Hyperbolic Prototypical Entailment Cones for Image Classification

Samuele Fonio
University of Turin

Roberto Esposito
University of Turin

Marco Aldinucci
University of Turin

Abstract

Non-Euclidean geometries have garnered significant research interest, particularly in their application to Deep Learning. In this paper we focus on hyperbolic manifolds and introduce a novel framework, Hyperbolic Prototypical Entailment Cones (HPEC). The core innovation of HPEC lies in utilizing angular relationships, rather than traditional distance metrics, to more effectively capture the similarity between data representations and their corresponding prototypes. This is achieved by leveraging hyperbolic entailment cones, a mathematical construct particularly suited for embedding hierarchical structures in the Poincaré ball, along with a novel *Back-clip* mechanism. Our experimental results demonstrate that this approach significantly enhances performance in high-dimensional embedding spaces. To substantiate these findings, we evaluate HPEC on four datasets and various embedding dimensions, consistently surpassing state-of-the-art methods in Prototype Learning.

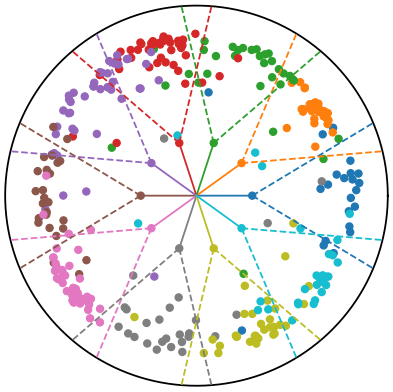


Figure 1: Illustration of the hyperbolic entailment cones on MNIST with embedding dimension 2. The embeddings are colored with respect to the true class.

1 INTRODUCTION

The pursuit of optimal metrics during the learning process is a longstanding and fundamental challenge in deep learning (DL). Selecting an appropriate metric is critical, as it shapes the relationships between data points and influences the definition of classification boundaries, which are key to effective learning.

The recent interest in non-Euclidean geometries has opened new avenues for research in the machine learning (ML) community. These alternative geometries offer novel approaches to defining similarity metrics, which may provide more expressive representations of the underlying structure of data, potentially improving learning outcomes and generalization performance.

Among the non-Euclidean geometries of major interest, the hyperbolic geometry seems to be the most promising one (Mettes et al., 2024). Hyperbolic spaces are claimed to represent hierarchical data with arbitrarily low distortion (Sala et al., 2018), providing the best geometry to embed data distributions that are intrinsically hierarchical. In fact, in recent years several works extended ML paradigms to work in hyperbolic spaces, e.g. Support Vector Machines (Cho et al., 2019), Neural Networks (Ganea et al., 2018a; Ryohei et al., 2021; Guo et al., 2022), Neural networks equipped with Attention Mechanism (Gulcehre et al., 2018) and Vision Transformers (Ermolov et al., 2022). While there are some fields in which data are explicitly hierarchical and, thus, naturally lend themselves to be processed in hyperbolic spaces (e.g. Natural Language Processing and Graph Representation Learning), this is not the case for Computer Vision (CV), where the hierarchical structure is usually hidden. Nonetheless, several recent

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

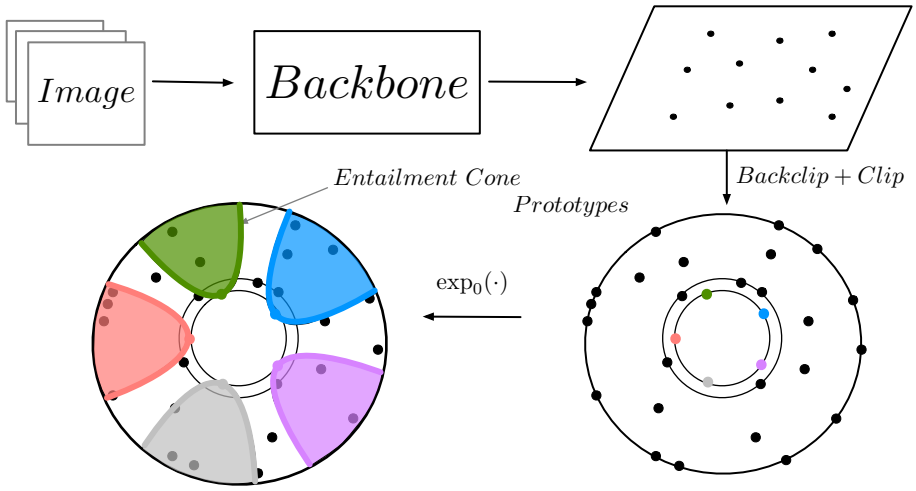


Figure 2: Illustration of HPEC. Specifically, we show the embedding phase, the application of Backclip and Clip, and the final projection into the embedding space, i.e. the Poincaré ball. Notice that the exponential map shrinks the norm of the embeddings, as we are going to explain in the following.

works have provided evidence that images are intrinsically hierarchical (Khrulkov et al., 2020; Bdeir et al., 2024), which spawned a growing interest in applying hyperbolic spaces in CV (Mettes et al., 2024).

Following this line of research, the CV community recently focused on developing fully hyperbolic neural network architectures (Lensink et al., 2022; van Spengler et al., 2023; Bdeir et al., 2024) and on exploiting metric elements in hyperbolic spaces (Yue et al., 2024; Hamzaoui et al., 2024).

We extend the effort in this last direction, leveraging a ML paradigm that relies *directly* on the use of metric elements: Prototype Learning (PL). PL Snell et al. (2017) is a well-established ML paradigm that leverages a representation of the classes in the embedding space (i.e. the *prototypes*) and the use of distances (or similarities) between data representations and the prototypes to optimize the learning process.

Integrating PL and hyperbolic spaces is a mostly unexplored field in image classification (to the best of our knowledge only Ghadimi Atigh et al. (2021) tackled this challenge), while it is well-studied for downstream tasks, such as few-shot learning (Nickel and Kiela, 2018; Khrulkov et al., 2020; Hamzaoui et al., 2024) and zero-shot learning (Liu et al., 2020).

Based on the foundational and theoretically well-established work by Ganea et al. (2018b), we propose to leverage the use of hyperbolic spaces and PL through *Hyperbolic Entailment Cones*, i.e. the generalization of convex cones on the Poincaré ball. Indeed, we introduce a novel framework, called Hyperbolic Prototypical Entailment Cones (HPEC), which is able to outperform the state-of-the-art in PL for image classification. The core of HPEC's enhanced performance lies in the introduction of an angle-based loss function and a *Backclip* mechanism.

