \boldsymbol{z}_i, \gamma\rangle}\gamma \\
&= \boldsymbol{z}_i + \kappa\gamma.
\end{aligned} \tag{4}$$

Then the first order approximation of $f_{\mathbb{D}}(\boldsymbol{z}_i + \kappa\gamma)$ can be obtained:

$$\begin{aligned}
f_{\mathbb{D}}(\boldsymbol{z}_i + \kappa\gamma) &= \tanh^{-1}(\sqrt{c}\|\boldsymbol{z}_i + \kappa\gamma\|)\frac{\boldsymbol{z}_i + \kappa\gamma}{\sqrt{c}\|\boldsymbol{z}_i + \kappa\gamma\|} \\
&\approx \boldsymbol{z}_i + \kappa\gamma + \frac{c\|\boldsymbol{z}_i + \kappa\gamma\|^2}{3}(\boldsymbol{z}_i + \kappa\gamma) \\
&\approx \boldsymbol{z}_i + \kappa\gamma + \frac{c\|\boldsymbol{z}_i + \kappa\gamma\|^2}{3}\boldsymbol{z}_i \\
&+ \frac{c\|\boldsymbol{z}_i + \kappa\gamma\|^2}{3}\kappa\gamma \\
&\approx \boldsymbol{z}_i + \kappa\gamma + \frac{c\|\boldsymbol{z}_i\|^2}{3}\boldsymbol{z}_i \\
&+ \frac{2c\langle \boldsymbol{z}_i, \kappa\gamma\rangle}{3}\boldsymbol{z}_i + \frac{c\|\boldsymbol{z}_i + \kappa\gamma\|^2}{3}\kappa\gamma
\end{aligned} \tag{5}$$

The first order approximation of $f_{\mathbb{D}}(\boldsymbol{z}_i)$ and $f_{\mathbb{D}}(\gamma)$ can also been obtained:

$$f_{\mathbb{D}}(\boldsymbol{z}_i) \approx \boldsymbol{z}_i + \frac{c\|\boldsymbol{z}_i\|^2}{3}\boldsymbol{z}_i \tag{6}$$

and

$$f_{\mathbb{D}}(\gamma) \approx \gamma + \frac{c\|\gamma\|^2}{3}\gamma. \tag{7}$$

Then we can see:

$$\lim_{\gamma\to \mathbf{0}} \big(f_{\mathbb{D}}(\boldsymbol{z}_i + \kappa\gamma) - f_{\mathbb{D}}(\boldsymbol{z}_i) - f_{\mathbb{D}}(\gamma)\big) = 0. \tag{8}$$

Since it holds that $f_{\mathbb{D}}(\boldsymbol{z}_i \oplus_c \gamma) = f_{\mathbb{D}}(\boldsymbol{z}_i) + f_{\mathbb{D}}(\gamma)$ for $\gamma \to \mathbf{0}$, then we have

$$\begin{aligned}
\lim_{\gamma\to\mathbf{0}} \frac{\tilde{\lambda}_c d_e(\boldsymbol{z}_i, \boldsymbol{z}_j)}{d_c(\boldsymbol{z}_i, \boldsymbol{z}_j)} &= \lim_{\gamma\to\mathbf{0}} \frac{\tilde{\lambda}_c \|f_{\mathbb{D}}(\boldsymbol{z}_i) - f_{\mathbb{D}}(\boldsymbol{z}_j)\|}{\frac{2}{\sqrt{c}}\tanh^{-1}(\sqrt{c}\|-\boldsymbol{z}_i \oplus_c \boldsymbol{z}_j\|)} \\
&= \lim_{\gamma\to\mathbf{0}} \frac{\tilde{\lambda}_c \|f_{\mathbb{D}}(\boldsymbol{z}_i) - f_{\mathbb{D}}(\boldsymbol{z}_i \oplus_c \gamma)\|}{\frac{2}{\sqrt{c}}\tanh^{-1}\big(\sqrt{c}\|(-\boldsymbol{z}_i) \oplus_c (\boldsymbol{z}_i \oplus_c \gamma)\|\big)} \\
&= \lim_{\gamma\to\mathbf{0}} \frac{\tilde{\lambda}_c \|f_{\mathbb{D}}(\boldsymbol{z}_i) - (f_{\mathbb{D}}(\boldsymbol{z}_i) + f_{\mathbb{D}}(\gamma))\|}{\frac{2}{\sqrt{c}}\tanh^{-1}(\sqrt{c}\|\gamma\|)} \\
&= \lim_{\gamma\to\mathbf{0}} \frac{\tilde{\lambda}_c \|f_{\mathbb{D}}(\gamma)\|}{\frac{2}{\sqrt{c}}\tanh^{-1}(\sqrt{c}\|\gamma\|)} \\
&= \lim_{\gamma\to\mathbf{0}} \frac{\tilde{\lambda}_c \|\tanh^{-1}(\sqrt{c}\|\gamma\|)\frac{\gamma}{\sqrt{c}\|\gamma\|}\|}{\frac{2}{\sqrt{c}}\tanh^{-1}(\sqrt{c}\|\gamma\|)} \\
&= \lim_{\gamma\to\mathbf{0}} \frac{\frac{\tilde{\lambda}_c}{\sqrt{c}}\tanh^{-1}(\sqrt{c}\|\gamma\|)\|\frac{\gamma}{\|\gamma\|}\|}{\frac{2}{\sqrt{c}}\tanh^{-1}(\sqrt{c}\|\gamma\|)} \\
&= 1.
\end{aligned} \tag{9}$$

This ends the proof. □

2. Understanding of Length Equivalence Theorem

To better understand the theorem, consider a curve $\gamma(\cdot) : [0, 1] \to \mathbb{D}_c^n$ in the Poincaré ball. To measure the length of this curve, one needs to break it down into infinitesimal segments, use the geodesic distance between the end points of each segment and sum up the results. A partition of the interval $[0, 1]$ is a sequence of points $\mathscr{T} = \{t_i\}_{i=0}^n$, $t_0 = 0, t_n = 1$, $t_i < t_{i+1}$. Then the length of the "inscribed polygon" can be defined as $L(\gamma; \mathscr{T}) = \sum_{i=1}^n d_c(\gamma(t_{i-1}), \gamma(t_i))$ (recall that d_c denotes the geodesic distance). The length of the curve γ is now simply the supremum of all possible partitions, meaning $L(\gamma) = \sup_{\mathscr{T}} L(\gamma; \mathscr{T})$ which realizes the above statement w.r.t. to the length of a curve. We recall that an RBF function with geodesic distance does not realize a pd kernel (see Theorem 2 in the main paper). As such, to devise pd kernels, we need to embed the Poincaré ball gracefully into a space where pd kernels can be realized, meaning that the embedding function should capture properties of the Poincaré ball one way or another. In our work, we choose the embedding to be Eq. (5). Now the theorem on the length of the curves establishes a very strong connection between the proposed embedding and the geometry of the Poincaré ball, as the length of the curves is the equivalent under the geodesic distance d_c and d_e up to a scale of $\tilde{\lambda}_c$. Note that this theorem does not imply that geodesics are equivalent, only the curves have the equivalent length.

3. Proof of Negative Definite Kernels

In § 4 of the main paper, Lemma 2 states the squared norm of the proposed distance (*i.e.*, d_e^2) is a negative definite kernel. Here, we give the proof of Lemma 2.

