Hyperbolic Busemann Learning with Ideal Prototypes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Hyperbolic space has become a popular choice of manifold for representation learning of arbitrary data, from tree-like structures and text to graphs. Building on the success of deep learning with prototypes in Euclidean and hyperspherical spaces, a few recent works have proposed hyperbolic prototypes for classification. Such approaches enable effective learning in low-dimensional output spaces and can exploit hierarchical relations amongst classes, but require privileged information about class labels to position the hyperbolic prototypes. In this work, we propose Hyperbolic Busemann Learning. The main idea behind our approach is to position prototypes on the ideal boundary of the Poincaré ball, which does not require prior label knowledge. To be able to compute proximities to ideal prototypes, we introduce the penalised Busemann loss. We provide theory supporting the use of ideal prototypes and the proposed loss by proving its equivalence to logistic regression in the one-dimensional case. Empirically, we show that our approach provides a natural interpretation of classification confidence, while outperforming recent hyperspherical and hyperbolic prototype approaches.

1 Introduction

Classification by prototypes has a long tradition in machine learning. Foundational solutions such as the Nearest Mean Classifier [43] represent classes as the mean prototypes in a fixed feature space. Following approaches that learning a metric space for class prototypes [27], deep learning with prototypes as points in network output spaces has gained traction [13, 16, 25, 31, 34, 39, 45, 47]. In line with foundational approaches, the prototypes are positioned by computing the mean over all training examples for each class. Deep learning by mean prototypes has shown to be effective for tasks such as few-shot learning [13, 31, 34, 39] and zero-shot recognition [39, 45, 47].

To avoid the need to re-learn prototypes or to enable the use of prior label knowledge, several recent works have proposed deep networks with prototypes in non-Euclidean output spaces. Hyperspherical prototype approaches initialize prototypes on the sphere or its higher-dimensional generalization, based on pair-wise angular separation [28, 37] or with the help of word embeddings [4, 41] using a cosine similarity loss between example outputs and class prototypes. Hyperspherical prototypes alleviate the need to re-position prototypes continuously and allow to optionally include prior semantic knowledge about the classes. Recently, a few works have extended prototype-based learning to the hyperbolic domain, where prototypes are obtained by embedding label hierarchies [22, 23]. While hyperbolic prototype approaches provide more hierarchically coherent classification results, prior hierarchical knowledge is required to embed prototypes. This paper strives to combine the best of both non-Euclidean worlds, namely efficient low-dimensional embeddings from hyperbolic prototypes and the knowledge-free positioning from hyperspherical prototypes.

We make three contributions in this work. First, we introduce a hyperbolic prototype network with class prototypes given as points on the ideal boundary of the Poincaré ball model of hyperbolic geometry. Second, we propose the penalized Busemann loss, which enables us to compute proximities

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

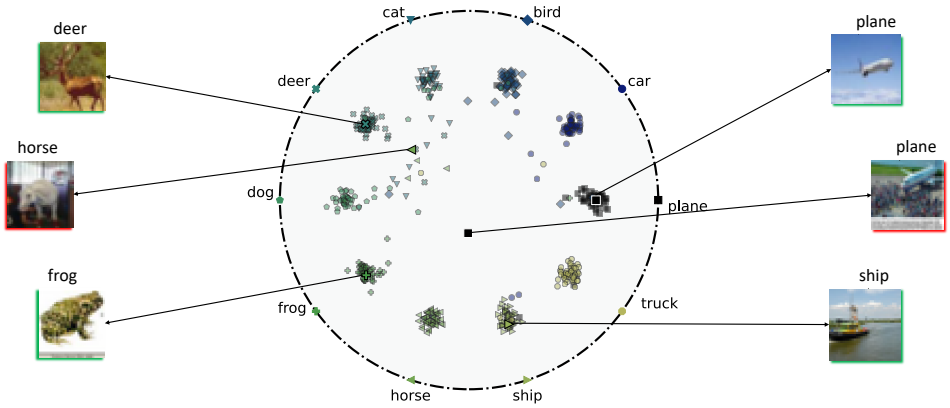


Figure 1: **Visualization of the output space for Hyperbolic Busemann Learning** in two dimensions on CIFAR-10. The penalized Busemann loss optimizes images to be close to their corresponding ideal prototypes while forcing all examples towards the origin to avoid over-confidence, *e.g.,* for the *deer*, top *plane*, and *ship* examples. Ambiguous cases, such as the *horse* and bottom *plane* examples are projected closer to the origin, providing a natural interpretation of classification confidence.

between example outputs in hyperbolic space and prototypes at the ideal boundary, an impossible task for existing distance metrics, which put the ideal boundary at infinite distance from all other points in hyperbolic space. Third, we provide a theoretical link between our hyperbolic prototype approach and logistic regression. We show that our choices for output manifold, for prototypes on the ideal boundary, and for the proposed loss provide a direct generalization of the logistic regression model. Experiments on three datasets show that our approach outperforms both hyperspherical and hyperbolic prototype approaches. Moreover, our hyperbolic output space provides a natural interpretation of closeness to the ideal boundary as classification confidence, see Figure 1. The code is available in the supplementary files.

