Horospherical Learning with Smart Prototypes

Paul Berg, Bjoern Michele, Minh-Tan Pham, Laetitia Chapel, Nicolas Courty

Horospherical Learning with Smart Prototypes

Paul Berg[1]
paul.berg@irisa.fr
Bjoern Michele[1,2]
bjoern.michele@valeo.ai
Minh-Tan Pham[1]
minh-tan.pham@irisa.fr
Laetitia Chapel[1,3]
laetitia.chapel@irisa.fr
Nicolas Courty[1]
nicolas.courty@irisa.fr

[1] IRISA
Université de Bretagne Sud
Vannes, France
[2] Valeo AI
Paris, France
[3] Institut Agro
Rennes, France

Abstract

Hyperbolic spaces have emerged as an effective manifold to learn representations due to their ability to efficiently represent hierarchical data structures, with little distortion, even for low-dimensional embeddings. In the chosen hyperbolic model, such as the Poincaré ball, classification is usually conducted by leveraging a signed distance function to the hyperbolic equivalent of a plane (gyroplanes) or by measuring the alignment to a virtual fixed prototype. We propose, in a deep learning context, to leverage a different characterization of a decision boundary: Horospheres, which are level-sets of the Busemann function. They are geometrically equivalent to spheres tangent to the boundary of the hyperbolic space on a virtual point akin to a prototype. Accordingly, we define a new horospherical layer that can be adapted to any neural network backbone. In previous works, prototypes are usually uniformly distributed without using a potentially available label hierarchy for the task at hand. We also propose a hierarchically informed method for positioning these prototypes, based on the Gromov-Wasserstein distance. We find that the combination of a good initialization and optimization of the prototypes improves the baseline performance for image classification on hierarchical datasets and in two semantic segmentation tasks, conducted on image and point cloud datasets. Source code will be released upon acceptance.

1 Introduction

Euclidean representations are a natural choice for deep representation learning methods because of their well defined distance and inner-product. Recently, other embedding spaces have emerged as attractive alternatives. Indeed, embedding into a different metric space can improve the representation capabilities [5]. Spherical representations, for instance, have had

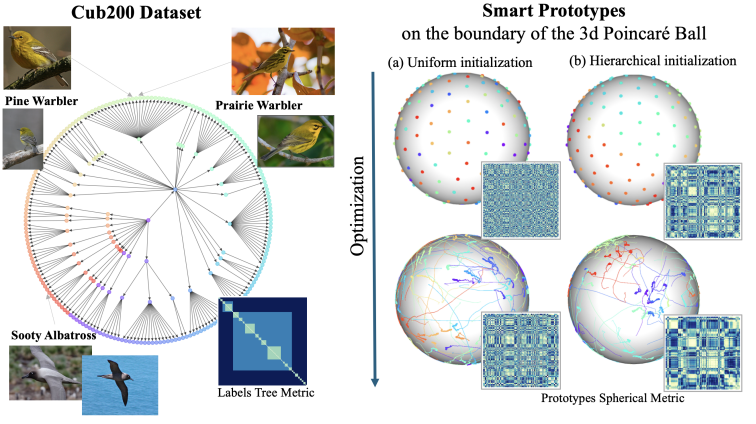

Figure 1: **Horospherical learning with smart prototypes.** We propose to position ideal prototypes following the label hierarchy and then optimize their position during training.

a large success in self-supervised representation learning [9, 53] and supervised learning [57] leveraging the so-called cosine similarity as a measure of alignment between samples representations. Other notable examples include the cone of Positive Semi-Definite (PSD) matrices (*e.g.* [42]) used for covariance matrices or symmetric spaces such as the Siegel space [55] that naturally embed graphs.

Among those alternative spaces, Hyperbolic spaces have attracted a lot of attention. Indeed, they were shown to be efficient at embedding trees with minimal distortion [44] compared to euclidean spaces. As a result, Hyperbolic spaces have successfully been used in several key machine learning tasks.

It is also often possible to derive a hierarchy from the labels of an image dataset. For example, the ImageNet dataset [13] labels are directly derived from the WordNet [38] hierarchy. However, label hierarchies are often unused by deep learning models which discriminate only among leaf nodes in the hierarchy. We consider that adding hierarchical information in the training process can be beneficial for the resulting performance.

In this article, we build upon the ability of hyperbolic spaces to embed trees with low distortion to better model hierarchical classification problems. Our contributions are the following:

- We introduce a hyperbolic classifier based on trainable horospheres parameterized by their prototype location and a bias term.
- This classifier can be combined with a hierarchically informed scheme to initially position horosphere prototypes at the boundary of the Hyperbolic space.
- Our experiments on image classification and semantic segmentation show that the proposed approach improves performance on hierarchical datasets compared to other hyperbolic baselines.

2 Related Work

Prototypical Networks. Prototypical networks rely on comparing the position of samples with prototypes and learning to project samples as close to the corresponding prototype as possible. While prototypical networks have been used with success in euclidean spaces [46] by computing prototypes as the centroid of representations for each class, research has expanded them to other embedding manifolds such as graphs [51], hyperspheres [37] or hyperbolic spaces [23, 25] where prototypes are positioned independently from the data or have their positions updated during training. The initial prototype positioning can sometimes be directed by prior information such as word embeddings of class labels [33, 37]. In a similar fashion, we consider prototypes which are not defined as centroids but instead live as points in the latent representation space.

Hierarchical Learning. Another active area of research in computer vision is leveraging label hierarchies to improve downstream performance [3] because it can cause the model to make less important mistakes. For example, misclassifying between birds of the same biological family may be less critical than between different biological orders [7]. The hierarchical information can be used either to design the classification layer itself or as a regularisation by post-processing predictions hover labels during training [14]. Hou et al. [27] leverage the Wasserstein distance to penalize predictions which do not respect the class ordering which can be seen as a sort of hierarchy. Similarly, Garnot et Landrieu. [21] propose to use the hierarchy present in datasets combined with prototypical networks in euclidean space by regularizing the training to limit the distortion between prototypes and the hierarchy. Atigh et al. [1] propose to compute probabilities for each node in the label hierarchy as posteriors for the actual class prediction with a hierarchical softmax.

Hyperbolic Spaces. Hyperbolic spaces have successfully been used in a number of text [15, 20, 40, 48], data classification [10, 17], and vision [1, 16, 22, 23] related tasks as an alternative to euclidean or other embedding spaces. Compared to euclidean spaces, a tree can be embedded in hyperbolic space with minimum distortion [24]. A number of hyperbolic neural network layers have been proposed [41] to replace their euclidean counterparts such as linear layers [19], convolutional layers [45] or even entire hyperbolic networks [49] where all activations are in hyperbolic space. In particular, hyperbolic representations show promising results in hierarchical tasks such as action recognition [34, 47] or image segmentation [1] where label hierarchies are well-defined. Most related to us, is Atigh et al. [23] who introduces Busemann learning with ideal prototypes. We build upon their method by proposing a hierarchically informed positioning scheme for ideal prototypes as well as introducing a trainable variant of the Busemann prototype classification layer.