Proof. Suppose $\sum_{i=1}^m c_i = 0$, then we have:

$$\begin{aligned}
&\sum_{i,j=1}^m c_i c_j \|f(\boldsymbol{z}_i) - f(\boldsymbol{z}_j)\|^2 \\
&= \sum_{i,j=1}^m c_i c_j \big\{\|f(\boldsymbol{z}_i)\|^2 + \|f(\boldsymbol{z}_j)\|^2 \\
&- \langle f(\boldsymbol{z}_i), f(\boldsymbol{z}_j)\rangle - \langle f(\boldsymbol{z}_j), f(\boldsymbol{z}_i)\rangle\big\} \\
&= \sum_{i=1}^m c_i \|f(\boldsymbol{z}_i)\|^2 \sum_{j=1}^m c_j + \sum_{j=1}^m c_j \|f(\boldsymbol{z}_j)\|^2 \sum_{i=1}^m c_i \\
&- \langle \sum_{i=1}^n c_i f(\boldsymbol{z}_i), \sum_{j=1}^n c_j f(\boldsymbol{z}_j)\rangle - \langle \sum_{j=1}^n c_j f(\boldsymbol{z}_j), \sum_{i=1}^n c_i f(\boldsymbol{z}_i)\rangle \\
&= -2\|\sum_{i=1}^n c_i f(\boldsymbol{z}_i)\|^2 \leq 0.
\end{aligned} \tag{10}$$

Thus $k(\boldsymbol{z}_i, \boldsymbol{z}_j) = \|f(\boldsymbol{z}_i) - f(\boldsymbol{z}_j)\|^2$ is negative definite. This ends the proof. □

4. Dataset Details

4.1. Few-shot Learning

The ***mini*ImageNet** is a subset of the ImageNet dataset, and contains 60,000 images in total. It has 100 classes and each class has 600 images. We also follow the standard evaluation protocol, which splits the 100 classes into 64 for training, 16 for validation and 20 for testing. The **CUB** dataset is a fine-grained image recognition dataset and we also use it to evaluate our few-shot learning algorithms. The CUB dataset consists of 200 different species of birds and 11,788 images in total. We also follow the standard setting to split the dataset into 100 base classes, 50 validation classes and 50 test classes. Similar to *mini*ImageNet, ***tiered*-ImageNet** is also a subset of ImageNet with broader classes (*i.e.*, 608 classes in total). The *tiered*-ImageNet contains 351 classes for training, 97 classes for validation and 160 classes for testing. **FC100**, which is based on the CIFAR-100, is proposed for the FSL task. It also contains three data splits, *i.e.* training split, validation split and test split, with each having 60, 20, 20 classes. The statistics of the dataset are summarized in Table 1.

4.2. Zero-shot Learning

We evaluate the zero-shot learning on **SUN**, **CUB**, **AWA1** and **AWA2** datasets. The visual features of

Table 1. Statistics of the datasets for few-shot learning.

Dataset	#classes / image train	#classes / image validation	#classes / image test	#classes / image total
*mini*ImageNet	64 / 38,400	16 / 9,600	20 / 12,000	100 / 60,000
CUB	100 / 5,891	50 / 2,932	50 / 2,965	200 / 11,788
tiered-ImageNet	351 / 448,695	97 / 124,261	160 / 206,209	608 / 779,165
FC100	60 / 36,000	20 / 12,000	20 / 12,000	100 / 60,000

Table 2. Statistics of the datasets for zero-shot learning.

Dataset	#images (train + val)	#image (test seen / unseen)	#seen classes	#unseen classes	#total classes
SUN	10,320	2,580 / 1,440	645	72	717
CUB	7,075	1,764 / 2,679	150	50	200
AWA1	19,832	4,958 / 5,685	40	10	50
AWA2	23,527	5,882 / 7,913	40	10	50

Table 3. Statistics of the datasets for person re-identification.

Dataset	#IDs / Image train	#IDs / Image test query	#IDs / Image test gallery	#IDs / Image total	Test setting
Market-1501	751 / 12,936	750 / 3,368	751 / 15,913	1,501 / 32,217	single query
DukeMTMC-reID	702 / 16,522	702 / 2,228	1,110 / 17,661	1,404 / 36,411	single query

Table 4. Statistics of the datasets for knowledge distillation.

Dataset	#classes / Image train	#classes / Image test	Test setting
CIFAR-10	10 / 50,000	10 / 10,000	classification accuracy
CIFAR-100	100 / 50,000	100 / 10,000	classification accuracy

all datasets are extracted from the ImageNet pre-trained ResNet-101 and the dimension are 2048. The dimensions of semantic features are 102, 312, 85, and 85 for SUN, CUB, AWA1 and AWA2, respectively. SUN is a fine-grained dataset and contains 717 classes with 14,340 images in total. Those 717 classes are annotated with 102 attributes. CUB, another fine-grained dataset, contains 11,788 images of 200 different species of birds, annotated with 312 attributes. The AWA1 is a coarse-grained dataset with animal images. It has 30,475 images with 50 classes, which are annotated by 85 attributes. Similar to AWA1, AWA2 consists of 37,322 images with the same animal classes and attributes as AWA1. The statistics of the dataset are summarized in Table 2.

4.3. Person Re-identification

The **Market-1501** dataset consists of 32,668 pedestrian images, captured by 6 disjoint cameras. The person bounding boxes are detected automatically by DPM [3]. This dataset is split into 12,936 images of 751 identities for training and 19,732 of 750 identities for testing. **DukeMTMC-reID** is collected by 8 non-overlapped cameras and the person bounding boxes are manually annotated. Following the standard training protocol, this dataset is divided into 16,522 and 19,889 images for training and testing, respectively. The statistics of the dataset are summarized in Table 3.

4.4. Knowledge Distillation

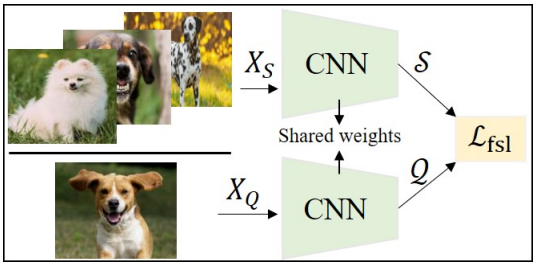
Both **CIFAR-10** and **CIFAR-100** have 50,000 images for training and 10,000 images for evaluation. CIFAR-10 contains 10 classes, with each containing 5,000 samples, while CIFAR-100 contains 100 classes, and each class has 500 samples. The input size of CIFAR-10 and CIFAR-100 are fixed to 32×32. The statistics of the dataset are summarized in Table 4.

5. Experimental Details

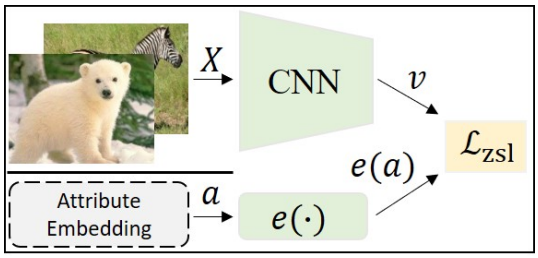
This section shows the implementation details and loss functions using our proposed kernels for the applications presented in the main paper.