2 Related work

As an early adaptation of prototypes in deep networks, the center loss [44] served as an additional loss for softmax cross-entropy optimization, where all outputs of a class were pulled to one point to help minimize intra-class variation. Closer to the original Nearest Mean Classifier objective, Prototypical Networks introduce a prototype-only approach where each class is represented as the mean output of its corresponding training examples [39]. Learning with prototypes from class means has shown to be effective for few-shot learning [13, 31, 34, 39], zero-shot recognition [39, 45, 47], and domain adaptation [32]. Defining prototypes as class means is feasible in few-shot learning by sampling all samples of a class in a mini-batch. For general supervised learning, DeepNCM [16] and DeepNNO [25] propose alternating optimization schemes, where the prototypes are fixed to update the network backbone and vice versa. To avoid the need for constant prototype updating, several works have proposed to use the hypersphere as output space instead of the Euclidean space. On the hypersphere, prototypes can be positioned *a priori* based on angular separation [28, 37], using an orthonormal basis [4], or using prior knowledge from word embeddings [4, 28, 41]. Similar to hyperspherical prototype approaches, we seek to position prototypes at unit distance from the origin. In contrast, we operate on a hyperbolic manifold, which provides a theoretical link to logistic regression and obtains better empirical results than a hypershperical manifold and loss.

In this work, we strive for a prototype-based network on hyperbolic manifolds, which have become a popular choice for embedding data and deep network layers. Hyperbolic embeddings have shown to represent tree-like structures [36, 35], text [1, 42, 51], cells [19], networks [17], multi-relational knowledge graphs [3], and general taxonomies [15, 20, 29, 30, 46] with minimal distortion. In deep networks, hyperbolic alternatives have been proposed for fully-connected layers [14, 38], convolutional layers [38], recurrent layers [14], classification layers [12, 14, 38], and optimizers [5, 6, 48]. Hyperbolic generalizations have also been introduced for graph networks [2, 11, 21, 24, 49, 50] and generative models [7, 26]. Where hyperbolic-based deep learning approaches commonly operate

within the Poincaré model represented by the open unit ball, we propose to position class prototypes on its ideal boundary – the shell of the unit ball – allowing us to benefit from the representational power of hyperbolic geometry without requiring prior knowledge to operate.

A few works have recently proposed prototype-based approaches with hyperbolic output spaces. Hyperbolic Image Embeddings by Khrulkov *et al.* [18] include a prototype-based setting as a direct hyperbolic generalization of Prototypical Networks [39], allowing for few-shot learning through hyperbolic averaging. Beyond few-shot learning, Long *et al.* [23] introduce a hyperbolic action network, where action prototypes are projected to a hyperbole based on prior knowledge about hierarchical relations. Videos are in turn projected to the same space and compared to the prototypes using the hyperbolic distance, enabling hierarchically consistent action recognition and zero-shot recognition. Similarly, Liu *et al.* [22] perform zero-shot recognition using hyperbolic prototypes positioned based on known hierarchical relations. In this work, we also employ prototypes in hyperbolic space, but do not require prior knowledge about hierarchies.

3 Learning with hyperbolic ideal prototypes

3.1 Problem formulation

We are given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with N examples, where $\mathbf{x}_i \in \mathbb{R}^I$ denotes an input with dimensionality I and $y_i \in \{1, .., C\}$ denotes a label from a set of size C. We seek to learn a projection of inputs to a hyperbolic manifold in which can compute proximities to class prototypes. We do so through a transformation $\mathcal{F}$ in Euclidean space, followed by an exponential map from the tangent space $\mathcal{T}_x\mathcal{M}$ to a hyperbolic manifold $\mathcal{M}$:

$$\mathbf{z} = \exp_{\mathbf{v}}(\mathcal{F}(\mathbf{x}; \theta)), \tag{1}$$

where θ denotes the parameters to learn the transformation. The transformation can be any function, e.g., a deep network or a linear layer akin to logistic regression. The hyperbolic manifold in which $\mathbf{z}$ operates is described throughout this work by the Poincaré ball model $(\mathbb{B}_d, g_{\mathbf{z}}^{\mathbb{B}})$, with open ball $\mathbb{B}_d = \{\mathbf{z} \in \mathbb{R}^d \mid ||\mathbf{z}||^2 < 1\}$ and the Riemannian metric tensor $g_{\mathbf{z}}^{\mathbb{B}} = 4(1 - ||\mathbf{z}||^2)^{-2}\mathbf{I}_d$ [10, 33]. Setting its center to $\mathbf{v} = 0$ the exponential map in the Poincaré model is given by:

$$\exp_0(\mathbf{x}) = \tanh(||\mathbf{x}||/2)\frac{\mathbf{x}}{||\mathbf{x}||}. \tag{2}$$

Furthermore, the geodesic distance between two points $z_1, z_2 \in \mathbb{B}_d$ is given by:

$$d_{\mathbb{B}}(\mathbf{z}_1, \mathbf{z}_2) = \operatorname{arcosh}\left(1 + 2\frac{||\mathbf{z}_1 - \mathbf{z}_2||^2}{(1 - ||\mathbf{z}_1||^2)(1 - ||\mathbf{z}_2||^2)}\right). \tag{3}$$

To train our model, we compare the projected inputs to their class labels, represented by the prototypes $P = \{\mathbf{p}_1, .., \mathbf{p}_C\}$. That is, the overall training loss is given as the aggregation of distance-based individual losses ℓ between projected inputs and the prototype of their respective class labels:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \ell(\exp_0(\mathcal{F}(\mathbf{x}; \theta)), \mathbf{p}_{y_i}). \tag{4}$$

3.2 Ideal prototypes and the penalized Busemann loss

The central idea of our approach is to position class prototypes at ideal points, which represent points at infinity in hyperbolic geometry. In the Poincaré model, the ideal points form the boundary of the ball:

$$\mathbb{I}_d = \{\mathbf{z} \in \mathbb{R}^d : z_1^2 + \cdots + z_d^2 = 1\}. \tag{5}$$

Thus, the set of ideal points of hyperbolic space $\mathbb{B}_d$ is homeomorphic to the hypersphere $\mathbb{S}_d$. As a consequence, any of the methods used for prototype embedding into $\mathbb{S}_d$ can be used to embed prototypes into $\mathbb{I}_d$. In particular, for $d = 2$, prototypes can be placed uniformly on the unit sphere $\mathbb{S}_2$, while class-agnostic prototype embedding based on separation can be used for $d \geq 3$ [28].