3 Background on Hyperbolic Spaces

While euclidean spaces are often the embedding space of choice for representation learning, hyperbolic spaces have recently emerged as an attractive alternative embedding space [28, 40]. Hyperbolic spaces are Riemannian manifolds with constant negative curvature. There are multiple models to represent the hyperbolic spaces, in this paper we leverage the Poincaré Ball model which is defined as:

$$\mathcal{H}_d^c = \{x \in \mathbb{R}^d, \|x\|_2^2 \leq 1/c\}, \tag{1}$$

where $c > 0$. On the Poincaré ball, the hyperbolic distance between two points $x, y \in \mathcal{H}_c^d$ is defined as:

$$d_{\mathcal{H}}(x,y) = \frac{1}{\sqrt{c}}\cosh^{-1}\left(1 + 2c\frac{\|x-y\|_2^2}{(1-c\|x\|_2^2)(1-c\|y\|_2^2)}\right). \tag{2}$$

In order to project between euclidean and hyperbolic space, one has to resort to exponential and log maps which project from, respectively to, the tangent space at a defined point. The tangent space at any point of the Poincaré ball is the euclidean space of the same dimension, which means that points in euclidean space can be projected on the Poincaré ball using the exponential map $\exp_p^c : \mathbb{R}^d \to \mathcal{H}_d^c$.

In euclidean spaces, linear layers are often used as the last layer of a model to discriminate samples among different classes. A linear layer is composed of a set of hyperplanes, each responsible for an output feature of the layer. The value for each feature is computed from the distance to the hyperplane. Ganea et al. [19] introduced a derivation of euclidean hyperplanes in the hyperbolic space, more specifically the Poincaré ball model, called hyperbolic gyroplanes. An example of gyroplanes for classification on the Poincaré disk can be seen in Figure 2. Ideal prototypes [23] have also been proposed as a method to perform hyperbolic classification. An ideal prototype p is positioned at the infinite of the hyperbolic space. In the Poincaré ball model, these so-called ideal prototypes are located on the surface of the corresponding hypersphere $p \in \{x \in \mathbb{R}^d; \|x\| = 1\}$. The Busemann function can then be used to measure a distance between p and any point $z \in \mathcal{H}_d^c$. It is defined as such:

$$B_{\mathrm{p}}(z) = \lim_{t \to +\infty} d(\gamma_{\mathrm{p}}(t), z) - t, \tag{3}$$

where $\gamma_{\mathrm{p}} : [0, +\infty) \to \mathcal{H}_d^c$ is a geodesic ray with p at the infinity such $\gamma_{\mathrm{p}}(+\infty) = \mathrm{p}$. During training, the model tries to minimize the Busemann distance between samples and the prototype corresponding to the ground truth label.

4 Hierarchically Informed Horospherical Classification

4.1 Horospherical Classifiers

In the article introducing hyperbolic classification based on ideal prototypes [23], authors left the prototypes fixed during training, thus giving a lot of importance to the prototype initialisation step. In our work, we propose to update the weight position and add an additional scalar parameter for each horosphere induced by the prototypes. This parameter acts analogously to the bias term in euclidean hyperplanes. Therefore, a parameterized horosphere can be defined from its prototypes $\mathrm{p} \in \mathbb{S}^{d-1}$ and its bias term $a \in \mathbb{R}$. We define this horosphere level-set as:

$$H_{\mathrm{p},a} = \{x \in \mathcal{H}_d^c, -B_{\mathrm{p}}(x) + a = 0\}, \tag{4}$$

where B_p is the Busemann function defined in Equation 3. In the Poincaré ball model, the Busemann function has the following form:

$$B_{\mathrm{p}}(x) = \log\left(\frac{\|\mathrm{p}-x\|_2^2}{1-\|x\|_2^2}\right) \tag{5}$$

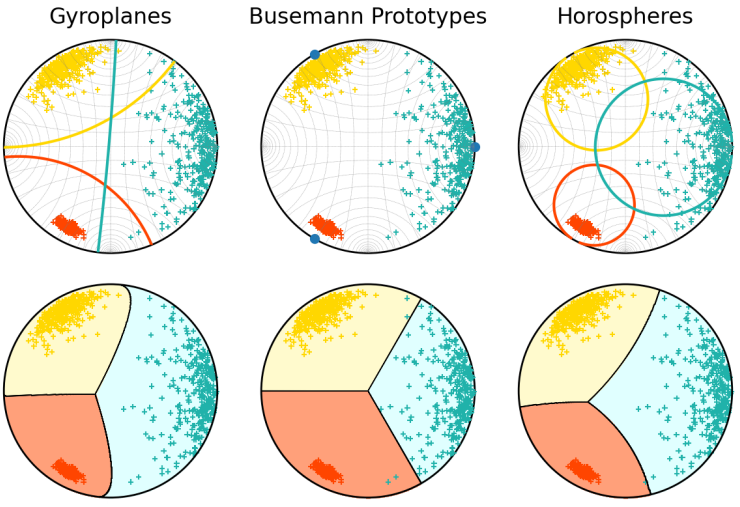

Figure 2: **Illustration of three hyperbolic classification methods** on a simple 3-classes toy dataset. The first row depicts Gyroplanes [19] (first column), that are arc of circles orthogonal to the boundaries of the Poincaré ball, Busemann prototypes [1] (second column), represented in Blue, that are fixed points located at infinity, and our Horospherical classifiers (third column), depicted as circles tangent to the boundary in one infinite prototypical point. The second row illustrates the resulting decision function.

and therefore the horosphere $H_{\mathrm{p},a}$ has a diameter of $1+\tanh(a/2)$ (see Supplementary).