5.1. Few-shot Learning

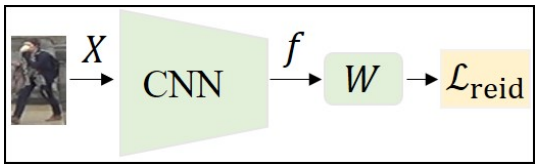
Implementation Details. The network is trained in a meta-learning manner (see Fig. 1(a)), which is also known as task-agnostic FSL. In each iteration, we sample an episode of data to train the network. Specifically, this protocol is well-known as N-way K-shot classification. We implement our methods for few-shot learning (FSL) in the PyTorch machine learning package [7]. The embedding dimensions



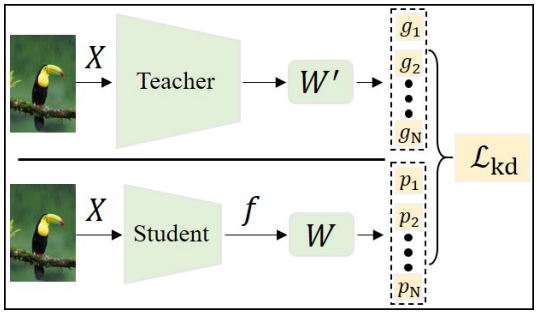
(a) The pipeline of the deep network for few-shot recognition. $\boldsymbol{X}_S$ and $\boldsymbol{X}_Q$ denote the input images in the support set and query set.



(b) The pipeline of the deep network for zero-shot learning. $\boldsymbol{X}$ and $\boldsymbol{a}$ denotes input images and attribute descriptors.



(c) The pipeline of the deep network for person re-identification. $\boldsymbol{X}$ denotes the input pedestrian images.



(d) The pipeline of teacher-student network for knowledge distillation. $\boldsymbol{X}$ denotes the input images.

Figure 1. The pipeline of three applications we consider: (a) few-shot learning, (b) zero-shot learning , (c) person re-identification and (d) knowledge distillation.

for the Conv-4 and ResNet-18 backbones are 1600 and 512 dimensions. We use the Adam [6] optimizer with default momentum values (*i.e.*, $[\beta_1, \beta_2]$ = [0.9, 0.999]). Our network is trained for 400 epochs with each epoch sampling 100 episodes. In the *mini*ImageNet dataset, the initial learning rate is initialized to 5×10^{-3} for Conv-4 and 10^{-3} for ResNet-18. At the 80-th epoch, the learning rate is decayed by 0.5. In the CUB dataset, the initial learning rate is 10^{-3} for both two backbones. At the 80-th epoch, the learning rate is decayed by 0.8 for Conv-4 and 0.5 for ResNet-18. In the testing stage, the accuracy is calculated by the mean of 10,000 episodes. In FSL, we set the curvature of the geometry to 0.01.

Loss Details. In the training phase, each episode is composed of a support set $\mathcal{S} = \{(\{\boldsymbol{s}_{i,1}, \ldots, \boldsymbol{s}_{i,K}\}, l_i) : i = 1, \ldots, N\}$ and a query set $\mathcal{Q} = \{(\boldsymbol{q}_i, l_i) : i = 1, \ldots, N\}$. The prototype of each class is computed by $\hat{\boldsymbol{s}}_i = \frac{1}{K}\sum_{j=1}^{K} \boldsymbol{s}_{i,j}$. Then the prototypical network (ProtoNet) formulates the loss function as:

$$\mathcal{L}_{\text{fsl}}^{E} = -\frac{1}{N_q}\sum_{i=1}^{N_q} \log\Big(\frac{\exp(-\|\boldsymbol{q}_i - \hat{\boldsymbol{s}}^*\|)}{\sum_{j=1}^{N}\exp(-\|\boldsymbol{q}_i - \hat{\boldsymbol{s}}_j\|)}\Big), \quad (11)$$

where $\boldsymbol{q}_i$ and $\hat{\boldsymbol{s}}^*$ share the same label, and N_q is the number of query samples in one episode.

Noted that $\boldsymbol{q}_i, \hat{\boldsymbol{s}}_i \in \mathbb{R}^n$ for the vanilla ProtoNet, thus the distance used in Eq. (11) is the L_2 distance. Then in the hyperbolic version (*i.e.*, Hyper ProtoNet), where $\boldsymbol{q}_i, \hat{\boldsymbol{s}}_i \in \mathbb{D}_c^n$, the loss is further formulated as:

$$\mathcal{L}_{\text{fsl}}^{H} = -\frac{1}{N_q}\sum_{i=1}^{N_q} \log\Big(\frac{\exp(-d_c(\boldsymbol{q}_i, \hat{\boldsymbol{s}}^*))}{\sum_{j=1}^{N}\exp(-d_c(\boldsymbol{q}_i, \hat{\boldsymbol{s}}_j))}\Big), \quad (12)$$

where d_c is the geodesic distance in the Poincaré ball.

We further plug our kernels in the loss functions, as:

$$\mathcal{L}_{\text{fsl}}^{K} = -\frac{1}{N_q}\sum_{i=1}^{N_q} \log\Big(\frac{g(k(\boldsymbol{q}_i, \hat{\boldsymbol{s}}^*))}{\sum_{j=1}^{N} g(k(\boldsymbol{q}_i, \hat{\boldsymbol{s}}_j))}\Big), \quad (13)$$

where $k(\cdot,\cdot)$ indicates the kernel, and $\boldsymbol{q}_i, \hat{\boldsymbol{s}}_i \in \mathbb{D}_c^n$. Here, $g(\cdot)$ is exp mapping if $k(\cdot,\cdot)$ is non-exponential type kernels. Otherwise, $g(\cdot)$ is the identity mapping. Table 6 shows the detailed formulations of loss functions with the proposed kernels.

5.2. Zero-shot Learning

Zero-shot learning (ZSL) aims to identify objects that are unseen during the training phase. Formally, suppose we have a seen set $\mathcal{D}^s$ and an unseen set $\mathcal{D}^u$. Specifically, the seen set, $\mathcal{D}^s = \{(\boldsymbol{v}_i^s, l_i^s, \boldsymbol{a}_i^s), i = 1, \ldots, N^s\}$,

contains the visual feature $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$, the semantic feature $\boldsymbol{a}_i \in \mathbb{R}^{d_a}$ for the seen class $l_i^s \in L^s$. Similarly, the unseen set, $\mathcal{D}^u = \{(\boldsymbol{v}_i^u, l_i^u, \boldsymbol{a}_i^u), i = 1, \ldots, N^u\}$, also contains unseen visual feature $\boldsymbol{v}_i^u$, unseen semantic feature $\boldsymbol{a}_i^u$ with the unseen class $l_i^u \in L^u$. It is noted that L^s and L^u should be disjoint, *i.e.*, $L^s \cap L^u = \emptyset$. The pipeline of the network for ZSL is illustrated in Fig. 1(b).

Implementation Details. We implement our methods for zero-shot learning (ZSL) in the PyTorch machine learning package [7]. The visual features are extracted from ResNet-101, which is pre-trained on ImageNet. The feature dimension is 2048. Following the standard protocol in ZSL, the visual features are kept fixed during training. During training, we sample randomly for a mini-batch. We use a simple two layers' MLP (*i.e.*, $\boldsymbol{e}(\cdot)$) to embed the semantic features. The MLP receives the semantic features as input and produces 2048-dimensional embedding features. The dimension of hidden layer is 1200. We use the Adam [6] optimizer with default momentum values (*i.e.*, [β_1, β_2] = [0.9, 0.999]). We choose 5×10^{-3} as the initial learning rate. The network is trained for 100 epochs for SUN and CUB datasets, and 30 epochs for AWA1 and AWA2 datasets. In ZSL, we set the curvature of the geometry to 1.