A key issue that needs to be solved, is that ideal points are at infinite geodesic distance from all other points in $\mathbb{B}_d$ and hence the geodesic distance can not be used directly for prototype-based learning.

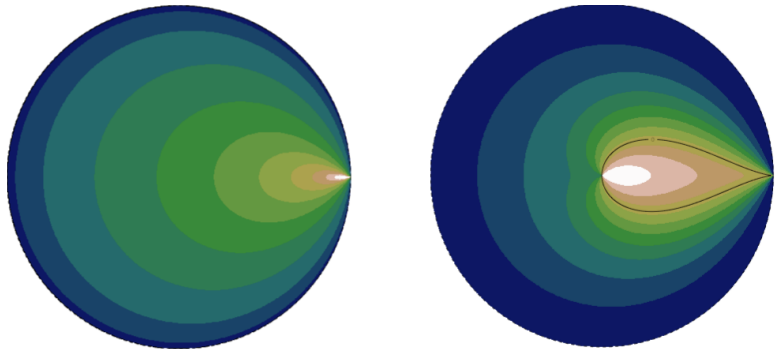


Figure 2: **Visualization of the penalized Busemann loss.** The left plot shows a color map of the penalized Busemann loss function, see (8), in dimension two for a single ideal prototype located at $\mathbf{p} = (1, 0)$; contour lines are log-spaced. The right plot shows its gradient with respect to the radial coordinate. During loss minimization, points in the drop-shaped region adjacent to the prototype will move outwards, while points outside of the region will move inwards.

Therefore, we propose the penalized Busemann loss for hyperbolic learning with ideal prototypes. The Busemann function, originally introduced in [9] (see also Def. II.8.17 in [8]), can be considered a distance to infinity and may be defined in any metric space. Let $\mathbf{p}$ be an ideal point and $\gamma_{\mathbf{p}}$ a geodesic ray, parametrized by arc length, tending to $\mathbf{p}$. Then the Busemann function with respect to $\mathbf{p}$ is defined for $\mathbf{z} \in \mathbb{B}_d$ as:

$$b_{\mathbf{p}}(\mathbf{z}) = \lim_{t \to \infty} (d_{\mathbb{B}}(\gamma_{\mathbf{p}}(t), \mathbf{z}) - t). \tag{6}$$

In the Poincaré model, the limit can be explicitly calculated and the Busemann function is given as:

$$b_{\mathbf{p}}(\mathbf{z}) = \log \frac{||\mathbf{p} - \mathbf{z}||^2}{(1 - ||\mathbf{z}||^2)}. \tag{7}$$

Our proposed loss combines the Busemann function with a penalty term:

$$\ell(\mathbf{z}, \mathbf{p}) = b_{\mathbf{p}}(\mathbf{z}) - \phi(d) \cdot \log(1 - ||\mathbf{z}||^2), \tag{8}$$

with $\phi(d)$ a scaling factor for the penalty term which is a function of the dimension of the hyperbolic space. The first term in the penalized Busemann loss steers a projected input $\mathbf{z}$ towards class prototype $\mathbf{p}$, while the second term penalized overconfidence, *i.e.,* values close to the ideal boundary of $\mathbb{B}_d$. The prototypes are positioned before training and remain fixed, hence the backpropagation only needs to be done with respect to the inputs. The gradient of $\ell(\exp_0(\mathbf{x}), \mathbf{p})$ with respect to $\mathbf{x}$ is given as:

$$\nabla_{\mathbf{x}}\, \ell(\exp_0(\mathbf{x}), \mathbf{p}) = (\mathbf{x} - \mathbf{p}) \frac{\tanh(||\mathbf{x}||)}{||\mathbf{x}|| - \tanh(||\mathbf{x}||)\, \mathbf{p} \cdot \mathbf{x}} + \mathbf{1}_d\, \mathbf{p} \cdot \mathbf{x} \frac{\tanh(||\mathbf{x}||)/||\mathbf{x}|| - 1}{||\mathbf{x}|| - \tanh(||\mathbf{x}||)\, \mathbf{p} \cdot \mathbf{x}} + \tanh(||\mathbf{x}||/2), \tag{9}$$

with $\mathbf{1}_d$ a d-dimensional vector of ones. Figure 2 shows a color map of the penalized Busemann loss itself and its radial gradient for a prototype located at ideal point $\mathbf{p} = (1, 0)$.

In the penalized Busemann loss, $\phi(d)$ governs the amount of regularization. Below, we show that this function should be linear in the number of dimensions.