To recap, the contributions of this work are threefold: *i)* we introduce Hyperbolic Prototypical Entailment Cones (HPEC), a framework that leverages hyperbolic entailment cones (depicted in Figure 2); *ii)* we introduce the *Backclip* mechanism, which ensures our framework remains competitive in high-dimensional spaces; *iii)* we surpass the current state-of-the-art in PL, setting a new benchmark for PL in image classification.

2 RELATED WORKS

Prototype Learning Prototype Learning is a well established ML paradigm generalizing learning Vector Quantization machines (Somervuo and Kohonen, 1999) and nearest centroid classifiers (Tibshirani et al., 2002). In the recent literature, the prototypes are defined as centroids of the representations (Snell et al., 2017), positioned a priori (Mettes et al., 2019; Fonio et al., 2023; Long et al., 2020) or learnt during the training of the model (Yang et al., 2018; Landrieu and Garnot, 2021). The choice of updating the prototypes during the training is a key aspect for making PL competitive in large-scale tasks, such as image classification. For example, Landrieu and Garnot (2021) shows leading performances in tasks involving both images and video, leveraging a Euclidean geometry and updating the prototypes, adding hierarchical information through a metric-guided approach. On the other end of the spectrum, some work keep the prototypes fixed and leverage

non-Euclidean geometries to cope with the hierarchical structure present in the data. For example, Mettes et al. (2019) introduces hyperspherical prototypical networks, i.e., a PL system based on non-Euclidean geometry, where prototypes are kept fixed on the boundary of an hypersphere. The importance of adding hierarchical information in this non-Euclidean context is investigated by Fonio et al. (2023) where a metric-guided approach is leveraged to weight the impact the hierarchy. It is worth mentioning that many methods use metric learning while not explicitely referring to prototype learning. In particular *NormFace* by Wang et al. (2017), very similar to using a cosine softmax by Kornblith et al. (2021) is equivalent to using hyperspherical dynamic prototypes.

However, our method extends the effort of using fixed prototypes. Leveraging the use of the hyperbolic entailment cones proposed by Ganea et al. (2018b), we replace the distance employed by the aforementioned methods as metric of the objective function with angles computed using the prototypes as the anchor points, resulting in a strategy that we empirically show to be advantageous in Section 5.

Hyperbolic Representation Learning Hyperbolic manifolds are claimed to represent hierarchical data with arbitrarily low distortion (Sala et al., 2018). Among the hyperbolic models proposed in the literature, we use the Poincaré ball as embedding space aligning with Guo et al. (2022); van Spengler et al. (2023); Khrulkov et al. (2020); Long et al. (2020). Hyperbolic spaces are particularly effective in few-shot learning settings, where they are at the state-of-the-art Khrulkov et al. (2020); Hamzaoui et al. (2024). Recent research also focuses on image-text embeddings (Desai et al., 2023), unsupervised image retrieval (Qiu et al., 2024) and Federated Learning (Liao et al., 2023).

Regarding broader tasks such as image classification, as far as we are aware, only Ghadimi Atigh et al. (2021) addresses this challenge, while Yue et al. (2024) investigates the influence of temperature in contrastive losses. In particular, Ghadimi Atigh et al. (2021) introduces a method with fixed ideal prototypes positioned on the boundary of the Poincaré ball, and introduce the Busemann distance between a point on the manifold and the prototypes. This approach resolves a crucial challenge, as the boundary (and consequently the prototypes) theoretically lie at an infinite distance from any point within the manifold.

Hyperbolic entailment cones were introduced by Ganea et al. (2018b) and applied to graph representations. Our effort in this work is to extend this approach for image classification. In Long et al. (2020) the prototypes are fixed inside the Poincaré ball with a hierarchical-aware method involving the entailment cones, tackling the task of action recognition. However, the learning paradigm is based on hyperbolic distances. We change this paradigm by using hyperbolic entailment cones, showing that through HPEC it is possible to outperform the current state-of-the-art in PL for image classification task.

Table 1: Table of Notation

Symbol	Description
κ	Curvature
$\mathbb{B}^n_\kappa$	Poincaré ball
$\mathcal{T}_x\mathbb{B}^n$	Tangent space
g	Riemannian metric
$\exp^\kappa_O$	Exponential map
ψ	Aperture function (amplitude)
$f(\cdot,\theta)$	Backbone network
π	Prototype
x	Data point
z	Data representation on $\mathbb{B}^n_\kappa$
r^*	Prototype ray
Ξ	Hyperbolic angle
ξ	Margin
ζ	Shrinking factor
ϵ	Backclip margin

3 Background

Definition 1. *A **manifold** $\mathcal{M}$ of dimension n is a topological space such that each point's neighborhood can be locally approximated by the Euclidean space $\mathbb{R}^n$.*

In this paper we are considering the **Poincaré ball**, i.e.:

$$\mathbb{B}^n_\kappa = \{x \in \mathbb{R}^n : \kappa\|x\|^2 < 1\}$$

Definition 2. *Given a point $x \in \mathcal{M}$, the **tangent space** $\mathcal{T}_x\mathcal{M}$ of $\mathcal{M}$ at x is the n-dimensional vector-space, homeomorphic to $\mathbb{R}^n$, built as the first order approximation of $\mathcal{M}$ around x.*

Definition 3. *The **Riemannian metric** is the metric tensor that gives a local notion of angle, length of curves, surface and volume. For a manifold $\mathcal{M}$, a Riemannian metric g_x is a smooth collection of inner products on the associated tangent space: $g_x : \mathcal{T}_x\mathcal{M} \times \mathcal{T}_x\mathcal{M} \to \mathbb{R}$. A **Riemannian manifold** is defined as a manifold equipped with a Riemannian metric g, and is written $(\mathcal{M}, g)$.*

The Riemannian metric of $\mathbb{B}^n$ is given by:

$$g^{\mathbb{B}}_x = \lambda^\kappa_x g^E, \tag{1}$$

with:

$$\lambda^\kappa_x = \frac{1}{1-\kappa\|x\|^2_2}, \tag{2}$$

where $x \in \mathbb{B}^n$, κ is the curvature of the hyperbolic manifold, and $g^E = I_n$ is the Euclidean metric (the identity matrix).