In order to prevent samples from moving too close to the boundary, one common solution is to use euclidean clipping before projecting in hyperbolic space [24] to restrict representations to only a subset of the Poincaré ball. However even with this clipping, the majority of representations end up close to the new boundary such that only comparing samples based on their orientation can be enough [39]. In Ghadimi Atigh et al. [23], authors instead use an additional regularisation term which prevents representations' radii from growing too close to the boundary. We include a similar regularisation, not as a loss term, but directly in the logit $\xi_{\mathrm{p},a}$ for an hypersphere parameterized with p, a:

$$\xi_{\mathrm{p},a}(x) = -B_{\mathrm{p}}(x) + a + \phi(d) \cdot \log\left(1 - \|x\|_2^2\right), \tag{6}$$

where $\phi(d)$ weights the impact of this regularisation depending on the dimension d of the hyperbolic space $\mathcal{H}_d^c$. Adding this regularisation to the scoring function $\xi_{\mathrm{p},a}$ penalizes points from moving too close to the boundary and therefore the level set $\{x \in \mathcal{H}_d^c, \xi_{\mathrm{p},a}(x) = 0\}$ is not strictly an hypersphere tangent to the Poincaré ball at point p. Note that with $\phi(d) = 0$, we fall back to the parameterized definition from Equation 4. In practice and similarly to Ghadimi Atigh et al. [23], we use a single scalar hyper-parameter $\lambda \geq 0$ which is fixed during training to parameterize regularisation such that $\phi(d) = \lambda \times d$.

While the logit $\xi_{\mathrm{p},a}$ can be used to perform binary classification by computing $P(\hat{y}=1) = \sigma(\xi_{\mathrm{p},a}(x))$, with $\sigma : \mathbb{R}^d \to [0,1]$ being the logistic function. Horospheres can be adapted in a multi-class classification setting by having an horosphere per class and using a softmax normalization to obtain class membership probabilities. An horospherical layer consists of

K horospheres and we define the probability of a sample $x \in \mathcal{H}^c_d$ with respect to one of the K classes as the softmax over the K horosphere logits:

$$P(\hat{y} = k \mid x, \mathrm{p}, a) = \frac{\exp(\xi_{\mathrm{p}_k, a_k}(x))}{\sum_{l=1}^{K} \exp(\xi_{\mathrm{p}_l, a_l}(x))}, \forall k \in \{1..K\}. \tag{7}$$

This horospherical multi-class classifier can be trained using the negative log-likelihood loss over the training data x, y:

$$\mathcal{L}_{\mathrm{p},a}(x, y) = \frac{1}{N} \sum_{i=1}^{N} -\log P(\hat{y} = y_i \mid x_i, \mathrm{p}, a). \tag{8}$$

We can replace the last linear layer of a euclidean classification fully-connected layer with an exponential map to project in the Poincaré ball followed by an evaluation of the horospherical layer in hyperbolic space.

4.2 Uniformly Distributed Ideal Prototypes

The horospherical classifier presented in Section 4.1 is initialized with a set of ideal prototypes. In order to leverage the embedding space, we initialize the layer with prototypes uniformly distributed around the hypersphere $\mathbb{S}^{d-1} = \{x \in \mathbb{R}^d, \|x\|_2 = 1\}$. Computing uniform positions for a set of points on the hypersphere can be done in different manners [4, 53]. We chose to optimize a set of K points $\{\mathrm{p}_i \in \mathbb{S}^{d-1}; i = 1, \ldots, K\}$ on the sphere using the uniform loss proposed by Wang and Isola [53]. This uniform loss is based on the pairwise Gaussian potential. It effectively maximises the pairwise distance between each pairs $\mathrm{p}_i, \mathrm{p}_j$:

$$\mathcal{L}_{\mathrm{unif}}(\mathrm{p}) = \log \left(\frac{1}{K(K-1)} \sum_{i \neq j} \exp(-\|\mathrm{p}_i - \mathrm{p}_j\|_2^2) \right). \tag{9}$$

After optimization, this set of points can be used as ideal prototypes in the Poincaré ball model. Whereas the Busemann classifier [23] randomly assigns prototypes to class labels, in the following section, we consider the problem of assigning a class label to each prototype in a hierarchically informed manner to improve the performance of the classification model.

4.3 Hierarchical Prototypes Assignment

Given a classification task, a label hierarchy encodes the semantic relationship between classes. The hierarchy can be represented as a tree, where ancestors in the tree correspond to super-classes. The depth of the hierarchy corresponds to the longest path from the root to any leaves. Label hierarchies can be built from the semantic and lexical relationships between labels as is the case in the WordNet hierarchy [38] which is the base for the ImageNet [13] label hierarchy. In this section, we investigate how to leverage a label hierarchy to position ideal prototypes to minimize the distortion between the prototypes and the tree induced by the hierarchy.

Our goal is to assign a prototype for each leaf node in the label hierarchy. However, labels are placed in the hierarchy tree whereas prototypes are located on the hypersphere. These two spaces are incomparable, that is, one cannot easily define a metric to compute a distance between a label on the tree induced by the hierarchy and an ideal prototype on the hypersphere. If both labels and prototypes were embedded in the same space, they could be

compared and one could use Optimal Transport [43, 50] to compute a transport plan between labels and prototypes.

Garnot and Landrieu [21] work around this problem by introducing a scale-free variant of their comparison loss. Instead, in this context of incomparable spaces, one can resort to the Gromov-Wasserstein distance [36] (GW). The GW distance is used to compute a transport plan P between samples from two empirical distributions whose supports are embedded in incomparable spaces. The GW transport plan is computed by comparing the inner metrics from each distribution. Therefore, the transport plan is based on the inner structure from each distribution instead of on the pairwise distances between samples from each distribution. Given two empirical distributions $\mu = \sum_{i=1}^{n} \delta_{x_i} a_i \in \mathcal{P}(\mathcal{X})$ and $\nu = \sum_{j=1}^{n'} \delta_{y_j} b_j \in \mathcal{P}(Y)$ over two metric spaces $(\mathcal{X}, d_{\mathcal{X}})$ and $(\mathcal{Y}, d_{\mathcal{Y}})$ respectively with a and b, two simplex histograms of n and n' bins respectively such that $\sum_{i=1}^{n} a_i = 1$ and $\sum_{j=1}^{n'} b_j = 1$. δ_x is the Dirac measure in x. M_1 and M_2 are the matrices of pairwise distance between atoms of μ and ν respectively. The GW distance solves the following problem:

$$GW_p^p(M_1, M_2, a, b) = \min_{P \in U(a,b)} \sum_{i,j,k,l} |(M_1)_{i,k} - (M_2)_{j,l}|^p P_{i,j} P_{k,l}, \tag{10}$$

where $U(a,b)$ is the set of linear transport constraints defined by a and b. That is, the set of doubly stochastic matrices which respect the marginal constraints a and b, $U(a,b) = \left\{P \mid P \geq 0, P1_{|a|} = a, P^\top 1_{|b|} = b\right\}$.