Theorem 1. *For any $d \geq 4$ and ideal point $\mathbf{p}$ the penalized Busemann loss $\ell(\mathbf{z}, \mathbf{p})$ is the log-likelihood of a density $f(\mathbf{z}, \mathbf{p}) = \frac{1}{C(d)} \exp(-\ell(\mathbf{z}, \mathbf{p}))$ on the Poincaré ball $\mathbb{B}_d$ if and only if $\phi(d) > d - 2$.*

Proof. To show that $f(\mathbf{z}, \mathbf{p})$ is a density, we have to show that the normalization constant

$$C(d) = \int_{\mathbb{B}_d} \exp(-\ell(\mathbf{z}, \mathbf{p}))\, dV$$

is finite. Here, dV is the hyperbolic volume element on $\mathbb{B}_d$, which is equal to $2^d(1 - ||\mathbf{z}||^2)^{-d} dz_1 \cdots dz_d$, see [33]. Furthermore, using invariance of the integrand under rotation, we can assume without loss of generality that $\mathbf{p}$ is equal to the unit vector $e_1 = (1, 0, \ldots, 0)$, which leads to

$$C(d) = 2^d \int_{\mathbb{B}_d} \left(\frac{(1 - ||\mathbf{z}||^2)^{\phi(d)+1-d}}{1 - 2z_1 + ||\mathbf{z}||^2} \right) dz_1 \cdots dz_d.$$

Switching to (Euclidean) hyperspherical coordinates $(r, \varphi_1, \ldots, \varphi_d)$, the integral factorizes into $C(d) = 2^d \cdot C_1(d) \cdot C_2(d)$, where

$$C_1(d) = \int_0^1 \int_0^\pi \left(\frac{(1-r^2)^{\phi(d)+1-d}}{1 - 2r\cos(\varphi_1) + r^2} \right) r^{d-1} \sin(\varphi_1)^{d-2} dr\, d\varphi_1$$

$$C_2(d) = \int_0^\pi \sin(\varphi_2)^{d-3} d\varphi_2 \ldots \int_0^\pi \sin(\varphi_{d-2})^2 d\varphi_{d-2} \cdot \int_0^{2\pi} \sin(\varphi_{d-1}) d\varphi_{d-1}.$$

Clearly, $C_2(d)$ is finite, independently of d, such that it remains to consider $C_1(d)$. Instead of solving the integral exactly, we use the elementary estimate

$$\sin(\varphi_1)^2 \leq 1 - 2r\cos(\varphi_1) + r^2 \leq 4,$$

valid for all $r \in [0, 1]$. This estimate allows us to obtain both a lower and an upper bound on $C_1(d)$. The upper bound becomes

$$C_1(d) \leq \int_0^1 (1-r^2)^{\phi(d)+1-d} r^{d-1} dr \int_0^\pi \sin(\varphi_1)^{d-4} d\varphi_1,$$

which (given that $d \geq 4$) is finite if and only if $\phi(d) > d - 2$. The lower bound becomes

$$C_1(d) \geq \frac{1}{4} \int_0^1 (1-r^2)^{\phi(d)+1-d} r^{d-1} dr \int_0^\pi \sin(\varphi_1)^{d-2} d\varphi_1,$$

which (given that $d \geq 2$) is also finite if and only if $\phi(d) > d - 2$, which completes the proof. □

Given a trained model, inference is performed by projecting the test example $\mathbf{x}$ and selecting the class with the smallest prototype distance:

$$y^\star = \arg\min_{\mathbf{p} \in P} \ell(\exp_0(\mathcal{F}(\mathbf{x}; \theta)), \mathbf{p}). \tag{10}$$

As the penalty term is the same for all prototypes, we can equivalently minimize the Busemann function of the prototypes directly. Moreover, as the Busemann function is a decreasing function of the cosine similarity, prediction is equivalent to hyperspherical prototype inference:

$$y^\star = \arg\max_{\mathbf{p} \in P} \frac{\mathbf{z}}{||\mathbf{z}||} \cdot \mathbf{p}, \qquad \mathbf{z} = \exp_0(\mathcal{F}(\mathbf{x}; \theta)). \tag{11}$$

3.3 Relation to logistic regression

Logistic regression is a foundational algorithm for classification and a pillar in supervised learning for deep networks with cross entropy optimization. Here, we show that hyperbolic ideal prototype learning is a natural generalization of logistic regression by showing its equivalence when using $d = 1$ dimensions in our approach.

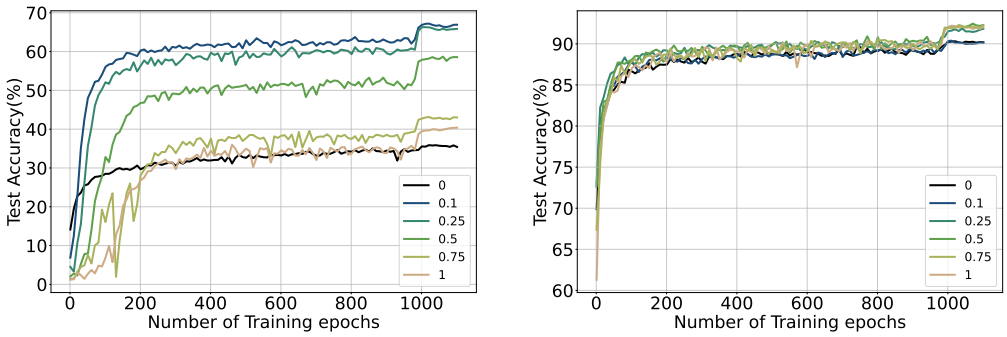
Theorem 2. *For dimension d = 1 and with penalty $\phi(1) = 1$, Hyperbolic Busemann Learning with a single linear layer is equivalent to logistic regression.*

Proof. Recall that from the machine learning point of view logistic regression can be described as a single linear layer in combination with the logistic activation function and cross-entropy-loss. In one dimension, hyperbolic space $\mathbb{B}_1$ becomes the interval $(-1, 1)$. The set of ideal points $\mathbb{I}_1$ consists of the two points ± 1. To identify these prototypes with $0/1$ (the class labels in logistic regression) we introduce a linear change of coordinates to

$$z' = \frac{z+1}{2}.$$

Under this change of coordinates $\mathbb{B}_1$ becomes the unit interval $(0, 1)$ and the prototypes p' (the ideal points) become the endpoints 0 and 1. The exponential map, i.e., the embedding of the base learner output $y = w^\top x + w_0$ into $\mathbb{B}_1$, maps $y \in \mathbb{R}$ to

$$z' = \exp_0(y) = \frac{\tanh(y/2) + 1}{2} = \frac{1}{1 + e^{-y}},$$



(a) CIFAR-100 (50D). (b) CIFAR-10 (5D).