Definition 4. *Given a point $x \in \mathcal{M}$ and a vector $v \in \mathcal{T}_x\mathcal{M}$, the **exponential map** projects v to the manifold $\mathcal{M}$, $\exp_x^\kappa(v) : \mathcal{T}_x\mathcal{M} \to \mathcal{M}$. The projection is obtained by moving the point along the geodesic $\gamma : [0, 1] \to \mathcal{M}$ uniquely defined by $\gamma(0) = x$ and $\gamma'(0) = v$. The projection is defined to be $\exp_x^\kappa(v) = \gamma(1)$. The precise definition of the exponential map depends on the manifold; its inverse function is called the **logarithmic map**, $\log_x^\kappa(\cdot)$.*

If $\mathcal{M} = \mathbb{B}_\kappa^n$ and $x = O$:

$$\exp_O^\kappa(v) = \frac{1}{\sqrt{\kappa}} \tanh(\sqrt{\kappa}\|v\|/2)\frac{v}{\|v\|}. \tag{3}$$

In this work, if not otherwise stated, we assume fixed curvature $\kappa = 1$ and the origin $O = (0, \ldots, 0)$ point as the projection point.

Definition 5. *For any points $A, B, C \in \mathcal{M}$, the angle $\angle ABC$ is the angle between the tangent vectors of the geodesics connecting B with A, and B with C respectively. In the Poincaré ball, the angle between two tangent vectors $u, v \in \mathcal{T}_x\mathbb{B}^n$ is given by:*

$$cos(\angle(u, v)) = \frac{g_x^{\mathbb{B}}(u, v)}{\sqrt{(g_x^{\mathbb{B}}(u, u))}\sqrt{(g_x^{\mathbb{B}}(v, v))}} = \frac{\langle u, v\rangle}{\|u\|\|v\|}, \tag{4}$$

where $\langle\cdot,\cdot\rangle$ is the euclidean scalar product. Notice that the second equality happens since the Euclidean and the hyperbolic geometries are conformal (i.e. the angles are the same in both geometries).

Entailment cones, as described by the foundational work Ganea et al. (2018b), are used to embed hierarchical structures in the Poincaré ball. They are the generalization of convex cones on a generic manifold $\mathcal{M}$. Here, we provide a brief introduction of entailment cones. We refer to Ganea et al. (2018b) for further details.

In a vector space, a convex cone S at the origin is a set that is closed under non-negative linear combinations:

$$v_1, v_2 \in S \Rightarrow \alpha v_1 + \beta v_2 \in S, \ \forall \alpha, \beta \geq 0.$$

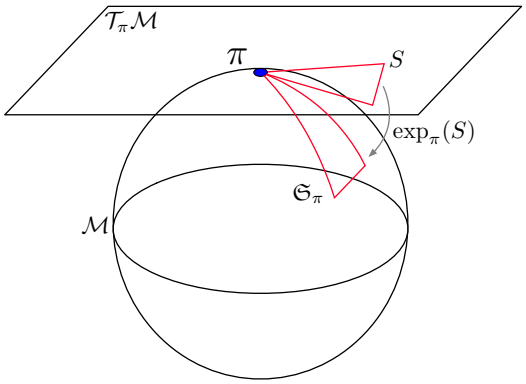
The idea proposed in Ganea et al. (2018b) is to generalize this concept to any manifold $\mathcal{M}$, using the exponential map at any point $x \in \mathcal{M}$, which allows to fold any convex cone onto the manifold $\mathcal{M}$. In particular, we can take any cone in the tangent space $S \subseteq \mathcal{T}_x\mathcal{M}$ at a fixed point x and map it into a set $\mathfrak{S}_x \in \mathcal{M}$, which is called the S-cone at x, via:

$$\mathfrak{S}_x := \exp_x(S).$$

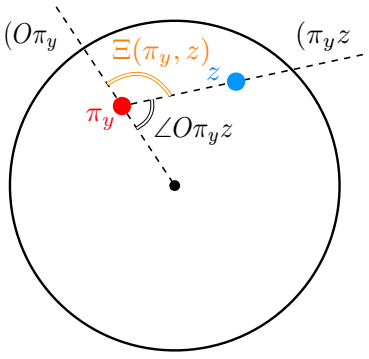
An illustration of this operation is shown in Figure 3a. This projection of the cone onto a manifold is constrained to have some properties, such as axial symmetry, rotation invariance, a continuous cone aperture function and transitivity of nested angular cones. Using these properties it is possible to define a so called **aperture function** $\psi(\cdot)$ in the Poincaré ball in n dimension $\mathbb{B}^n$.

In particular, given $x \in \mathbb{B}^n$, we fix any tangent vector with the same direction as x, e.g. $\bar{x} = \log_x(\frac{1+\|x\|}{2\|x\|}x) \in \mathcal{T}_x\mathbb{B}^n$. We next define the angle $\angle(v, \bar{x})$ for any tangent vector $v \in \mathcal{T}_x\mathbb{B}^n$ as in eq. 4. Then, the axial symmetry property is satisfied if we define the angular cone at x to have a non-negative aperture $2\psi(x) > 0$ as follows:

$$\begin{aligned} S_x^{\psi(x)} &:= \{v \in \mathcal{T}_x\mathbb{B}^n : \angle(v, \bar{x}) \leq \psi(x)\} \\ \mathfrak{S}_x^{\psi(x)} &:= \exp_x(S_x^{\psi(x)}). \end{aligned} \tag{5}$$



(a) Representation of an entailment cone on a generic manifold.



(b) Illustration of the angle Ξ on the Poincaré ball.

Figure 3

To reach a closed formulation of the aperture function $\psi(x)$ (Ganea et al., 2018b) use the transitivity property, developing the closed formula for the aperture function

(or *amplitude*):

$$\begin{aligned} \psi :&\mathbb{B}^n \setminus \mathcal{B}^n(O,\epsilon) \to \left(0, \frac{\pi}{2}\right) \\ &x \to \arcsin\left(K\left(\frac{(1-\|x\|^2)}{\|x\|}\right)\right), \end{aligned} \tag{6}$$

where K is a constant (see (Ganea et al., 2018b)), $\mathcal{B}^n(O,\epsilon)$ represents a ball in the origin O. We note that O is excluded from the domain of ψ since the entailment cones cannot be defined at the origin.

Theorem 1. *(From(Ganea et al., 2018b)) For any $x, y \in \mathbb{B}^n \setminus \mathcal{B}^n(O,\epsilon)$, consider the angle between the half-lines $(xy$ and $(Ox$ as :*

$$\Xi(x,y) := \pi - \angle Oxy.$$

Then, $\Xi(x,y)$ is defined by

$$\arccos\left(\frac{\langle x,y\rangle(1+\|x\|^2) - \|x\|^2(1+\|y\|^2)}{\|x\| \cdot \|x-y\|\sqrt{(1+\|x\|^2\|y\|^2 - 2\langle x,y\rangle)}}\right).$$

Moreover, we have the following equivalent expression of the Poincarè entailment cones satisfying (6)*:*

$$\mathfrak{S}_x^{\psi(x)} = \{y \in \mathbb{B}^n | \Xi(x,y) \leq \psi(x)\}. \tag{7}$$

An illustration of the Ξ angle is displayed in Figure 3b.