Loss Details. In the training phase, we randomly sample N_b seen visual features as $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{N_b}\}$. All the semantic features are projected to the visual space, denoted by $\boldsymbol{E} = \{e(\boldsymbol{a}_1), \ldots, e(\boldsymbol{a}_{|L^s|})\}$, where $|L^s|$ denotes the number of seen classes in the training set. In our implementation, the embedding function (*i.e.*, $e(\cdot)$) is a simple two layer MLP, with each layer stacking the linear transformation, ReLU activation and batch normalization. Then the network is trained by the following cross-entropy type loss:

$$\mathcal{L}_{\text{zsl}} = -\frac{1}{N_b}\sum_{i=1}^{N_b} \log\Big(\frac{\exp\big(-\|(e(\boldsymbol{a}^*) - \boldsymbol{v}_i\|\big)}{\sum_{j=1}^{|L^s|}\exp\big(-\|e(\boldsymbol{a}_j) - \boldsymbol{v}_i\|\big)}\Big), \tag{14}$$

where $\boldsymbol{a}^*$ shares the same label with $\boldsymbol{v}_i$. Note that the baseline network is conducted on Euclidean spaces (*i.e.*, $e(\boldsymbol{a}), \boldsymbol{v} \in \mathbb{R}^n$).

Then in our work, the kernelized loss function for the hyperbolic representations (*i.e.*, $e(\boldsymbol{a}), \boldsymbol{v} \in \mathbb{D}_c^n$) can be modified as:

$$\mathcal{L}_{\text{zsl}}^K = -\frac{1}{N_b}\sum_{i=1}^{N_b} \log\Big(\frac{g\big(k((e(\boldsymbol{a}^*), \boldsymbol{v}_i)\big)}{\sum_{j=1}^{|L^s|} g\big(k(e(\boldsymbol{a}_j), \boldsymbol{v}_i)\big)}\Big), \tag{15}$$

where $k(\cdot,\cdot)$ indicates the kernel. Here, $g(\cdot)$ is exp mapping if $k(\cdot,\cdot)$ is non-exponential type kernels. Otherwise, $g(\cdot)$ is the identity mapping. Table 6 shows the detailed formulations of loss functions with the proposed kernels.

5.3. Person Re-identification

Person re-identification (re-ID) aims to retrieve correct person images from a gallery dataset for the query person of interest. The goal of training a re-ID machine is to learn an embedding space, where the intra- (or inter-) person variance is minimized (or maximized). The feature extractor is trained by a classification task. In the inference phase, the penultimate layer of the network is used as a feature representation for the unseen person. The pipeline of the deep network for person re-ID is shown in Fig. 1(c).

Implementation Details. We implement our methods on person re-ID in the PyTorch machine learning package [7]. The size of the input image is cropped to 256×128. Each mini-batch is sampled randomly with the size of 64. The ResNet-50 first generates 2048-dimensional features and a following fully-connected (FC) layer embeds the feature to a final representation (*i.e.*, $\boldsymbol{f}$). An additional FC layer is further used to predict the person's identity. We use the Adam [6] optimizer with default momentum values (*i.e.*, [β_1, β_2] = [0.9, 0.999]). The network is trained for 300 epochs. The initial learning rate is 5×10^{-4}. In the 50-th, 100-th and 150-th epoch, the learning rate is decayed by 0.1. We report the performance of the trained network at the 300-th epoch. We do not apply extra data augmentation techniques and re-ranking to boost the result in the testing stage. It is also worth noting that we apply the identical training strategy and hyper-parameter for Market-1501 and DukeMTMC-reID datasets. In person re-ID, we set the curvature of the geometry to 0.01.

Loss Details. Given a person image with associated identity (*i.e.*, y), the network first extracts its appearance representation (*i.e.*, $\boldsymbol{f} \in \mathbb{R}^n$). The a fully connected layer (*i.e.*, $\boldsymbol{W}$) is applied to predict the identity of person and a softmax function is used to normalize the output (*i.e.*, $\boldsymbol{p} = \boldsymbol{W}^\top \boldsymbol{f}$). The probability of the person $\boldsymbol{f}$ w.r.t. its label y is denoted by $p(y|\boldsymbol{f}) = \frac{\exp(\langle \boldsymbol{w}^*, \boldsymbol{f}\rangle)}{\sum_j^N \exp(\langle \boldsymbol{w}_j, \boldsymbol{f}\rangle)}$. The training will minimize the negative log-probability, as

$$\begin{aligned}\mathcal{L}_{\text{reid}} &= -\log\big(p(y|\boldsymbol{f})\big) \\ &= -\log\Big(\frac{\exp(\langle \boldsymbol{w}^*, \boldsymbol{f}\rangle)}{\sum_j^N \exp(\langle \boldsymbol{w}_j, \boldsymbol{f}\rangle)}\Big).\end{aligned} \tag{16}$$

The kernelized loss function can further be obtained:

$$\mathcal{L}_{\text{reid}}^K = -\log\Big(\frac{g(k(\boldsymbol{w}^*, \boldsymbol{f}))}{\sum_j^N g(k(\boldsymbol{w}_j, \boldsymbol{f}))}\Big), \tag{17}$$

where $k(\cdot,\cdot)$ indicates the kernel and $\boldsymbol{w}^*, \boldsymbol{f} \in \mathbb{D}_c^n$. Here, $g(\cdot)$ is exp mapping if $k(\cdot,\cdot)$ is non-exponential type kernels. Otherwise, $g(\cdot)$ is the identity mapping. Table 6 shows the detailed formulations of loss functions with the proposed kernels.

5.4. Knowledge Distillation

Implementation Details. We implement our methods on knowledge distillation (see Fig. 1(d)) in the PyTorch machine learning package [7]. Table 5 illustrates the CNN architectures for teacher and student networks. The size of the input image is fixed to 32×32. We randomly sample 128 images for each mini-batch. The ResNet-20 is first trained as a teacher network. Then the output of ResNet-20 is used as ground truth to train the student network. We use the SGD optimizer with 0.9 momentum value. The network is trained for 200 epochs. The learning rate is initialized to 0.1 and decayed at the 100-th, 150-th epochs by a factor of 0.1. We report the result at the 200-th epoch. In KD, the curvature of hyperbolic geometry is set to 0.001 for all experiments. Also, we stay consistent with the popular choice for the temperature: $T = 4$ across all experiments [2, 8]

Table 5. Network architecture for knowledge distillation on CIFAR-10 and CIFAR-100 datasets. PNs indicates the parameter numbers.