Figure 3: **Ablation study on the penalized Busemann loss** for CIFAR-100 and CIFAR-10. On both datasets, adding a non-negative penalty (0.1 to 1) improves the classification results over the Busemann loss without penalty (0). With more classes, a lower penalty is preferred. The y-axis on CIFAR-10 starts from 60% to better visualize differences between settings.

which is the logistic function. In z'-coordinates, the penalized Busemann loss becomes

$$\ell(z', p') = 2\log\left(\frac{|p' - z'|}{2z'(1-z')}\right) = \begin{cases} 2\log\left(\frac{1}{2(1-z')}\right) & \text{if } p' = 0 \\ \log\left(\frac{1}{2z'}\right) & \text{if } p' = 1 \end{cases}.$$

Rewriting this as

$$\frac{1}{2}\ell(z', p') + \log(2) = -p'\log(z') - (1-p')\log(1-z'),$$

shows that the shifted and scaled penalized Busemann loss coincides with the cross-entropy loss in dimension $d = 1$. □

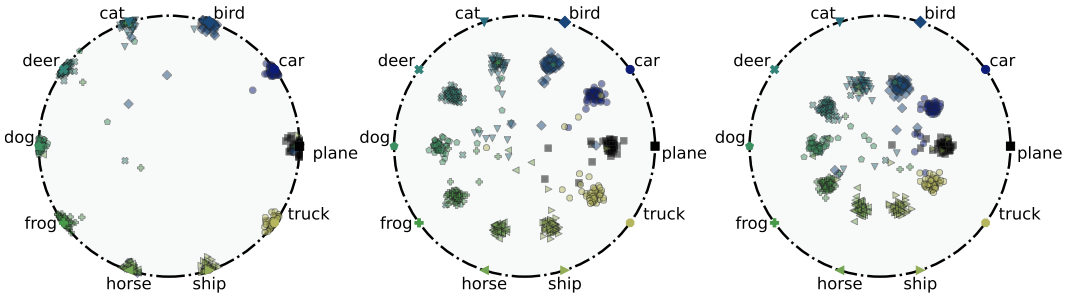
Beyond binary classification, multinomial logistic regression is performed in a one-vs-rest manner with a K-dimensional softmax output for K classes. , in conventional multinomial logistic regression the output space scales one-to-one with the number of classes. With Hyperbolic Busemann Learning, we provide a different generalization of logistic regression, where the number of output dimensions is independent of the number of classes. Most prominently, we find that most of the classification performance can be maintained in output spaces of small dimension, cf., Tables 1 and 3.

4 Experiments

4.1 Ablation studies

Setup. We first investigate the workings of the penalized Busemann loss. We evaluate the effect of the penalty term on CIFAR-10 and CIFAR-100 using a ResNet-32 backbone. For the experiment, we use Adam with a learning rate of 5e-4, weight decay of 5e-5, batch size of 128, without pre-training. The network is trained for 1,110 epochs with learning rate decay of 10 after 1,000 and 1,100 epochs. The prototypes are given by [28] without prior knowledge.

Results. In Figure 3, we show the results for CIFAR-100 and CIFAR-10 for various settings of the penalty term in the Busemann loss. The penalty term $\phi(d)$ depends on the output dimensionality and should be linear in form as per Theorem 1. In the Figure, we show five settings, corresponding to the slope s of the linear function, *i.e.,* $\phi(d; s) = s \cdot d$. On both datasets the setting with $s = 0$, where the penalty term is excluded, performs worst, highlighting the importance of including a penalty in the Busemann loss. For the other settings, the higher the value, the larger the penalty and the stronger the pull towards the origin of the output space. On CIFAR-10, all non-zero slopes improve the classification accuracy. Slope $s = 0.75$ is slightly preferred with an accuracy of 92.3% versus 91.9% for $s = 1$ and 90.1% for $s = 0$. For CIFAR-100, a too strong penalty negatively impacts the classification results. A penalty of $s = 0.1$ performs best with an accuracy of 65.8% compared



(a) slope = 0 (b) slope = 0.75 (c) slope = 1

Figure 4: **Visualization of the penalty term** and its effect on the output space with a ResNet-32 on CIFAR-10. Without a penalty, examples are close to the boundary, leading to overconfidence. Contrarily, a high penalty forces examples towards the origin of the Poincaré ball, leading to inter-class confusion. With the right slope, these factors are balanced, resulting in improved classification.

to 39.7% and 35.9% for $s = 1$ and $s = 0$. We conclude that the penalty term is an influential component of the proposed loss and with more classes a lower slope is recommended for best performance.