4 METHOD

In a traditional PL setting (Yang et al., 2018), distances are employed during training. To summarize the approach, let us consider a dataset $X = \{(x_i, y_i)\}$, with x_i taking values in a sample space $\mathcal{X}$, $y_i \in \mathcal{C} = \{1 \dots C\}$, and $|X| = N$. For each class there is a corresponding prototype in the embedding space $\Pi = \{\pi_j : j \in \mathcal{C}\}$. A backbone network $f(\cdot,\theta) : \mathcal{X} \to \mathbb{R}^d$ is trained to solve:

$$\arg\min_{\theta,\Pi} \mathcal{L}(\theta, \Pi; \gamma), \tag{8}$$

i.e., finding the parameters θ minimizing over the training set the distance based cross-entropy loss Yang et al. (2018):

$$\mathcal{L}(\theta,\Pi;\gamma) = \frac{1}{N} \sum_{(x_i,y_i)\in X} -\log \frac{e^{-\gamma d(f(x_i),\pi_{y_i})}}{\sum_{j\in\mathcal{C}} e^{-\gamma d(f(x_i),\pi_j)}}, \tag{9}$$

where $f(x_i) \in \mathbb{R}^d$, γ is the *temperature* parameter (Yang et al., 2018), and $d(\cdot,\cdot)$ can be any distance measure.

We propose to improve over this approach by employing hyperbolic entailment cones. From the definition of an entailment cone (7), it is clear that the main elements that define $\mathfrak{S}_v^{\psi(v)}$ are the point v (referred to as *vertex*) and a fixed amplitude $\psi(v)$. We leverage these elements in a PL setting, introducing one prototype for each class and assigning the vertex of a cone to it. Specifically, the prototypes $\Pi = \{\pi_j : j \in \mathcal{C}\}$ are positioned to maximize pairwise cosine separation (Mettes et al., 2019). This approach ensures a uniform displacement of the prototypes and consequently leads to a uniform partition of the space. Conceptually, these prototypes are positioned on a hypersphere $\mathcal{S}^d$ (where d is the embedding dimension) and possess unit norm. To project them onto the Poincaré ball, we first scale them to have a norm equal to a hyperparameter r^* referred to as the *prototype ray*. Subsequently, we project them onto the Poincaré ball using the exponential map (3). An illustration of the resulting set of prototypes is provided in Figure 4.

We assign to each prototype a vertex of an entailment cone, with amplitude $\psi(\pi)$. By construction, $\psi(\pi)$ is constant for all entailment cones since it only depends on $\|\pi\|$ and prototypes are at fixed distance r^* from the origin. It is worth noting that the amplitude, as defined in (6), depends on the parameter K, which we fix to 0.1 as done by Ganea et al. (2018b).

In a traditional scenario, a backbone $f(\cdot,\theta)$ is trained to optimize (8), and the loss is usually defined as in Eq. (9). Here, we leverage the representation power of entailment cones and use an adaptation of the loss function proposed by Ganea et al. (2018b). It is worth mentioning that the hypersphere $\mathcal{S}^n$ and the Poincaré ball are related by the conformal property, meaning that angles measured from the origin are the same in both manifolds. Using the entailment cones, we propose to use an angle-based metric that is centered on the prototypes (i.e. *prototype-centered*) rather than

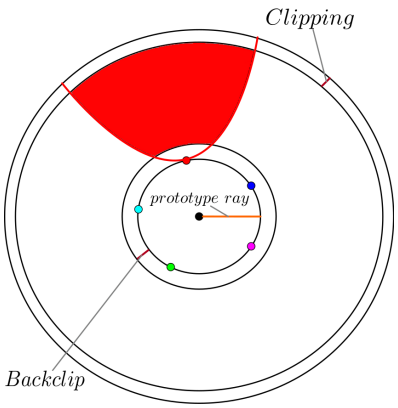


Figure 4: Illustration of the displacement of the prototypes in HPEC, the Backclip mechanism proposed and the Clip mechanism by Guo et al. (2022).

the origin, which differs from the approaches used in traditional hyperspherical methods that use the cosine similarity. In particular, we train the network to minimize:

$$\mathcal{L} = E(\pi_{y_i}, z_i) + \sum_{j \neq i} \max(0, \xi - E(\pi_{y_j}, z_j)), \quad (10)$$

where $z_i = \exp_O(f(x_i, \theta))$, x_i is an example, and π_{y_i} is the prototype of x_i's class and ξ is the margin hyperparameter. The energy $E(\pi, z)$, firstly introduced by Ganea et al. (2018b), is a penalty term that grows with the distance of point z from from the entailment cone $\mathfrak{S}_\pi^{\psi(\pi)}$:

$$E(\pi, z) = \max(0, \Xi(\pi, z) - \psi(\pi)). \quad (11)$$

The expression computes the angle $\Xi(\pi, z)$ needed to align point z with the prototype π using a rotation around π (see Figure 3b); the energy penalizes the network for z values that are farther from ($O\pi$ more than the size of $\psi(\pi)$ of the corresponding cone $\mathfrak{S}_\pi^{\psi(\pi)}$.

It is worth mentioning that the angle Ξ defined in (1) can be used to predict a class by selecting the class whose prototype forms the smallest angle with respect to the example embedding:

$$\hat{y} = \arg\min_{j \in \mathcal{C}} \Xi(\pi_j, z). \quad (12)$$

4.1 Backclip

To represent the images into the embedding space, we use a backbone network $f(\cdot, \theta) : \mathcal{X} \to \mathbb{R}^d$(e.g. a ResNet18). Then we project them on the Poincaré ball, through the exponential map (3). However, before projecting the embeddings, we apply a novel *Backclip* mechanism. Given a euclidean embedding $\hat{x} = f(x, \theta)$, where $x \in \mathcal{X}$, and its representation on the Poincaré ball $z = \exp_O(\hat{x})$, we want to avoid that $\|z\| < r^*$, i.e. that the embedding belongs to the inner circle in Figure 4. To do so, we introduce the *shrinking factor* ζ_x:

$$\zeta_x = \|z\| = \frac{1}{\sqrt{\kappa}} \tanh(\sqrt{\kappa}\frac{\|\hat{x}\|}{2}), \quad (13)$$

where κ is the curvature of the Poincaré ball. In addition, we introduce a *backclip margin* ϵ, a hyperparameter to further increase the distance between the representations and the prototypes. We define the *Backclip* of $\hat{x}$ to r as:

$$\text{Backclip}(\hat{x}, r) = \max\left\{1, \frac{r}{\|\hat{x}\|}\right\} \hat{x} \quad (14)$$