Conv layer	Teacher ResNet-20	Student 4-layer CNN
Conv_1	conv, 3×3, 16	conv, 3×3, 16
Conv_2	$\begin{bmatrix}\text{conv}, 1 \times 1, 16\\ \text{conv}, 3 \times 3, 16\end{bmatrix} \times 3$	conv, 3×3, 16
Conv_3	$\begin{bmatrix}\text{conv}, 1 \times 1, 32\\ \text{conv}, 3 \times 3, 32\end{bmatrix} \times 3$	conv, 3×3, 32
Conv_4	$\begin{bmatrix}\text{conv}, 1 \times 1, 64\\ \text{conv}, 3 \times 3, 64\end{bmatrix} \times 3$	conv, 3×3, 64
CIFAR-10 / 100	global average pooling, 10 / 100-classes, softmax	
PNs ($\times 10^{-6}$)	0.272 / 0.278	0.027 / 0.032

Loss Details. In the knowledge distillation task, the teacher network generates the prediction scores $\boldsymbol{g} = [g_1, g_2, \ldots, g_N]^\top$ for a input image (*e.g.*, $\boldsymbol{X}$). Then the student network first extract the feature vector of input image as $\boldsymbol{f} \in \mathbb{R}^n$, and a fully connected layer $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N]$ is used to produce the predication, *i.e.* $\boldsymbol{p} = \text{softmax}(\boldsymbol{W}^\top \boldsymbol{f})$ and each p_i is given by:

$$p_i = \frac{\exp(\langle \boldsymbol{w}_i, \boldsymbol{f} \rangle / T)}{\sum_{j=1}^{N} \exp(\langle \boldsymbol{w}_j, \boldsymbol{f} \rangle / T)}, \tag{18}$$

where T is the temperature. Then the KD loss is:

$$\begin{aligned} \mathcal{L}_{kd} &= -\sum_{i=1}^{N} g_i \log(p_i) \\ &= -\sum_{i=1}^{N} g_i \log\Big(\frac{\exp(\langle \boldsymbol{w}_i, \boldsymbol{f} \rangle / T)}{\sum_{j=1}^{N} \exp(\langle \boldsymbol{w}_j, \boldsymbol{f} \rangle / T)}\Big). \end{aligned} \tag{19}$$

The kernelized KD loss for the hyperbolic representation $\boldsymbol{f} \in \mathbb{D}_c^n$ can be obtained as:

$$\mathcal{L}_{kd}^K = -\sum_{i=1}^{N} g_i \log\Big(\frac{g(k(\boldsymbol{w}_i, \boldsymbol{f})/T)}{\sum_{j=1}^{N} g(k(\boldsymbol{w}_j, \boldsymbol{f})/T)}\Big), \tag{20}$$

where $k(\cdot, \cdot)$ indicates the kernel and $\boldsymbol{w}_i, \boldsymbol{f} \in \mathbb{D}_c^n$. Here, $g(\cdot)$ is exp mapping if $k(\cdot, \cdot)$ is non-exponential type kernels. Otherwise, $g(\cdot)$ is the identity mapping. Table 6 shows the detailed formulations of loss functions with the proposed kernels.

6. Details for the Indefinite Kernel

Here, we give the detail formulation of the indefinite kernel used in the main paper. Let $k_E(\cdot, \cdot)$ be a Euclidean inner product kernel, satisfying $K_E(\boldsymbol{x}, \boldsymbol{x}) < 1$ for all $\|\boldsymbol{x}\| \leq 1$. Given two points in the Minkowski spaces, also known as hyperboloid model, (*i.e.*, $\boldsymbol{r}_i, \boldsymbol{r}_j \in \mathbb{M}^n$), then the kernel is defined as:

$$k^M(\boldsymbol{r}_i, \boldsymbol{r}_j) = \frac{k_E(g(\boldsymbol{r}_i), g(\boldsymbol{r}_j)) - 1}{\sqrt{\big(1 - k_E(g(\boldsymbol{r}_i), g(\boldsymbol{r}_i))\big)\big(1 - k_E(g(\boldsymbol{r}_j), g(\boldsymbol{r}_j))\big)}}, \tag{21}$$

where $g(\cdot)$ maps the point in the hyperboloid model to the Klein ball model.

Then we are going to explain the details to implement the indefinite kernel. In our implementation, the network first generates features in the poincaré ball, *i.e.*, $\boldsymbol{z} \in \mathbb{D}_c^n$. Then we can further project the point to Klein mode, as:

$$\mathbb{D}_c^n \to \mathbb{K}_c^n : h(\boldsymbol{z}) = \frac{2\boldsymbol{z}}{1 + c\|\boldsymbol{z}\|^2}. \tag{22}$$

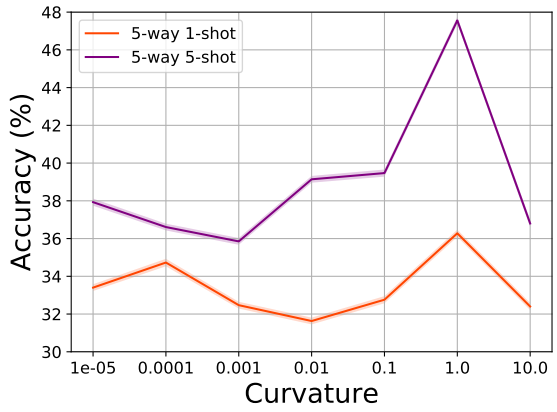
Since the Hyperboloid model and poincaré ball are isometric, we can obtain the indefinite kernel as:

$$k^{in}(\boldsymbol{z}_i, \boldsymbol{z}_j) = \frac{K_E(h(\boldsymbol{z}_i), h(\boldsymbol{z}_j) - 1)}{\sqrt{\big(1 - k_E(h(\boldsymbol{z}_i), h(\boldsymbol{z}_i))\big)\big(1 - k_E(h(\boldsymbol{z}_j), h(\boldsymbol{z}_j))\big)}}. \tag{23}$$

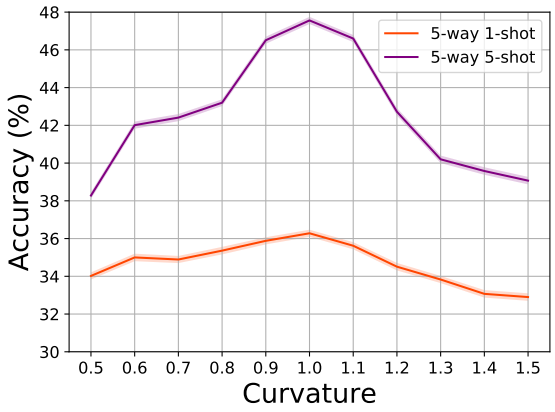
In this study, we also conduct experiments to choose a proper curvature (*i.e.*, c) for the indefinite kernel on the few-shot learning task. Here, we use the setting of 5-way 1-shot and 5-way 5-shot on the FC100 dataset across various curvature values. Fig. 2 shows that the performance of the indefinite kernel reaches the peak. Hence, the curvature is set to 1 for the indefinite kernel in the main paper.

7. Robustness of kernel machines

We evaluate the robustness of the proposed kernel machines. We compare our methods against the original hyperbolic embeddings with various curvature of hyperbolic geometry (*i.e.*, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1} and 1). This study is conducted on *mini*ImageNet for a 5-way 5-shot setting under Conv-4 backbone. Fig. 3 indicates that: **(1)** the discrimination of embedding in hyperbolic space is sensitive to the curvature of geometry. Increasing the curvature will degrade the network performance. **(2)** The hyperbolic tangent kernel, hyperbolic RBF kernel and hyperbolic binomial kernel are robust against the curvature of geometry. **(3)** The hyperbolic Laplace kernel is also sensitive to the curvature. However, it degrades gracefully. For example, the



(a) In this plot, we set the curvature values to 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1} and 1.