Analysis. To better understand the penalty term and its effect, we have trained a ResNet-32 backbone on CIFAR-10 using only two output dimensions. In Figure 4, we visualize the output space for three slope values, along with the ideal prototypes and test examples. The visualization clearly shows the importance of the penalty; without it, examples move closely towards the ideal prototypes leading to over-confidence. On the other hand, a high penalty pulls all examples close to the origin of the Poincaré ball. Especially with many classes, this leads to smaller inter-class variation and higher confusion. The visualization explains the lower performance for a high penalty on CIFAR-100 compared to CIFAR-10. By balancing confidence and inter-class separation, the best classification results are achieved. Figure 1 and Figure 4 show intuitively that the distance to the origin for an example reflects the confidence of the network in classifying the example. We have performed a quantitative analysis to show that our approach provides a natural interpretation of classification confidence, see Figure 5. In general, the closer the example to the origin, the lower the overall confidence, as also indicated in other hyperbolic approaches [18, 40]. The red line in Figure 5 shows that the distance to the origin in our approach also correlates with angular similarity to the correct class prototype. In other words, the larger the distance to the origin, the higher the likelihood of the classification being correct. Again, this provides a parallel to logistic regression, where the output score can be interpreted as a classification probability, conditioned on the input. We conclude from the analysis that the severity of the linear penalty matters to balance over-confidence and inter-class confusion, while the networks outputs enable a direct interpretation of the classification results.

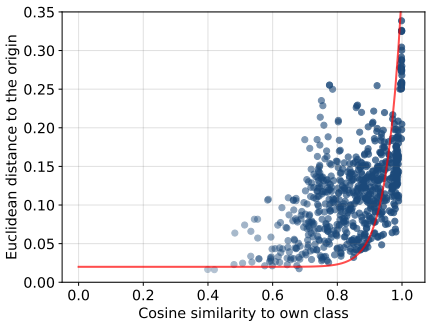


Figure 5: **Confidence interpretation.** The larger the distance to the origin, the more likely the example is correctly classified, showing that our hyperbolic approach provides a natural interpretation of classification confidence.

4.2 Comparison to hyperspherical prototypes

Setup. In the second experiment, we compare our approach to its hyperspherical counterpart, in which the output space is the positively curved hypersphere instead of negatively curved hyperbolic space. We draw a comparison to Hyperspherical Prototype Networks [28], which has shown to be competitive or better than conventional softmax cross-entropy and using Euclidean prototypes. For a fair comparison, we use the same prototypes and the same network backbone. The difference lies in the choice of manifold and accompanying loss function. Following [28], we report on CIFAR-100 and CIFAR-10 across multiple dimensionalities. On CIFAR-100, we use 3, 5, 10, 25, and 50 output dimensions, while we use 2, 3, 4, and 5 output dimensions for CIFAR-10.

Table 1: **Comparison to Hyperspherical Prototype Networks** on CIFAR-100 and CIFAR-10 using the same prototypes and a ResNet-32. * denotes results from code provided by [28] in case the number was not reported in the original paper. For both datasets, our approach obtains a high accuracy, especially with few output dimensions.

	CIFAR-100					**CIFAR-10**			
Dimensions	3	5	10	25	50	2	3	4	5
Mettes *et al.* [28]	5.5	28.7	51.1	63.0	64.7	30.0*	84.9*	88.8*	89.5*
This paper	**49.0**	**54.6**	**59.1**	**65.7**	**65.8**	**91.2**	**92.2**	**92.2**	**92.3**

Results. We show the classification results for CIFAR-100 and CIFAR-10 on ResNet-32 in Table 1. Across all dimensions, our approach performs better and the gap increases when using fewer dimensions. On CIFAR-100 for five output dimensions, we obtain 54.6% accuracy versus 28.7% for hyperspherical prototypes and the relative difference grows for three dimensions with 49.0% versus 5.5%. On CIFAR-10, our performance is hardly affected by using few dimensions. The accuracy for our approach in two dimensions is still 91.2% compared to 92.3% in five dimensions and 30.0% in two dimensions for hyperspherical prototypes. The high scores with few dimensions highlights the effectiveness of our approach and is in line with the effectiveness of hyperbolic embeddings in lower dimensionalities [29].

To show that the obtained results generalize to other network backbones and choice of prototypes, we report additional results in Table 2 for CIFAR-100. Using a DenseNet-121 architecture improves both our approach and the baseline with similar relative performance improvements with lower dimensionalities. We also investigate the addition of prior semantic knowledge in prototypes as per [28]. With these prototypes, the improvements with few dimensions holds and additionally boost the performance in high dimensions. With 50 dimensions, we obtain an accuracy of 68.0%, compared to 65.8% without using prior knowledge. In Table 3, we show additional comparisons to softmax cross-entropy and Euclidean prototypes, which re-iterate the potential of our approach both with and without prior knowledge for prototypes. Overall, we conclude that our approach is applicable and effective for classification regardless of backbone and choice of prototypes.

Table 3: **Comparative evaluation** to softmax cross-entropy, Euclidean prototypes, and hyperspherical prototypes on CIFAR-100 with ResNet-32. Baseline numbers taken from [28].