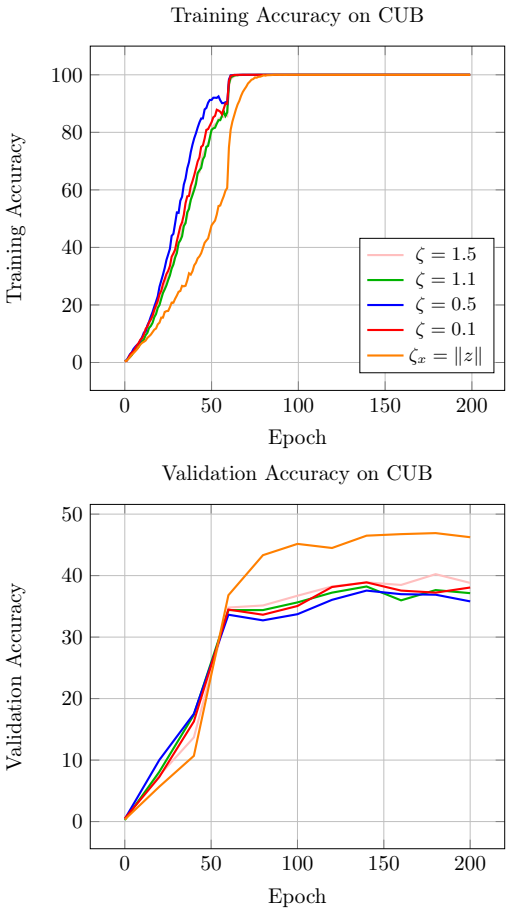


Figure 5: Effect of the shrinking factor of the Backclip mechanism, both during training and validation. In this case, the validation set was set to be the 20% of the training set. As we can notice, putting a fixed shrinking factor does not bring performance improvements. n the other hand, the ζ_x shows to lead the performances, due to its capacity of clipping the embedding based on their specific norm.

and propose to use

$$r = \hat{r} = r^* + \epsilon + \zeta_x. \quad (15)$$

It is worth noting that constant shrinking factors empirically show worse performances than the ones based on the norm of x. Figure 5 shows the accuracy of systems using constant shrinking factors of 0.1, 0.5, 1.1 and 1.5, as well as the proposed shrinking factor ζ_x. The plots clearly show a tendency to overfitting when constant clipping values are used. At the same time, Table 2 shows the impact of the Backclip on several datasets and embedding dimensions. The embedding dimensions tested were $\{8, 32, 64, 100\}$ for CIFAR-100 and Aircraft, $\{16, 64, 128, 196\}$ for Cars and $\{16, 64, 128, 200\}$ for CUB. The r parameter refers to Equation 14. Using $r = 0$ is equivalent to not using

Table 2: Percentage of test accuracy on 4 datasets and 4 embedding dimensions for HPEC with and without Backclip mechanism. In bold we indicate the results that are significantly different from the other.

		Embedding Dimension			
	r	1^{st}	2^{nd}	3^{rd}	4^{th}
CIFAR-100	0	72.48	75.78	76.01	76.31
	$\hat{r}$	72.42	75.31	75.73	76.04
Aircraft	0	61.86	72.87	73.95	77.85
	$\hat{r}$	**64.57**	**75.38**	**76.88**	77.62
Cars	0	34.14	62.40	72.00	70.68
	$\hat{r}$	**40.07**	**73.50**	**77.18**	**78.75**
CUB	0	24.35	47.58	52.88	54.19
	$\hat{r}$	**30.59**	**53.73**	**55.76**	**57.40**

the Backclip, while $r = \hat{r}$ indicates our choice. This Table clearly depicts the crucial role of Backclip in boosting the performance, especially for Aircraft, Cars and CUB. For CIFAR-100 its impact is limited. The intuition behind the Backclip derives from the original employment of entailment cones of representing tree-like structures, and it wants to constraint the partial ordering between embeddings and prototypes. While this is not a mathematical constraint from the definition of the entailment cones, it proved to be very effective.

On the other hand, as explained in Guo et al. (2022) and confirmed empirically in Hamzaoui et al. (2024), embeddings naturally tend towards the border of the Poincaré ball, resulting in vanishing gradient issues during the optimization of the backbone network parameters. To mitigate this problem, we clip the embeddings as proposed by Guo et al. (2022), to prevent them from approaching the Poincaré ball border too closely.

Figure 2 shows the HPEC framework, and Figure 1 shows a toy example on MNIST displaying the results of training a simple convolutional network for a few epochs with embedding dimension 2.

5 EXPERIMENTS

We compare our method with several state-of-the art PL methods for image classification and with a baseline non-PL method. Specifically, we use as our baseline the output of the backbone network optimized with a cross-entropy loss (*XE* in the following), and with the following PL methods: *ECL* (Landrieu and Garnot, 2021) a method working in the Euclidean geometry, which in addition to learning examples embeddings, optimizes prototype positions to best work with the learnt representations; *HPS* (Mettes et al., 2019) a method working on a hyperspherical manifold, using fixed prototypes positioned uniformly on the border of the hypersphere; *CHPS* (Fonio et al., 2023) a method working on a hyperspherical manifold that leverages a contrastive loss function; *NF* (Kornblith et al., 2021) using cosine softmax and temperature equal to 0.08 is a method that uses cosine similarity while updating the prototypes; *HBL* is a method that updates the prototypes that lie on the Poincaré ball and uses hyperbolic distances inside a Cross-entropy loss function; *BSM* (Ghadimi Atigh et al., 2021) a method working on a hyperbolic manifold, i.e., the Poincaré ball, that position the prototypes on the border of Poincaré ball and uses the Busemann distance during training;

We compare the performances of the algorithms on four public datasets: *CIFAR-100* (Krizhevsky, 2009) 100 classes, 50000/10000 examples (train/test); *CUB* (Wah et al., 2011) 200 classes, 5994/5794 examples (train/test); *Aircraft* (Maji et al., 2013) 100 classes, 6667/3333 examples (train/test); *Cars* (Krause et al., 2013) 196 classes, 8144/8041 examples (train/test). The datasets were chosen because they provide a fine-grained hierarchy over the classes (Aircraft and Cars) or because they show low δ-hyperbolicity (CIFAR-100, CUB) (Khrulkov et al., 2020). Further experimental details can be found in the Appendix.