(b) In this plot, the curvature varies from 0.5 to 1.5.

Figure 2. The performance comparison for the indefinite kernel on the few-shot learning task. We evaluate the indefinite kernel on both 5-way 1-shot and 5-way 5-shot settings and use Conv-4 as the backbone network.

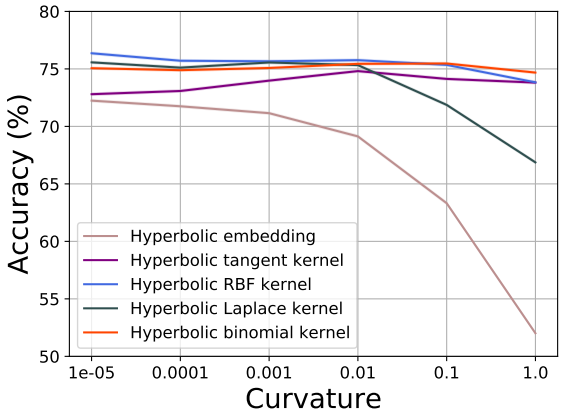


Figure 3. The performance comparison on *mini*ImageNet with various curvature of the hyperbolic geometry. The experiment is conducted on a few-shot learning task under the 5-way 5-shot setting and we use Conv-4 as the backbone network.

hyperbolic Laplace kernel decreases 10% accuracy, while the embedding in hyperbolic space losses more than 20% accuracy.

8. Good Practice of Employing Hyperbolic Geometry

In the main paper, we explained our work generates the hyperbolic representations in the Poincaré ball. In this supplementary material, we show the schematic comparison between existing works [1, 5] and our work in employing the geometry constraint in deep networks in Fig. 4. In Fig. 4(a), the mapping $\Gamma_0(\cdot)$ projects the point $\boldsymbol{x}$ in the Euclidean spaces $\mathbb{R}^n$ to the hyperbolic spaces $\mathbb{H}^n$ is given by:

$$\Gamma_0(\boldsymbol{x}) = \tanh(\sqrt{c}\|\boldsymbol{x}\|)\frac{\boldsymbol{x}}{\sqrt{c}\|\boldsymbol{x}\|}. \tag{24}$$

In Fig. 4(b), we generate the feature vector in the hyperbolic space and map the vector to the reproducing Kernel Hilbert space $\mathcal{H}$ by the proposed kernels.

We also visualize the features learned from the pipeline in Fig. 4(a) and Fig. 4(b), respectively. Both 2-D embeddings are trained by a small CNN for MNIST dataset. Fig. 5(a) and Fig. 5(b) visualize the features trained by the network under the constraint in Fig. 4(a), and in Fig. 4(b), respectively. From Fig. 5, we can find in both pipelines, most of the samples are distributed near the boundary, similar observation can be found in [5]. In addition, as compared to the embeddings learned from existing geometry constraint (see Fig. 5(a)), the feature embeddings from our practice (see Fig. 5(b)) is clustered evenly near the boundary of the Poincaré ball, showing the good practice of using the hyperbolic geometry in our work. In Fig. 6, we further visualization the feature embeddings for each class, which also verifies our observations. For example, in the blue, green and fuchsia classes (Fig. 6(e) vs. Fig. 6(f), Fig. 6(g) vs. Fig. 6(h) and Fig. 6(o) vs. Fig. 6(p)), the embeddings from our practice are more compact then those from existing works.

References

[1] Jiaxin Chen, Jie Qin, Yuming Shen, Li Liu, Fan Zhu, and Ling Shao. Learning attentive and hierarchical representations for 3d shape recognition. In *ECCV*, 2020. 7, 9

Table 6. Summary of the loss functions using the proposed positive definite kernels in hyperbolic spaces. g-Hyperbolic Laplace kernel indicates the generalized hyperbolic Laplace kernel.