	Manifold	Prior knowledge	
		w/o	w/
Softmax CE	-	62.1	-
Guerriero *et al.* [16]	$\mathbb{R}^{100}$	62.0	-
Mettes *et al.* [28]	$\mathbb{S}^{99}$	65.0	65.2
This paper	$\mathbb{B}^{50}$	**65.8**	**68.0**

4.3 Comparison to hyperbolic prototypes

Setup. The second experiment has shown that Hyperbolic Busemann Learning boosts the classification results compared to its hyperspherical counterpart. In the third experiment, we draw a comparison to the recent hyperbolic action network [23], the only hyperbolic prototype approach to date which is not restricted to few-shot or zero-shot settings. In [23], action classes are embedded as prototypes in the interior of the Poincaré ball based on given hierarchical relations. In contrast, we position prototypes on the ideal boundary of the Poincaré ball, which can be done both with and without prior

Table 2: **Comparison with varying architecture and prototypes** on CIFAR-100. * denotes results from code provided by [28] in case the number was not reported in the original paper. Our approach is applicable to and effective for any choice of network backbone and choice of prototypes.

	DenseNet-121					**ResNet-32 w/ prior knowledge**				
Dimensions	3	5	10	25	50	3	5	10	25	50
Mettes *et al.* [28]	11.5*	45.1*	60.5	69.1	71.1*	11.5	37.0	57.0	64.0	65.2*
This paper	**66.1**	**69.0**	**71.1**	**72.7**	**71.8**	**52.2**	**56.4**	**62.4**	**65.9**	**68.0**

Table 4: **Hyperbolic prototypes comparison** on ActivityNet for the search by video name task. When using prototypes from hierarchical knowledge, we score best across the three metrics when using 10 output dimensions. Where [23] is restricted to prototypes from hierarchies, our approach can also employ knowledge-free prototypes, *e.g.,* from separation, to boost the standard mAP metric.

	Prototypes from hierarchies				**Prototypes from separation**			
Dimensions	2	10	20	50	2	10	20	50
Long *et al.* [28]								
mAP	26.9	68.3	66.9	67.1	-	-	-	-
s-mAP	72.0	93.8	92.7	93.1	-	-	-	-
c-mAP	91.9	97.7	97.2	97.5	-	-	-	-
This paper								
mAP	40.8	**70.6**	66.3	65.6	44.1	78.0	**79.4**	78.5
s-mAP	79.2	**95.1**	94.5	94.7	48.9	**88.6**	86.2	86.7
c-mAP	92.9	98.0	97.8	**98.2**	60.8	84.0	90.6	**91.7**

hierarchical knowledge. To showcase this possibility, we have performed a comparative evaluation on ActivityNet for the search by action name task. We use both the prototypes from prior hierarchical knowledge [23] with entailment cones [14] and knowledge-free prototypes from separation [28].

For our approach, we project the hierarchical prototypes on the ideal boundary through ℓ_2-normalization. We use the same pre-trained model for feature extraction and three-layer MLP for final classification as [23], with Adam as optimizer, a learning rate of 1e-4, and learning rate drop after 900 iterations. All other hyperparameters are the same as in previous experiments. Following [23], we report three metrics: mean Average Precision (mAP), sibling-mAP (s-mAP), which counts a prediction as correct if it is within 2-hops away in the hierarchy, and cousin-mAP (c-mAP), which does the same for 4-hops.

Results. The comparative evaluation on ActivityNet is shown in Table 4. Using the prototypes from hierarchical knowledge, we find that both our approach and the baseline favor low-dimensional embeddings. Our highest score for the mAP metric (70.6%) outperforms the baseline (68.3%) and the same holds for sibling-mAP (95.1% versus 93.8%) and cousin-mAP (98.2% versus 97.5%). These results highlight our ability to employ prototypes with prior knowledge. We note that the baseline performs additional tuning for the curvature in hyperbolic space, while we only use a standard curvature of one. A unique benefit of our approach is the ability to employ hyperbolic spaces without requiring hierarchical priors for prototypes. We find that positioning prototypes solely based on data-independent separation provides a boost in mAP, from 68.3% by [23] to 79.4%. In contrast, the hierarchical metrics score lower with the prototypes from separation. This is a direct result of the lack of hierarchical knowledge during prototype positioning. Overall, we conclude that Hyperbolic Busemann Learning provides both an effective and general approach for prototype-based learning with hyperbolic output spaces.

5 Conclusions

In this paper, we introduce Hyperbolic Busemann Learning with ideal prototypes. Where existing work on prototype-based learning in hyperbolic space requires prior knowledge to operate, we propose to place prototypes at the ideal boundary of the Poincaré ball. This enables a prototype positioning without the need for prior knowledge. We introduce the penalized Busemann loss to optimize examples with respect to ideal prototypes. We provide both theoretical and empirical support for the idea of ideal prototypes and the proposed loss. We prove its equivalence to logistic regression in dimension one and show on three datasets that our approach outperforms recent hyperspherical and hyperbolic prototype approaches. **Broader impact.** Our approach allows for both knowledge-based and knowledge-free prototypes and this choice matters depending on its broader utilization. Knowledge-free prototypes make fewer errors but the remaining ones are less hierarchically consistent. Hence critical applications such as in the medical domain benefit from the former, while search-based applications benefit form the latter.

References

[1] Rami Aly, Shantanu Acharya, Alexander Ossa, Arne Köhn, Chris Biemann, and Alexander Panchenko. Every child should have parents: a taxonomy refinement algorithm based on hyperbolic term embeddings. *ACL*, 2019.

[2] Gregor Bachmann, Gary Bécigneul, and Octavian Ganea. Constant curvature graph convolutional networks. In *ICML*, 2021.

[3] Ivana Balazevic, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. *NeurIPS*, 2019.

[4] Bjorn Barz and Joachim Denzler. Deep learning on small datasets without pre-training using cosine loss. In *WACV*, 2020.

[5] Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. *ICLR*, 2019.