6 RESULTS AND DISCUSSIONS

Table 3 reports the results of the 6 tested methods. The results show that HPEC reaches leading results across the board. On the Aircraft dataset, HPEC is a little bit worse than ECL, but not in statistical significant way. In all other cases, HPEC bests the other methods with a statistically significant margin.

Among the datasets in which we show leading results, it is worth noting that on CUB (200 classes) the improvement is very significant. As detailed in Figure 5, setting the Backclip mechanism to $\zeta = \zeta_x$ is key to this result. Notably, also on Cars (196 classes) HPEC shows a significant improvement. An interesting facet of the shown results is that the XE method, which is the sole non-PL approach, is very competitive. As discussed in 1, PL is generally not very competitive in image classification, where established alternative methods developed over the past decades dominate. Nonetheless, HPEC manages to outperform XE in almost all cases, with a conspicuous widening gap as we proceed from left to right in the table, i.e., as the number of classes increases.

6.1 Comparison with Euclidean methods

The results displayed in Table 3 show that HPEC is capable of outperforming XE in image classification

Table 3: Percentage of test accuracy on 4 datasets for our proposed method and the competing approaches. The results displayed are the average over 5 runs with their standard deviation. In bold it is highlighted the best result, the second best result is underlined. If two leading results are significantly similar according to a paired t-test at 95% confidence level they are both shown in bold or underlined.

Method	**Dataset**			
	CIFAR-100	Aircraft	Cars	CUB
XE	<u>75.63 ± 0.26</u>	76.65 ± 0.36	<u>73.21 ± 0.53</u>	48.17 ± 0.59
ECL	74.28 ± 0.24	78.06 ± 0.37	70.69 ± 1.22	<u>50.16 ± 0.73</u>
HPS	69.17 ± 0.25	73.06 ± 0.63	60.94 ± 0.96	46.76 ± 0.46
CHPS	74.16 ± 0.34	75.87 ± 0.80	66.63 ± 0.64	46.37 ± 0.33
NF	72.11 ± 0.55	<u>78.45 ± 0.25</u>	72.61 ± 0.48	44.68 ± 0.28
HBL	75.16 ± 0.21	**78.77** ± 0.31	71.21 ± 0.83	46.97 ± 0.72
BSM	70.32 ± 1.15	63.89 ± 0.81	46.48 ± 2.77	44.10 ± 1.58
HPEC	**76.04** ± 0.22	77.62 ± 0.55	**78.75** ± 0.44	**57.40** ± 0.21

with a PL based approach. For what it concerns ECL, which is a prototypical method that relies on Euclidean distances to optimize the parameters of the neural network, we notice that it shows comparable performances with respect to XE, but it is unable to match the performances of HPEC. The differences observed may be due to two factors: the use of the Poincaré ball as embedding space, which aligns the data's latent hierarchical structure with the geometry of the embedding space, and the use of angles instead of distances. While there is a Euclidean version of Entailment cones (Ganea et al., 2018b), they were originally defined in the Poincaré ball and later adapted to Euclidean space. Currently, there is no adaptation of Euclidean entailment cones for a PL setting. Future work will aim to develop a Euclidean version of the entailment cones to fairly compare the use of distances versus angles across different geometries.

While standard hyperbolic methods excel in tasks like few-shot learning and low-dimensional spaces, this study demonstrates that a hyperbolic method can also surpass a Euclidean method in high-dimensional tasks such as image classification. However, the results shown in Table 4 and Table 5 indicate that angle-based methods that use fixed prototypes perform worse than ECL in low dimensions. It is worth mentioning that in the experiments of Table 3 the embedding dimension is equal to the number of classes. This complicates determining whether the technique's advantage is due to the number of classes or its effectiveness in large embedding spaces. This issue will be explored further in Section 6.3.

6.2 Comparison with non-Euclidean methods

Hyperspherical methods HPS, CHPS and NF involve normalizing embeddings to lie on the boundary of the n-dimensional hypersphere $\mathcal{S}^{n+1} = \{x \in \mathbb{R}^n | \|x\| = 1\}$ and using fixed prototypes. Both methods utilize cosine similarity to optimize neural network parameters. HPS maximizes the similarity between an embedding and its corresponding prototype, whereas CHPS also minimizes the similarity between the embedding and other prototypes. This comprehensive consideration of all prototypes allows CHPS to outperform HPS on all datasets.

As evidenced by the results in Table 3 for high dimensions and Table 5 and Table 4 for low dimensions, HPEC outperforms other angle-based methods such as HPS and CHPS. The use of entailment cones made this possible through leveraging a *prototype-centered* similarity metric, instead of the cosine similarity. This choice, coupled with a loss function that considers all prototypes (as CHPS does), allows HPEC to outperform the hyperspherical methods both in low and high dimensional spaces.

The comparison with NF, which employs dynamic prototypes and cosine similarity, is particularly insightful. The significant performance gap between NF and its counterpart with fixed prototypes highlights the crucial role of dynamic prototypes. However, HPEC surpasses NF by utilizing a hyperbolic metric, especially in datasets with a large number of classes. This suggests that leveraging an angle-based metric within the appropriate geometric framework (HPEC) is more effective than relying solely on a dynamic prototype-based angle metric (NF).

Hyperbolic Methods To the best of our knowledge, the only hyperbolic method addressing image classification is presented by Ghadimi Atigh et al. (2021), where the authors demonstrate superior performance relative to HPS. This approach primarily utilizes the Busemann distance between embeddings and prototypes on the

Table 4: Percentage of test accuracy on Aircraft and CIFAR-100 for our proposed method and the competing approaches. These results are the average over 3 runs. The best results among all the methods are in bold, while the second best method is underlined.