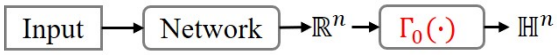
Kernel	Kernelized Loss Functions
Few-shot Learning	
$f_{\mathbb{D}}(\boldsymbol{z}) = \tanh^{-1}(\sqrt{c}\|\boldsymbol{z}\|)\frac{\boldsymbol{z}}{\sqrt{c}\|\boldsymbol{z}\|},\ c>0$ and $\boldsymbol{q}, \hat{\boldsymbol{s}} \in \mathbb{D}_c^n$	
Hyperbolic tangent kernel	$\mathcal{L}_{\text{fsl}}^{\text{tan}} = -\frac{1}{N_q}\sum_{i=1}^{N_q} \log\big(\frac{\exp(\langle f_{\mathbb{D}}(\boldsymbol{q}_i), f_{\mathbb{D}}(\hat{\boldsymbol{s}}^*)\rangle)}{\sum_{j=1}^{N}\exp(\langle f_{\mathbb{D}}(\boldsymbol{q}_i), f_{\mathbb{D}}(\hat{\boldsymbol{s}}_j)\rangle)}\big)$
Hyperbolic RBF kernel	$\mathcal{L}_{\text{fsl}}^{\text{rbf}} = -\frac{1}{N_q}\sum_{i=1}^{N_q} \log\big(\frac{\exp(-\xi\|f_{\mathbb{D}}(\boldsymbol{q}_i) - f_{\mathbb{D}}(\hat{\boldsymbol{s}}^*)\|^2)}{\sum_{j=1}^{N}\exp(-\xi\|f_{\mathbb{D}}(\boldsymbol{q}_i) - f_{\mathbb{D}}(\hat{\boldsymbol{s}}_j)\|^2)}\big)$
Hyperbolic Laplace kernel	$\mathcal{L}_{\text{fsl}}^{\text{lap}} = -\frac{1}{N_q}\sum_{i=1}^{N_q} \log\big(\frac{\exp(-\xi\|f_{\mathbb{D}}(\boldsymbol{q}_i) - f_{\mathbb{D}}(\hat{\boldsymbol{s}}^*)\|)}{\sum_{j=1}^{N}\exp(-\xi\|f_{\mathbb{D}}(\boldsymbol{q}_i) - f_{\mathbb{D}}(\hat{\boldsymbol{s}}_j)\|)}\big)$
Hyperbolic binomial kernel	$\mathcal{L}_{\text{fsl}}^{\text{bin}} = -\frac{1}{N_q}\sum_{i=1}^{N_q} \log\big(\frac{\exp\left(1-(\langle f_{\mathbb{D}}(\boldsymbol{q}_i), f_{\mathbb{D}}(\hat{\boldsymbol{s}}^*)\rangle)^{-\alpha}\right)}{\sum_{j=1}^{N}\exp\left((1-\langle f_{\mathbb{D}}(\boldsymbol{q}_i), f_{\mathbb{D}}(\hat{\boldsymbol{s}}_j)\rangle)^{-\alpha}\right)}\big)$
Zero-shot Learning	
$f_{\mathbb{D}}(\boldsymbol{z}) = \tanh^{-1}(\sqrt{c}\|\boldsymbol{z}\|)\frac{\boldsymbol{z}}{\sqrt{c}\|\boldsymbol{z}\|},\ c>0$ and $e(\boldsymbol{a}), \boldsymbol{v} \in \mathbb{D}_c^n$	
Hyperbolic tangent kernel	$\mathcal{L}_{\text{zsl}}^{\text{tan}} = -\frac{1}{N_b}\sum_{i=1}^{N_b} \log\big(\frac{\exp\left(\langle f_{\mathbb{D}}(e(\boldsymbol{a}^*)), f_{\mathbb{D}}(\boldsymbol{v}_i)\rangle\right)}{\sum_{j=1}^{\lvert L^s\rvert}\exp\left(\langle f_{\mathbb{D}}(e(\boldsymbol{a}_j)), f_{\mathbb{D}}(\boldsymbol{v}_i)\rangle\right)}\big)$
Hyperbolic RBF kernel	$\mathcal{L}_{\text{zsl}}^{\text{rbf}} = -\frac{1}{N_b}\sum_{i=1}^{N_b} \log\big(\frac{\exp\left(-\xi\|f_{\mathbb{D}}(e(\boldsymbol{a}^*)) - f_{\mathbb{D}}(\boldsymbol{v}_i)\|^2\right)}{\sum_{j=1}^{\lvert L^s\rvert}\exp\left(-\xi\|f_{\mathbb{D}}(e(\boldsymbol{a}_j)) - f_{\mathbb{D}}(\boldsymbol{v}_i)\|^2\right)}\big)$
Hyperbolic Laplace kernel	$\mathcal{L}_{\text{zsl}}^{\text{lap}} = -\frac{1}{N_b}\sum_{i=1}^{N_b} \log\big(\frac{\exp\left(-\xi\|f_{\mathbb{D}}(e(\boldsymbol{a}^*)) - f_{\mathbb{D}}(\boldsymbol{v}_i)\|\right)}{\sum_{j=1}^{\lvert L^s\rvert}\exp\left(-\xi\|f_{\mathbb{D}}(e(\boldsymbol{a}_j)) - f_{\mathbb{D}}(\boldsymbol{v}_i)\|\right)}\big)$
Hyperbolic binomial kernel	$\mathcal{L}_{\text{zsl}}^{\text{bin}} = -\frac{1}{N_b}\sum_{i=1}^{N_b} \log\big(\frac{\exp\left(\left(1-\langle f_{\mathbb{D}}(e(\boldsymbol{a}^*)) - f_{\mathbb{D}}(\boldsymbol{v}_i)\rangle\right)^{-\alpha}\right)}{\sum_{j=1}^{\lvert L^s\rvert}\exp\left(\left(1-\langle f_{\mathbb{D}}(e(\boldsymbol{a}_j)) - f_{\mathbb{D}}(\boldsymbol{v}_i)\rangle\right)^{-\alpha}\right)}\big)$
Person Re-identification	
$f_{\mathbb{D}}(\boldsymbol{z}) = \tanh^{-1}(\sqrt{c}\|\boldsymbol{z}\|)\frac{\boldsymbol{z}}{\sqrt{c}\|\boldsymbol{z}\|},\ c>0$ and $\boldsymbol{f} \in \mathbb{D}_c^n, \boldsymbol{w} \in \mathbb{R}^n$	
Hyperbolic tangent kernel	$\mathcal{L}_{\text{reid}}^{\text{tan}} = -\log\big(\frac{\exp(\langle \boldsymbol{w}^*, f_{\mathbb{D}}(\boldsymbol{f})\rangle)}{\sum_{j=1}^{N}\exp(\langle \boldsymbol{w}_j, f_{\mathbb{D}}(\boldsymbol{f})\rangle)}\big)$
Hyperbolic RBF kernel	$\mathcal{L}_{\text{reid}}^{\text{rbf}} = -\log\big(\frac{\exp(-\xi\|\boldsymbol{w}^* - f_{\mathbb{D}}(\boldsymbol{f})\|^2)}{\sum_{j=1}^{N}\exp(-\xi\|\boldsymbol{w}_j - f_{\mathbb{D}}(\boldsymbol{f})\|^2)}\big)$
g-Hyperbolic Laplace kernel	$\mathcal{L}_{\text{reid}}^{\text{glap}} = -\log\big(\frac{\exp(-\xi\|\boldsymbol{w}^* - f_{\mathbb{D}}(\boldsymbol{f})\|^{2\alpha})}{\sum_{j=1}^{N}\exp(-\xi\|\boldsymbol{w}_j - f_{\mathbb{D}}(\boldsymbol{f})\|^{2\alpha})}\big)$
Hyperbolic binomial kernel	$\mathcal{L}_{\text{reid}}^{\text{bin}} = -\log\big(\frac{\exp\left((1-\langle \boldsymbol{w}^*, f_{\mathbb{D}}(\boldsymbol{f})\rangle)^{-\alpha}\right)}{\sum_{j=1}^{N}\exp\left((1-\langle \boldsymbol{w}_j, f_{\mathbb{D}}(\boldsymbol{f})\rangle)^{-\alpha}\right)}\big)$
Knowledge Distillation	
$f_{\mathbb{D}}(\boldsymbol{z}) = \tanh^{-1}(\sqrt{c}\|\boldsymbol{z}\|)\frac{\boldsymbol{z}}{\sqrt{c}\|\boldsymbol{z}\|},\ c>0$ and $\boldsymbol{f} \in \mathbb{D}_c^n, \boldsymbol{w} \in \mathbb{R}^n$	
Hyperbolic tangent kernel	$\mathcal{L}_{\text{kd}}^{\text{tan}} = -\sum_{i=1}^{N} g_i \log\big(\frac{\exp(\langle \boldsymbol{w}_i, f_{\mathbb{D}}(\boldsymbol{f})\rangle/T)}{\sum_{j=1}^{N}\exp(\langle \boldsymbol{w}_j, f_{\mathbb{D}}(\boldsymbol{f})\rangle/T)}\big)$
Hyperbolic RBF kernel	$\mathcal{L}_{\text{kd}}^{\text{rbf}} = -\sum_{i=1}^{N} g_i \log\big(\frac{\exp(-\xi\|\boldsymbol{w}_i - f_{\mathbb{D}}(\boldsymbol{f})\|^2/T)}{\sum_{j=1}^{N}\exp(-\xi\|\boldsymbol{w}_j - f_{\mathbb{D}}(\boldsymbol{f})\|^2/T)}\big)$
g-Hyperbolic Laplace kernel	$\mathcal{L}_{\text{kd}}^{\text{lap}} = -\sum_{i=1}^{N} g_i \log\big(\frac{\exp(-\xi\|\boldsymbol{w}_i - f_{\mathbb{D}}(\boldsymbol{f})\|^{2\alpha}/T)}{\sum_{j=1}^{N}\exp(-\xi\|\boldsymbol{w}_j - f_{\mathbb{D}}(\boldsymbol{f})\|^{2\alpha}/T)}\big)$
Hyperbolic binomial kernel	$\mathcal{L}_{\text{kd}}^{\text{bin}} = -\sum_{i=1}^{N} g_i \log\big(\frac{\exp\left((1-\langle \boldsymbol{w}_i, f_{\mathbb{D}}(\boldsymbol{f})\rangle)^{-\alpha}/T\right)}{\sum_{j=1}^{N}\exp\left((1-\langle \boldsymbol{w}_j, f_{\mathbb{D}}(\boldsymbol{f})\rangle)^{-\alpha}/T\right)}\big)$

[2] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *ICCV*, October 2019. 6

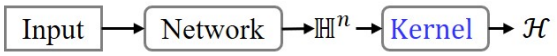
[3] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object Detection with Discriminatively Trained Part-Based Models. *PAMI*, 2010. 3

[4] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation average. *IJCV*, 2012. 1

[5] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova,



(a) Geometry constraints in existing works [1, 5].