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

[7] Joey Bose, Ariella Smofsky, Renjie Liao, Prakash Panangaden, and Will Hamilton. Latent variable modelling with hyperbolic normalizing flows. In *ICML*, 2020.

[8] Martin R Bridson and André Haefliger. *Metric spaces of non-positive curvature*. Springer Science & Business Media, 2013.

[9] Herbert Busemann. *The geometry of geodesics*. Academic Press, 1955.

[10] W. James Cannon, William J. Floyd, Richard Kenyon, and Walter R. Parry. Hyperbolic geometry. In Silvio Levy, editor, *Flavors of Geometry*, pages 59–115. MSRI Publications, 31 edition, 1997.

[11] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *NeurIPS*, 2019.

[12] Hyunghoon Cho, Benjamin DeMeo, Jian Peng, and Bonnie Berger. Large-margin classification in hyperbolic space. In *AISTATS*, 2019.

[13] Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. Graph prototypical networks for few-shot learning on attributed networks. In *CIKM*, 2020.

[14] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic entailment cones for learning hierarchical embeddings. In *ICML*, 2018.

[15] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *NeurIPS*, 2018.

[16] Samantha Guerriero, Barbara Caputo, and Thomas Mensink. Deep nearest class mean classifiers. In *ICLRw*, 2018.

[17] Martin Keller-Ressel and Stephanie Nargang. Hydra: a method for strain-minimizing hyperbolic embedding of network-and distance-based data. *Journal of Complex Networks*, 8(1), 2020.

[18] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *CVPR*, 2020.

[19] Anna Klimovskaia, David Lopez-Paz, Léon Bottou, and Maximilian Nickel. Poincaré maps for analyzing complex hierarchies in single-cell data. *Nature communications*, 11(1):1–9, 2020.

[20] Marc Law, Renjie Liao, Jake Snell, and Richard Zemel. Lorentzian distance learning for hyperbolic representations. In *ICML*, 2019.

[21] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *NeurIPS*, 2019.

[22] 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 *CVPR*, 2020.

[23] Teng Long, Pascal Mettes, Heng Tao Shen, and Cees GM Snoek. Searching for actions on the hyperbole. In *CVPR*, 2020.

[24] Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa. Differentiating through the fréchet mean. In *ICML*, 2020.

[25] Massimiliano Mancini, Hakan Karaoguz, Elisa Ricci, Patric Jensfelt, and Barbara Caputo. Knowledge is never enough: Towards web aided deep open world recognition. In *ICRA*, 2019.

[26] Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. *NeurIPS*, 2020.

[27] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *TPAMI*, 35(11):2624–2637, 2013.

[28] Pascal Mettes, Elise van der Pol, and Cees G.M. Snoek. Hyperspherical prototype networks. In *NeurIPS*, 2019.

[29] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *NeurIPS*, 2017.

[30] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *ICML*, 2018.

[31] Frederik Pahde, Mihai Puscas, Tassilo Klein, and Moin Nabi. Multimodal prototypical networks for few-shot learning. In *WACV*, 2021.

[32] Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. Transferrable prototypical networks for unsupervised domain adaptation. In *CVPR*, 2019.

[33] John Ratcliffe. *Foundations of hyperbolic manifolds*, volume 149. Springer Science & Business Media, 2006.

[34] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *ICLR*, 2018.

[35] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *ICML*, 2018.

[36] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *ISGD*, 2011.

[37] Jiayi Shen, Zehao Xiao, Xiantong Zhen, and Lei Zhang. Spherical zero-shot learning. *TCSVT*, 2021.

[38] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. *ICLR*, 2021.

[39] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *NeurIPS*, 2017.

[40] Dídac Surís, Ruoshi Liu, and Carl Vondrick. Learning the predictability of the future. In *CVPR*, 2021.

[41] William Thong, Pascal Mettes, and Cees G M Snoek. Open cross-domain visual search. *CVIU*, 2020.

[42] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincaré glove: Hyperbolic word embeddings. In *ICLR*, 2019.

[43] Andrew R Webb. *Statistical pattern recognition*. John Wiley & Sons, 2003.

[44] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, 2016.

[45] Wenjia Xu, Yongqin Xian, Jiuniu Wang, Bernt Schiele, and Zeynep Akata. Attribute prototype network for zero-shot learning. *NeurIPS*, 2020.

[46] Tao Yu and Chris De Sa. Numerically accurate hyperbolic embeddings using tiling-based models. In *NeurIPS*, 2019.

[47] Yunlong Yu, Zhong Ji, Jungong Han, and Zhongfei Zhang. Episode-based prototype generating network for zero-shot learning. In *CVPR*, 2020.

[48] Hongyi Zhang, Sashank J Reddi, and Suvrit Sra. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. In *NeurIPS*, 2016.

[49] Yiding Zhang, Xiao Wang, Chuan Shi, Nian Liu, and Guojie Song. Lorentzian graph convolutional networks. In *WWW*, 2021.

[50] Shichao Zhu, Shirui Pan, Chuan Zhou, Jia Wu, Yanan Cao, and Bin Wang. Graph geometry interaction learning. In *NeurIPS*, 2020.

[51] Yudong Zhu, Di Zhou, Jinghui Xiao, Xin Jiang, Xiao Chen, and Qun Liu. Hypertext: Endowing fasttext with hyperbolic geometry. In *EMNLP*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See abstract and introduction.
   (b) Did you describe the limitations of your work? [Yes] See conclusions.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See theorem 1 and 2.
   (b) Did you include complete proofs of all theoretical results? [Yes] See theorem 1 and 2.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See sections 4.1 and 4.3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See experiments.
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]