		Embedding dimension		
		8	**32**	**64**
CIFAR-100	ECL	**74.09** ± 0.43	<u>74.31</u> ± 0.19	<u>74.48</u> ± 0.19
	HPS	54.37 ± 1.32	67.38 ± 0.67	68.96 ± 0.14
	CHPS	70.18 ± 0.48	73.71 ± 0.15	74.16 ± 0.34
	NF	69.60 ± 0.12	72.48 ± 0.45	72.21 ± 0.61
	HBL	72.29 ± 0.18	74.73 ± 0.11	74.95 ± 0.20
	BSM	70.71 ± 0.30	72.03 ± 0.28	72.23 ± 0.36
	HPEC	<u>72.42</u> ± 0.12	**75.31** ± 0.31	**75.73** ± 0.26
Aircraft	ECL	75.27 ± 0.74	77.81 ± 0.34	78.15 ± 0.45
	HPS	29.76 ± 5.94	66.88 ± 2.08	69.79 ± 0.53
	CHPS	59.95 ± 0.47	72.75 ± 0.74	74.56 ± 0.52
	NF	**78.62** ± 0.12	**78.47** ± 1.00	<u>78.45</u> ± 0.25
	HBL	<u>78.13</u> ± 0.44	<u>77.65</u> ± 0.41	**78.77** ± 0.31
	BSM	57.18 ± 2.55	70.75 ± 0.41	61.67 ± 0.34
	HPEC	64.57 ± 0.45	75.38 ± 0.40	76.88 ± 0.43

Table 5: Percentage of test accuracy on CUB and Cars for our proposed method and the competing approaches. These results are the average over 3 runs. The best results among all the methods are in bold, while the second best method is underlined.

		Embedding dimension		
		16	**64**	**128**
CUB	ECL	<u>36.03</u> ± 0.51	43.64 ± 1.49	<u>47.84</u> ± 0.76
	HPS	14.33 ± 0.20	39.53 ± 0.71	43.91 ± 0.70
	CHPS	27.00 ± 0.97	40.19 ± 0.87	45.19 ± 0.69
	NF	**44.87** ± 0.94	<u>44.40 ± 0.31</u>	44.74 ± 1.41
	HBL	39.96 ± 0.84	45.96 ± 1.08	47.16 ± 0.48
	BSM	28.99 ± 0.87	27.28 ± 1.39	45.34 ± 1.00
	HPEC	30.59 ± 0.34	**53.73** ± 0.16	**55.76** ± 0.11
Cars	ECL	60.66 ± 1.58	65.94 ± 0.62	69.13 ± 0.10
	HPS	7.18 ± 2.09	42.82 ± 2.38	57.09 ± 1.45
	CHPS	25.23 ± 8.14	55.77 ± 1.27	63.58 ± 0.49
	NF	**72.12** ± 2.0	<u>72.37</u> ± 0.08	<u>72.20</u> ± 0.66
	HBL	69.15 ± 0.74	70.54 ± 0.59	71.21 ± 0.83
	BSM	32.68 ± 0.73	38.63 ± 1.67	58.28 ± 2.95
	HPEC	<u>40.07</u> ± 1.25	**73.50** ± 0.91	**77.18** ± 0.78

boundary. Table 3, Table 5 and Table 4 show that BSM exibits limited performances compared to all methods except HPS. Furthermore, its convergence necessitates an expensive experimental setting and the correct tuning of a highly sensitive parameter (the *penalty*) to prevent over-confidence. The main idea of HPEC, which is confirmed by the provided empirical evidence, is that using angle-based metric is a key aspect in improving performances with respect to specifically-tailored distances, as done by Ghadimi Atigh et al. (2021).

Furthermore, we have chosen HBL as baseline, a naive approach employing hyperbolic distances in a cross-entropy with dynamic prototypes. Despite showing competitive performances, HPEC outperforms it, providing an interesting winning comparison with respect to HBL. This indicates that it is crucial to identify the specific metric properties that enhance the model's representational capacity. In this case, angular information is more beneficial than distance information.

6.3 Number of classes and Embedding dimension

The relationship between performance and dimensionality was explored by varying the number of embedding dimensions, and the results are displayed in Table 5 and 4. The results indicate that using low dimensional representations negatively impact angle-based methods with fixed prototypes such as HPEC, HPS and CHPS, while solutions with dynamic prototypes (such as ECL, NF and HBL) do not change much as the number of dimensions grows. HPEC usually starts at a lower performance level, but quickly takes the lead (with the only exception of the Aircraft dataset) as the number of dimensions grows. Our findings suggest that HPEC benefits significantly from relatively high embedding dimensions, and our angle-based metric also outperforms the Busemann distance defined by Ghadimi Atigh et al. (2021).

Overall, our experiments demonstrate that the performance of angle-based methods with fixed prototypes such as HPEC and CHPS and HPS is impacted by the embedding dimensions. However, HPEC shows remarkable improvements in this regards, lowering the gap with dynamic prototypes methods, that are still leading in low dimensional spaces.

7 CONCLUSIONS

In this paper we introduced a PL framework that leverages the representation power of entailment cones and introduced the *Backclip* mechanism. Our experiments show that hyperbolic spaces benefit from considering angles, reaching leading performances in a traditional task like image classification. In the future, we will investigate approaches where the hierarchy is explicitly exploited, for instance by setting the prototype positions based on a hierarchy provided by the user.

Acknowledgments

This work has been partly supported by the Spoke "FutureHPC & BigData" of the ICSC - Centro Nazionale di Ricerca in "High Performance Computing, Big Data and Quantum Computing", funded by European Union - NextGenerationEU. We also thank the reviewers for their insightful feedback and constructive discussion.

References

Marco Aldinucci, Sergio Rabellino, Marco Pironti, Filippo Spiga, Paolo Viviani, Maurizio Drocco, Marco Guerzoni, Guido Boella, Marco Mellia, Paolo Margara, et al. Hpc4ai: an ai-on-demand federated platform endeavour. In *Proceedings of the 15th ACM International Conference on Computing Frontiers*, pages 279–286, 2018.

Ahmad Bdeir, Kristian Schwethelm, and Niels Landwehr. Fully hyperbolic convolutional neural networks for computer vision. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=ekz1hN5QNh`.

Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

Hyunghoon Cho, Benjamin DeMeo, Jian Peng, and Bonnie Berger. Large-margin classification in hyperbolic space. In *The 22nd international conference on artificial intelligence and statistics*, pages 1832–1840. PMLR, 2019.

Karan Desai, Maximilian Nickel, Tanmay Rajpurohit, Justin Johnson, and Shanmukha Ramakrishna Vedantam. Hyperbolic image-text representations. In *International Conference on Machine Learning*, pages 7694–7731. PMLR, 2023.

Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. Hyperbolic vision transformers: Combining improvements in metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7409–7419, 2022.

Samuele Fonio, Lorenzo Paletto, Mattia Cerrato, Dino Ienco, Roberto Esposito, et al. Hierarchical priors for hyperspherical prototypical networks. In *ESANN 2023-Proceedings*, pages 459–464. ESANN, 2023.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *Advances in neural information processing systems*, 31, 2018a.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*, pages 1646–1655. PMLR, 2018b.