Therefore, we place prototypes uniformly on the hypersphere and assign the prototype using the Gromov-Wasserstein transport plan between the tree metric and the metric of distances on the sphere. Given $M_{\mathbb{T}}$ and $M_{\mathbb{S}}$, the distance metrics and the tree and on the sphere respectively, one can compute the optimal transport plan between atoms from $\mathbb{T}$ and $\mathbb{S}$:

$$P^\star = \operatorname*{argmin}_{P \in U(a,b)} \sum_{i,j,i',j'} |(M_{\mathbb{T}})_{i,i'} - (M_{\mathbb{S}})_{j,j'}|^2 P_{i,j} P_{i',j'} \tag{11}$$

where $a_i = b_i = 1/K$ are uniform marginal distributions for both labels in the hierarchy and prototypes on the sphere respectively.

The tree metric is taken to be the length of the path between two nodes in the hierarchy. Therefore, two leaves i and j with the same parent will have a distance $(M_{\mathbb{T}})_{i,j} = 2$. For the sphere metric, we use the cosine distance $(M_{\mathbb{S}})_{i,j} = 1 - \langle \mathrm{p}_i, \mathrm{p}_j \rangle$ which is an increasing function of the length of the arc between p_i and p_j and $0 \leq (M_{\mathbb{S}})_{i,j} \leq 2$.

Since both $M_{\mathbb{S}}$ and $M_{\mathbb{T}}$ have the same cardinality, the optimal transport plan $P^\star$ is a permutation matrix which can be use to assign prototypes to nodes in the hierarchy. Therefore, label i will be assigned to prototype $j^\star$ as such:

$$j^\star = \operatorname*{argmax}_{j \in \{1..K\}} P^\star_{i,j}. \tag{12}$$

Additionally, we also consider the simpler case of positioning hierarchical prototypes on the Poincaré disk in two dimensions. Firstly, uniformly distributing points on the circle can be done in a closed form by slicing the circle in arcs of equal length $2\pi/K$ as explained in Section 4.2. The hierarchical structure can also be embedded on the circle without having to compute an optimal transport plan, by using the graph visualization algorithm used in Figure 1 called *twopi* introduced by Wills [54]. We rely on this method in 2 dimensions.

In order to visualize the impact of this positioning scheme, we can compare the pairwise distance matrices between the tree hierarchy, uniform prototypes and the prototypes

Table 1: Results on image classification.

Dataset	CIFAR10 [ ]				CIFAR100 [ ]				CUB200 [ ]			
Dimension	2	3	4	50	2	3	4	50	2	3	4	50
Euclidean	89.8±0.2	90.6±0.1	90.8±0.2	**91.0±0.1**	45.9±0.5	53.3±0.8	56.2±0.2	62.5±0.3	24.3±1.3	46.8±0.7	50.5±1.5	51.8±0.8
Hyperbolic Neural Networks [ ]	86.7±5.4	90.2±0.3	90.4±0.2	89.3±0.6	38.7±0.7	43.9±3.8	44.5±0.2	55.9±0.5	24.4±6.0	22.3±12.0	16.9±1.5	33.6±2.1
Metric Guided Prototypes [ ]	89.8±0.3	90.4±0.2	90.6±0.3	91.0±0.2	54.2±0.8	57.9±0.6	58.9±0.4	62.4±0.4	31.0±2.8	48.8±1.4	53.2±1.9	**57.3±0.7**
Busemann - Uniform [ ]	89.7±0.3	90.6±0.2	90.7±0.2	88.6±0.1	49.7±0.7	48.7±2.3	50.2±1.5	63.3±0.6	27.6±7.4	40.0±2.0	45.0±1.7	49.5±3.7
Busemann - Smart	89.8±0.1	90.4±0.5	90.7±0.1	88.6±0.1	**55.7±0.5**	54.8±0.4	55.4±0.3	**63.3±0.3**	**53.3±1.6**	54.3±0.4	**58.0±1.1**	50.4±0.6
Horospherical - Smart	**90.1±0.3**	**90.7±0.2**	**90.8±0.1**	90.8±0.2	53.7±0.8	**58.7±0.3**	**60.0±0.1**	60.9±0.1	38.7±4.3	**56.4±1.6**	56.3±0.8	56.6±0.6

positioned using Gromov-Wasserstein in Figure 1. Before training, the uniformly distributed prototypes do not exhibit any structure in the metric whereas after optimization, the distance between prototypes appear less noisy. Before optimization, the hierarchically positioned prototypes already exhibit a block structure in the spherical metric which by construction has similarities with the tree metric because their relative distortion is minimized.

5 Experiments

5.1 Image Classification

To evaluate the performance of our hyperbolic classification method, we first conduct experiments on hierarchical image classification datasets. We choose the CUB-200 [52] which contains 200 bird species. It is composed of 5994 training images and a test set of 5794 images. Each species belongs to a single family which in turn belongs to an order. This gives us a hierarchy of depth 3 with 252 nodes, which can be seen on Figure 1. We also experiment on the CIFAR10 [32] and CIFAR100 [32] which contains 10 and 100 classes of common visual objects, respectively. We compare the performance of the horospherical classifier with Hyperbolic Busemann learning [23], both with either uniformly distributed prototypes and hierarchical prototypes. We also compare them with the Hyperbolic neural networks [19] as well as the Metric Guided euclidean prototype learning [21]. We test different dimensions d for the embedding space.

As the backbone, we use a ResNet32 [26]. Models are trained for 1110 epochs for CIFAR datasets and 2110 for the CUB200 dataset, as done in [23]. The Riemannian variant [2] of the Adam optimizer [29] implemented in the Geoopt toolbox [30] is used for optimizing the parameters for the Horospherical method. For other methods without trainable parameters on manifolds, we use the Adam optimizer. A curvature of $c = 1.0$ is set for all hyperbolic methods. The Gromov-Wasserstein transport plan for the hierarchical positioning is computed using the Python Optimal Transport toolbox [18]. All experiments are done with 3 runs.

Results can be seen in Table 1. We can see that the performance is improved with hierarchically positioned prototypes and horospherical classifier, especially when the embedding dimension is small such as with $d \leq 4$. On its own, the performance is improved for the Busemann method when initialized with hierarchically-positioned prototypes.

In order to evaluate a hierarchical metric, we measure the Average Hierarchical Cost (AHC) [31] on the CUB200 dataset. This metric averages the hierarchical distance between predictions and the ground-truth by taking the length of the path on the hierarchy between the prediction and ground-truth label.

Given predictions $z_i \in \{1,\dots,C\}$ from a model of N test samples and ground $y_i \in \{1,\dots,C\}$ from the dataset with $i \in \{1,\dots,N\}$, The average hierarchical cost is defined as:

Table 2: Average Hierarchical Cost of different methods on the CUB200 dataset.