(b) Geometry constraints in our work.

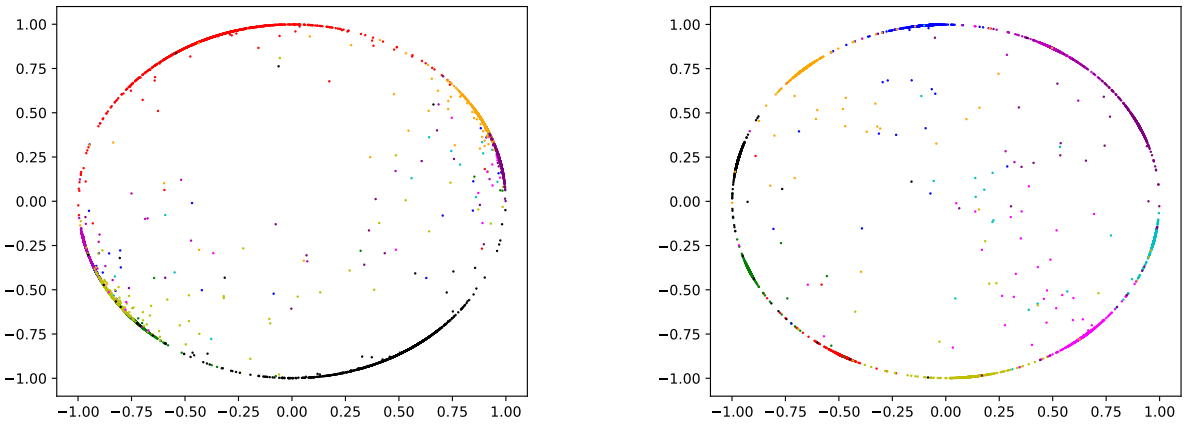
Figure 4. Schematic comparison between existing works and our work in employing constraints from the hyperbolic geometry.

Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *CVPR*, 2020. 7, 9

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4, 5

[7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NeurIPS*, 2017. 3, 5, 6

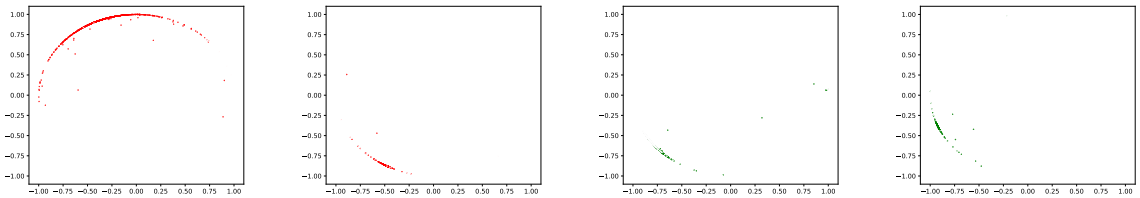
[8] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *ICLR*, 2017. 6



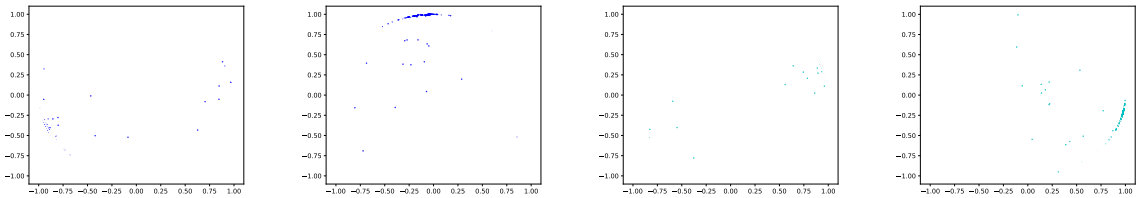
(a)

(b) Ours

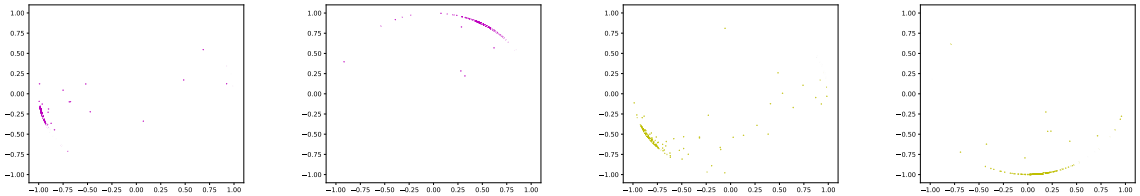
Figure 5. Visualization of the features in hyperbolic spaces learned for MNIST. (a): Hyperbolic embeddings trained by the pipeline in Fig. 4(a). (b):Hyperbolic embeddings trained by the pipeline in Fig. 4(b). Here, we use the hyperbolic tangent kernel. In both cases, most of embeddings are distributed near the boundary of the Poincaré ball. Also, in our practice (see (b)), the features from the same class are clustered evenly and compactly. Best viewed in color.



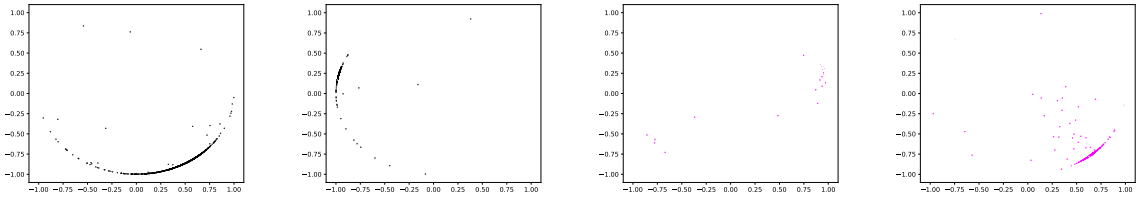
(a) (b) Ours (c) (d) Ours



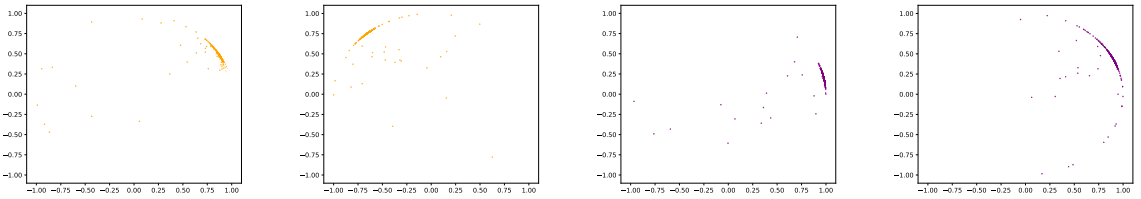
(e) (f) Ours (g) (h) Ours



(i) (j) Ours (k) (l) Ours



(m) (n) Ours (o) (p) Ours



(q) (r) Ours (s) (t) Ours

Figure 6. Visualization of the feature embeddings for each class on MNIST dataset. In each pair, the left one is the visualization for the pipeline in Fig. 4(a) and the right one is the visualization for the pipeline in Fig. 4(b). Best viewed in color.