Mina Ghadimi Atigh, Martin Keller-Ressel, and Pascal Mettes. Hyperbolic busemann learning with ideal prototypes. *Advances in Neural Information Processing Systems*, 34:103–115, 2021.

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*, 2018.

Yunhui Guo, Xudong Wang, Yubei Chen, and Stella X Yu. Clipped hyperbolic classifiers are super-hyperbolic classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2022.

Manal Hamzaoui, Laetitia Chapel, Minh-Tan Pham, and Sébastien Lefèvre. Hyperbolic prototypical network for few shot remote sensing scene classification. *Pattern Recognition Letters*, 177:151–156, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6418–6428, 2020.

Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian optimization in pytorch, 2020.

Simon Kornblith, Ting Chen, Honglak Lee, and Mohammad Norouzi. Why do better loss functions lead to less transferable features? *Advances in Neural Information Processing Systems*, 34:28648–28662, 2021.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*, 2009.

Loic Landrieu and Vivien Sainte Fare Garnot. Leveraging class hierarchies with metric-guided prototype learning. In *British Machine Vision Conference (BMVC)*, 2021.

Keegan Lensink, Bas Peters, and Eldad Haber. Fully hyperbolic convolutional neural networks. *Research in the Mathematical Sciences*, 9(4):60, 2022.

Xinting Liao, Weiming Liu, Chaochao Chen, Pengyang Zhou, Huabin Zhu, Yanchao Tan, Jun Wang, and Yue Qi. Hyperfed: hyperbolic prototypes exploration with consistent aggregation for non-iid data in federated learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pages 3957–3965, 2023.

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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9273–9281, 2020.

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

S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, 2013.

Pascal Mettes, Elise Van der Pol, and Cees Snoek. Hyperspherical prototype networks. *Advances in neural information processing systems*, 32, 2019.

Pascal Mettes, Mina Ghadimi Atigh, Martin Keller-Ressel, Jeffrey Gu, and Serena Yeung. Hyperbolic deep learning in computer vision: A survey. *International Journal of Computer Vision*, pages 1–25, 2024.

Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International conference on machine learning*, pages 3779–3788. PMLR, 2018.

Zexuan Qiu, Jiahong Liu, Yankai Chen, and Irwin King. Hihpq: Hierarchical hyperbolic product quantization for unsupervised image retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4614–4622, 2024.

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

Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning*, pages 4460–4469. PMLR, 2018.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Panu Somervuo and Teuvo Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10:151–159, 1999.

Robert Tibshirani, Trevor Hastie, Balasubramanian Narasimhan, and Gilbert Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99 (10):6567–6572, 2002.

Max van Spengler, Erwin Berkhout, and Pascal Mettes. Poincaré resnet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5419–5428, 2023.

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.

Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1041–1049, 2017.

Hong-Ming Yang, Xu-Yao Zhang, Fei Yin, and Cheng-Lin Liu. Robust classification with convolutional prototype learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3474–3482, 2018.

Yun Yue, Fangzhou Lin, Guanyi Mou, and Ziming Zhang. Understanding hyperbolic metric learning through hard negative sampling. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1891–1903, 2024.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] The background section 3 includes all the necessary mathematical tools to fully understand the paper
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] In Section 4.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] In Appendix 7 the code is available through an anonimized repository.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] In Appendix 7.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] The data splits are in the code, the most important hyperparameter was discussed in 5.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] The measure used was the accuracy on the test set, as reported in Section 5.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] Available in the Appendix 7.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

APPENDIX

Experimental details

Each method is equipped with a ResNet18 (He et al., 2016) backbone network. The competitors were riproduced using the hyperparameters setting found in the original papers. For HPEC the network was trained using RSGD (Bonnabel, 2013) with a learning rate of 0.1, momentum of 0.9, and weight decay of 0.0005, for 200 epochs. For setting the hyperparameters of HPEC, we conducted a light hyperparameter selection session by testing a few settings on 20% of the CUB training set. We selected the parameters $r^* = 0.3$, $\epsilon = 0.01$, $\xi = 1$, and $K = 0.1$, and kept them fixed for all experiments. It is worth mentioning that different parameters were tested, but their choice did not impact significantly the performances. The most important impact is from the Backclip mechanism and it is shown in figure 5. The experiments were run on HPC4AI (Aldinucci et al., 2018) a cluster with 4 ARM machine, which consists of Ampere Altra Q80-30 CPU (80-core Arm Neoverse N1), 512GB of memory, 2 x NVIDIA A100 GPU (40GB vram). We used geoopt (Kochurov et al., 2020) to implement the hyperbolic operations.

Computational Cost details

The experiments varies across the datasets for what regards computational need, due to the different sizes of the images and the diverse number of samples. Specifically, CIFAR-100 have 32×32 pixels per image, while Aircraft, Cars and CUB have 240×240 pixels per image.

For CIFAR-100, we used a modified ResNet18. The first convolutional layer, due to the small size of the images and following Ghadimi Atigh et al. (2021), has been changed to use a 3×3 kernel instead of the default 7×7 kernel. For the other datasets, we used the ResNet18 as implemented in the pytorch library.

The transformations used were:

- **Cars**: Resize((224,224)), RandomCrop(224, padding=4), RandomHorizontalFlip(), Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225));
- **Aircraft**: Resize((224,224)), RandomCrop(224, padding=4), RandomHorizontalFlip(), Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225));
- **CUB**: Resize((224,224)), RandomCrop(224, padding=4), RandomHorizontalFlip(), Normalize((0.485, 0.456, 0.406), (0.229, 0.224, 0.225));

- **CIFAR-100**: RandomCrop(32, 4),RandomHorizontalFlip(), RandomRotation(15), Normalize((0.507, 0.487, 0.441), std=(0.267, 0.256, 0.276)).

The code is available here: `https://github.com/samuelefonio/HPEC`.

The computational cost for any method is not significantly different from a standard training of a ResNet18 on the respective datasets. The most computationally demanding setting is the one of Ghadimi Atigh et al. (2021), due to the slow convergence.

Competitors

The competitors were reproduced with the hyperparameters indicated in their original papers. In order to make a fair comparison, the only things that were changed were the backbone and the output dimension. For the HBL method Ghadimi Atigh et al. (2021) we used the official repository of the original paper. After doing a light hyperparameter optimization, we kept the penalty equal to the ones in the original paper, and used 0.05 and 0.01 respectively for Aircraft and Cars. The other hyperparameters were set as stated in the original paper.