Method	AHC↓	
	$d = 4$	$d = 50$
Euclidean	2.12	2.18
Metric-Guided Prototypes [21]	1.72	**1.80**
Busemann - Uniform [23]	**1.63**	2.07
Horospherical - Smart	**1.63**	**1.80**

Table 3: Results on semantic segmentation (%mIoU).

Dataset	NuScenes [6]			Cityscapes [12]		
Dimension	$d = 2$	$d = 3$	$d = 128$	$d = 2$	$d = 3$	$d = 128$
Euclidean	54.0±2.4	67.4±0.3	**70.5±0.5**	35.9 ± 0.0	60.9±4.6	**78.8±0.4**
HIS [23]	40.2±5.4	58.0±2.7	69.5±0.2	41.3±4.8	45.1±7.7	77.9±0.2
Horospherical - Smart	**68.4±0.1**	**68.7±0.0**	69.2±0.3	**73.5±0.4**	**76.1±0.1**	78.2±0.4

$$AHC(z,y) = \frac{1}{N}\sum_{i=1}^{N}(M_{\mathbb{T}})_{y_i,z_i}. \tag{13}$$

This means that an average hierarchical cost of 0 is equivalent to having made predictions all equal to the ground-truth and the average distance is therefore 0. Correspondingly, a lower hierarchical cost means a better hierarchical performance. The results for the AHC experiment are shown in Table 2 where we can see that the AHC is better for hierarchically-aware method and specifically hyperbolic methods in lower dimensions. Our proposed horospherical classifier performs on par with the best method of Metric-Guided in higher dimensions ($d = 50$).

5.2 Image and Point Cloud Segmentation

In this section, we evaluate the proposed method on semantic segmentation tasks, for both 2D image and 3D point cloud datasets. We compare with the euclidean baseline as well as the Hyperbolic Image Segmentation (HIS) method from Atigh et al. [1].

For image segmentation, we consider the Cityscapes dataset [12] which contains 19 classes divided in 7 super-classes (*flat, construction, object, nature, sky, human*, and *vehicle*). We use a DeepLab-v3+ [8] backbone trained with the official split for 240 epochs. We set $\lambda = 0.5$. For all hyperbolic methods, c is set to 1.0. Experiments are done with 3 runs and results can be observed in Table 3.

For point cloud segmentation, we perform experiments on the NuScenes [6] dataset which contains 40k frames, sampled in Boston and Singapore, with a rotating LiDAR. Points are classified into 16 semantic classes and 1 ignore class. We use the official train split for training, and report the results obtained on the publicly available validation set. For the class hierarchy we leverage the NuScenes class description and split into 5 branches (*movable object, vehicle, pedestrian, flat* and *static*). We additionally split the vehicle branch into 4-wheeled and 2-wheeled. This leads to a hierarchy with 3 layers. As a point cloud processing backbone, we use the commonly used sparse-voxel Minkowski U-Net [11], with a 10 cm voxel size. We set $\lambda = 0.1$. Results can be seen in Table 3.

Table 4: Ablation study by freezing prototypes on our proposed horospherical method on semantic segmentation.

Prototypes	Frozen	NuScenes [6]			Cityscapes [12]		
		$d=2$	$d=3$	$d=128$	$d=2$	$d=3$	$d=128$
Uniform	✓	59.6	64.9	**67.2**	63.2	66.8	**76.9**
Smart	✓	**64.7**	**65.9**	67.1	**66.4**	**72.9**	74.8

Table 5: Ablation study by changing the initialization scheme between uniformly distributed and smart prototypes for our proposed horospherical classifier.

Dataset	CUB200				NuScenes [6]			Cityscapes [12]		
Dimension	$d=2$	$d=3$	$d=4$	$d=50$	$d=2$	$d=3$	$d=128$	$d=2$	$d=3$	$d=128$
Metric	%Acc	%Acc	%Acc	%Acc	%mIoU	%mIoU	%mIoU	%mIoU	%mIoU	%mIoU
Horospherical - Uniform	21.5±3.3	43.2±2.4	46.3±1.9	55.9±0.8	67.4	**68.9**	68.8	73.3	75.7	78.3
Horospherical - Smart	**38.7±4.3**	**56.4±1.6**	**56.3±0.8**	**56.6±0.6**	**68.4**	68.7	**69.2**	**73.8**	**76.0**	**78.6**

5.3 Ablation Study

In order to evaluate the impact of the prototype positions during training, we compare the performance with freezing them during training. This means that the last layer is, akin to [23], parameter-less. In this setting, we expect that the initial prototypes positions will have a larger impact than when the prototypes are trained.

This training setup offers similarities with Busemann learning [23] which also has frozen prototypes. The horospherical layer is different because it uses probabilities computed with Equation 7 along with the cross-entropy loss defined in Equation 8. Results from the ablation study can be seen in Table 4. We can see that the initialization has more impact when the representation dimension is small. We also compare the training performance of initializing our proposed horospherical classifier with uniformly distributed prototypes instead of hierarchically positioned prototypes. Results can be seen in Table 5. In this case, we can see that hierarchical prototypes perform better than uniformly distributed prototypes.

6 Conclusion

We have shown how prototype learning in hyperbolic spaces can be improved using hierarchically informed prototype positioning based on a measure of distance between the hyperbolic boundary and the label hierarchy. Our proposed classifier can be used in several hyperbolic classification tasks such as image classification, image segmentation and point cloud segmentation. Based on our experiments, our method has shown improvements over other euclidean and hyperbolic classification methods, in particular for small embedding dimensions. Future work can be oriented towards scaling this classifier in higher dimensions.

Funding

This work is funded by the ANR AI chair OTTOPIA under reference ANR-20-CHIA-0030. It was granted access to the HPC resources of IDRIS under the allocations 2023-AD011013839R1 and 2024-AD011013514R2 made by GENCI.

References

[1] Mina Ghadimi Atigh, Julian Schoep, Erman Acar, Nanne Van Noord, and Pascal Mettes. Hyperbolic image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4453–4462, 2022.

[2] Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. *arXiv preprint arXiv:1810.00760*, 2018.

[3] Luca Bertinetto, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A Lord. Making better mistakes: Leveraging class hierarchies with deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12506–12515, 2020.

[4] Clément Bonet, Paul Berg, Nicolas Courty, François Septier, Lucas Drumetz, and Minh-Tan Pham. Spherical sliced-wasserstein. In *International Conference on Learning Representations*, 2023.

[5] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[6] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[7] Dongliang Chang, Kaiyue Pang, Yixiao Zheng, Zhanyu Ma, Yi-Zhe Song, and Jun Guo. Your" flamingo" is my" bird": fine-grained, or not. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11476–11485, 2021.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[10] 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.

[11] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] Ankit Dhall, Anastasia Makarova, Octavian Ganea, Dario Pavllo, Michael Greeff, and Andreas Krause. Hierarchical image classification using entailment cone embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 836–837, 2020.

[15] Bhuwan Dhingra, Christopher J Shallue, Mohammad Norouzi, Andrew M Dai, and George E Dahl. Embedding text in hyperbolic spaces. *arXiv preprint arXiv:1806.04313*, 2018.

[16] 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.

[17] Xiran Fan, Chun-Hao Yang, and Baba Vemuri. Horospherical decision boundaries for large margin classification in hyperbolic space. *Advances in Neural Information Processing Systems*, 36, 2024.

[18] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, et al. Pot: Python optimal transport. *The Journal of Machine Learning Research*, 22(1):3571–3578, 2021.

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

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

[21] Vivien Sainte Fare Garnot and Loic Landrieu. Leveraging class hierarchies with metric-guided prototype learning. *arXiv preprint arXiv:2007.03047*, 2020.

[22] Songwei Ge, Shlok Mishra, Simon Kornblith, Chun-Liang Li, and David Jacobs. Hyperbolic contrastive learning for visual representations beyond objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6840–6849, 2023.

[23] 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.

[24] 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.

[25] Manal Hamzaoui, Laetitia Chapel, Minh-Tan Pham, and Sébastien Lefèvre. A hierarchical prototypical network for few-shot remote sensing scene classification. In *International Conference on Pattern Recognition and Artificial Intelligence*, pages 208–220. Springer, 2022.

[26] 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.

[27] Le Hou, Chen-Ping Yu, and Dimitris Samaras. Squared earth mover's distance-based loss for training deep neural networks. *arXiv preprint arXiv:1611.05916*, 2016.

[28] 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.

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

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

[31] Aris Kosmopoulos, Ioannis Partalas, Eric Gaussier, Georgios Paliouras, and Ion Androutsopoulos. Evaluation measures for hierarchical classification: a unified view and novel approaches. *Data Mining and Knowledge Discovery*, 29:820–865, 2015.

[32] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[33] 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.

[34] 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.

[35] Federico López, Beatrice Pozzetti, Steve Trettel, Michael Strube, and Anna Wienhard. Symmetric spaces for graph embeddings: A finsler-riemannian approach. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 7090–7101, 2021.

[36] Facundo Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11:417–487, 2011.

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

[38] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[39] Gabriel Moreira, Manuel Marques, João Paulo Costeira, and Alexander Hauptmann. Hyperbolic vs euclidean embeddings in few-shot learning: Two sides of the same coin. *arXiv preprint arXiv:2309.10013*, 2023.

[40] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017.

[41] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. Hyperbolic deep neural networks: A survey. *IEEE Transactions on pattern analysis and machine intelligence*, 44(12):10023–10044, 2021.

[42] Xavier Pennec. 3 - manifold-valued image processing with spd matrices. In Xavier Pennec, Stefan Sommer, and Tom Fletcher, editors, *Riemannian Geometric Statistics in Medical Image Analysis*, pages 75–134. Academic Press, 2020.

[43] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[44] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International symposium on graph drawing*, pages 355–366. Springer, 2011.

[45] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. *arXiv preprint arXiv:2006.08210*, 2020.

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

[47] Dídac Surís, Ruoshi Liu, and Carl Vondrick. Learning the predictability of the future. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12607–12617, 2021.

[48] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincar\'e glove: Hyperbolic word embeddings. *arXiv preprint arXiv:1810.06546*, 2018.

[49] Max van Spengler, Erwin Berkhout, and Pascal Mettes. Poincar\'e resnet. *arXiv preprint arXiv:2303.14027*, 2023.

[50] Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

[51] Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty. Template based graph neural network with optimal transport distances. In *Advances in Neural Information Processing Systems*, volume 35, pages 11800–11814, 2022.

[52] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[53] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020.

[54] Graham J Wills. Nicheworks—interactive visualization of very large graphs. *Journal of computational and Graphical Statistics*, 8(2):190–212, 1999.

Supplementary Materials - Horospherical Learning with Smart Prototypes

A Impact of $\phi(d)$ regularization

In order to evaluate the impact of the regularization introduced in Equation 6 we perform an experimental study of varying the value of the $\phi(d)$ parameter. Since we use a constant regularization defined as $\phi(d) = \lambda \times d$, we vary the value of the λ parameter in the range $[0, 2]$.

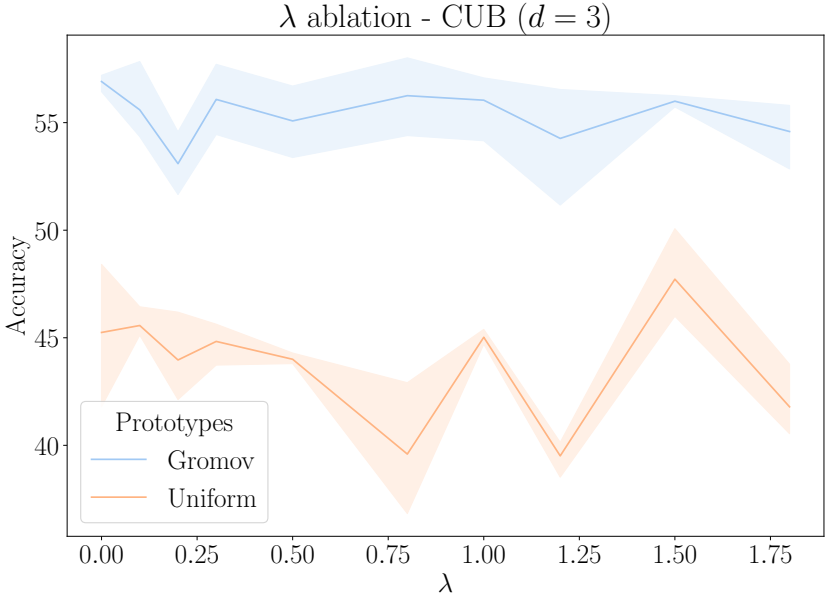

Figure 3: Evolution of the accuracy when moving the λ regularisation parameter. We observe that varying the value of the parameter has little impact on the resulting performance.

B Impact of the bias a

The bias a in Equation 6 acts on the radius of each horosphere. We experiment with disabling the bias parameter in order to evaluate its impact on the final performance of the model. As can be seen in Table 6, the bias provides in 2 dimensions an increase in performance but seems to be of less importance for higher dimensions.

C Radius of an Horosphere

Given an horosphere parameterized by an ideal prototype $\mathrm{p} \in \mathbb{S}^{d-1}$ and a bias $a \in \mathbb{R}$. We can compute the radius of the said horosphere. Remember that horospheres in the Poincaré ball

Table 6: Performance when enabling or disabling biases during training of horospherical classifiers on the CUB dataset.

Method	Bias	Dimensions 2	3	4	50
Horospherical - Smart		34.4±6.2	56.4±0.8	56.0±0.4	**57.5±0.2**
Horospherical - Smart	✓	**38.7±4.3**	56.4±1.6	**56.3±0.8**	56.6±0.6

model are hyperspheres tangent to the boundary of the ball. To compute the radius dependent on a, we will find the two points of the horosphere which are located on the Poincaré ball radius, one of this point is p, and we refer to the other one as x. By taking p as our first base vector, x has a single non-null dimension x_0.

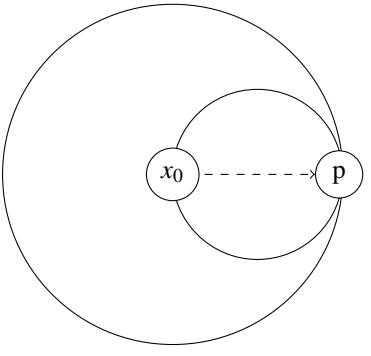

Figure 4: The position of x along the vector p is x_0.

$$-B_{\mathrm{p}}(x)+a=0 \tag{14}$$

$$\log\left(\frac{\|\mathrm{p}-x\|^2}{1-\|x\|^2}\right)-a=0 \tag{15}$$

$$\log\left(\exp(-a)\frac{\|\mathrm{p}-x\|^2}{1-\|x\|^2}\right)=0 \tag{16}$$

$$\exp(-a)\frac{\|\mathrm{p}-x\|^2}{1-\|x\|^2}=1 \tag{17}$$

$$\|\mathrm{p}-x\|^2=\frac{1-\|x\|^2}{\exp(-a)} \tag{18}$$

$$(1-x_0)^2=\frac{1-x_0^2}{\exp(-a)} \tag{19}$$

$$1-2x_0+x_0^2=\frac{1-x_0^2}{\exp(-a)} \tag{20}$$

$$1-2x_0+x_0^2=(1-x_0^2)\exp(a) \tag{21}$$

$$1-\exp(a)-2x_0+(1+\exp(a))x_0^2=0. \tag{22}$$

We find the roots for this polynomial:

$$\Delta = (-2)^2 - 4(1-\exp(a))(1+\exp(a)) \tag{23}$$
$$= 4 - 4(1-\exp(2a)) \tag{24}$$
$$= 4\exp(2a). \tag{25}$$

$$x_0 = \frac{2 \pm \sqrt{\Delta}}{2(1+\exp(a))} \tag{26}$$
$$= \frac{2 \pm \sqrt{4\exp(2a)}}{2(1+\exp(a))} \tag{27}$$
$$= \frac{2 \pm 2\sqrt{\exp(2a)}}{2(1+\exp(a))} \tag{28}$$
$$= \frac{2(1 \pm \exp(a))}{2(1+\exp(a))} \tag{29}$$
$$= \frac{1 \pm \exp(a)}{1+\exp(a)} \tag{30}$$
$$= \begin{cases} 1. & \text{if } x = \text{p.} \\ -\tanh(a/2) & \text{otherwise.} \end{cases} \tag{31}$$

Therefore, the radius of an horosphere with a bias term a is:

$$r(a) = \frac{1+\tanh(a/2)}{2}. \tag{32}$$

D Hierarchies

In this section, we include the hierarchies used for positioning prototypes in our experiments on different datasets. For large hierarchies, we do not include the label of nodes for readability purpose. The nodes are coloured following the topological sort of the tree.

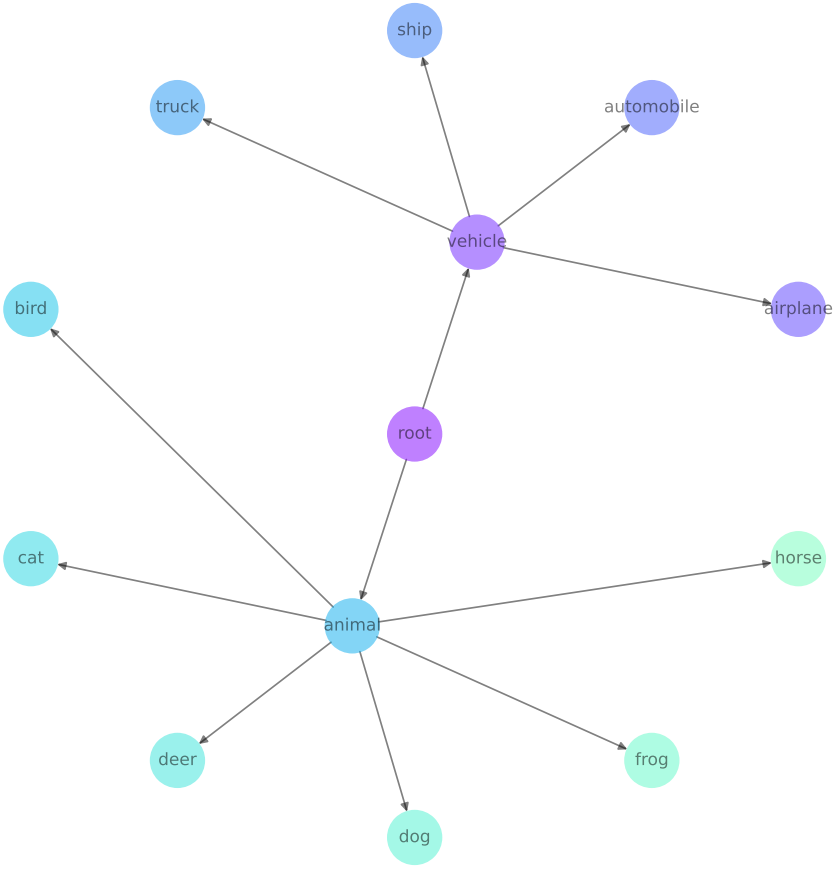

Figure 5: CIFAR10 Hierarchy.

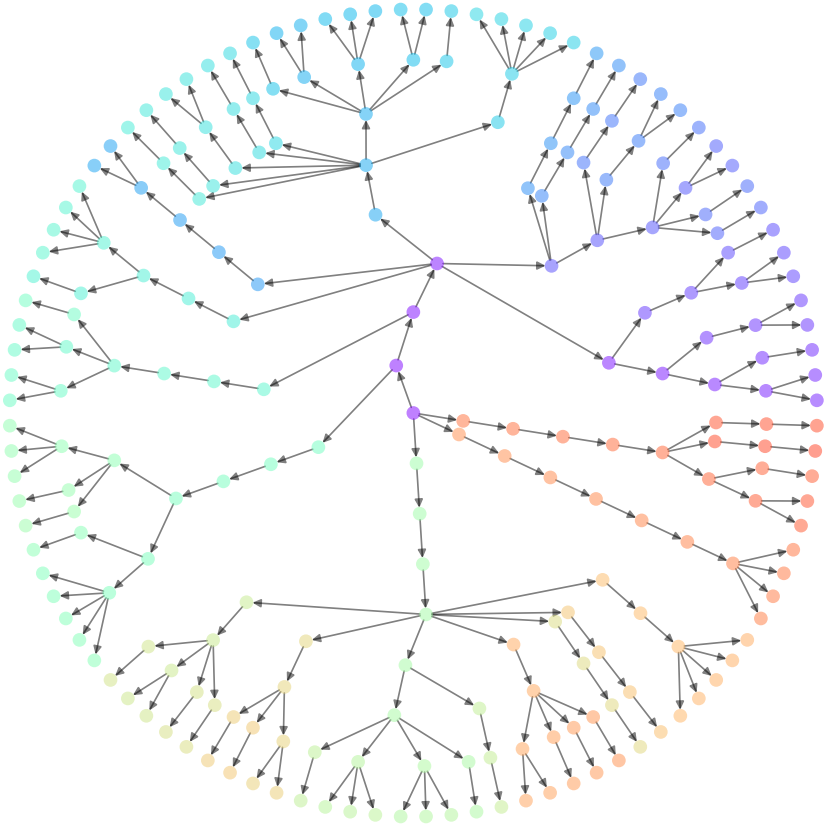

Figure 6: CIFAR100 Hierarchy.

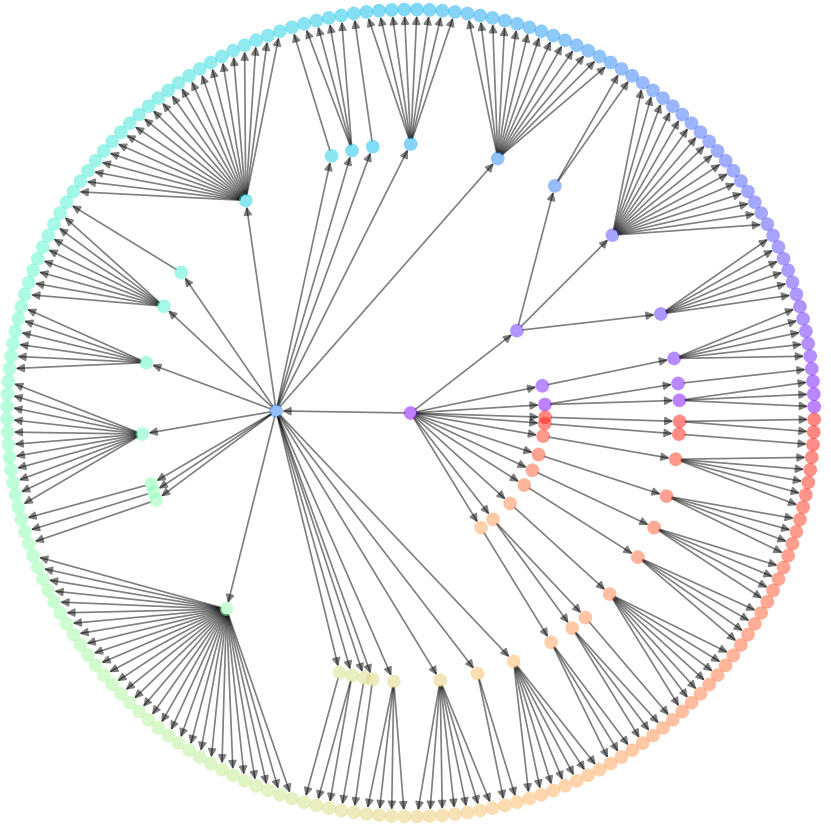

Figure 7: CUB200 Hierarchy.

Figure 8: NuScenes Hierarchy.

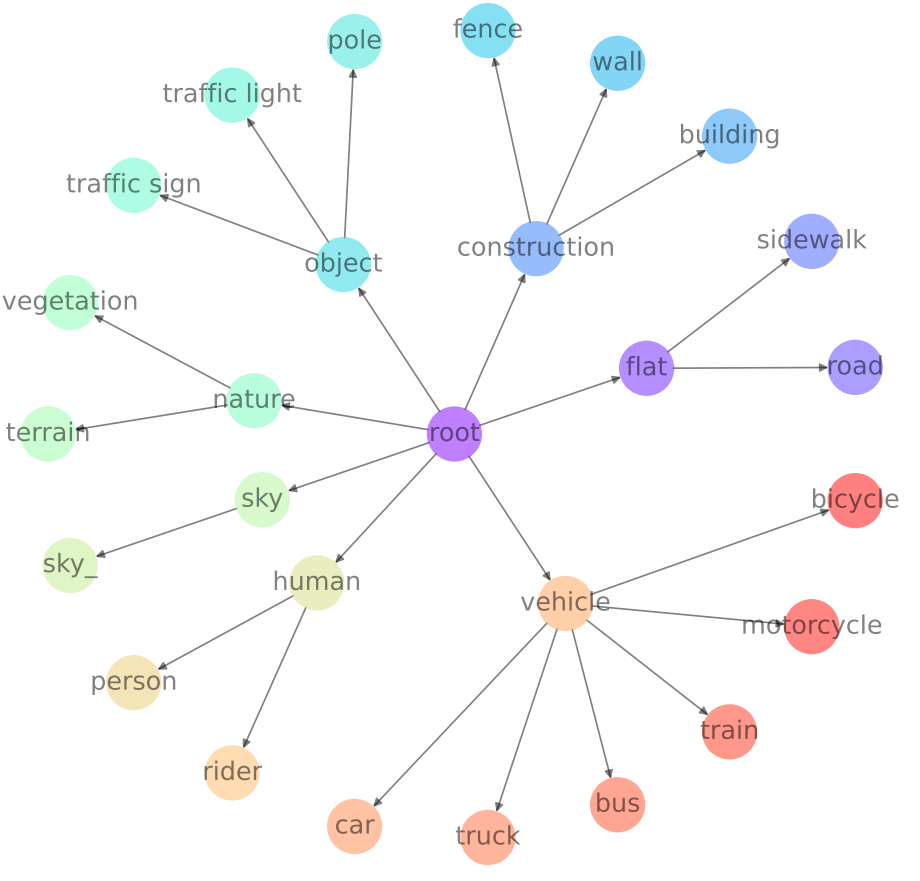

Figure 9: Cityscapes Hierarchy.