

# Hyperbolic Vision Transformers: Combining Improvements in Metric Learning

Aleksandr Ermolov

University of Trento, Italy

aleksandr.ermolov@unitn.it

Leyla Mirvakhabova

Skoltech\*, Russia

leyla.mirvakhabova@skoltech.ru

Valentin Khruikov

Yandex & Skoltech\*, Russia

khruikov.v@gmail.com

Nicu Sebe

University of Trento, Italy

niculae.sebe@unitn.it

Ivan Oseledets

Skoltech\* & AIRI, Russia

I.Oseledets@skoltech.ru

## Abstract

*Metric learning aims to learn a highly discriminative model encouraging the embeddings of similar classes to be close in the chosen metrics and pushed apart for dissimilar ones. The common recipe is to use an encoder to extract embeddings and a distance-based loss function to match the representations – usually, the Euclidean distance is utilized. An emerging interest in learning hyperbolic data embeddings suggests that hyperbolic geometry can be beneficial for natural data. Following this line of work, we propose a new hyperbolic-based model for metric learning. At the core of our method is a vision transformer with output embeddings mapped to hyperbolic space. These embeddings are directly optimized using modified pairwise cross-entropy loss. We evaluate the proposed model with six different formulations on four datasets achieving the new state-of-the-art performance. The source code is available at [https://github.com/htdt/hyp\\_metric](https://github.com/htdt/hyp_metric).*

## 1. Introduction

Metric learning task formulation is general and intuitive: the obtained distances between data embeddings must represent semantic similarity. It is a typical cognitive task to generalize similarity for new objects given some examples of similar and dissimilar pairs. Metric learning algorithms are widely applied in various computer vision tasks: content-based image retrieval [32, 46, 47], near-duplicate detection [65], face recognition [27, 44], person re-identification [5, 63], as a part of zero-shot [47] or few-shot learning [38, 45, 50].

Modern image retrieval methods can be decomposed into roughly two components: the encoder mapping the image to its compact representation and the loss function govern-

ing the training process. Encoders with backbones based on transformer architecture have been recently proposed as a competitive alternative to previously used convolutional neural networks (CNNs). Transformers lack some of CNN’s inductive biases, *e.g.*, translation equivariance, requiring more training data to achieve a fair generalization. On the other hand, it allows transformers to produce more general features, which presumably can be more beneficial for image retrieval [3, 8], as this task requires generalization to unseen classes of images. To alleviate the issue above, several training schemes have been proposed: using a large dataset [7], heavily augmenting training dataset and using distillation [53], using self-supervised learning scenario [3].

The choice of the embedding space directly influences the metrics used for comparing representations. Typically, embeddings are arranged on a hypersphere, *i.e.* the output of the encoder is  $L_2$  normalized, resulting in using cosine similarity as a distance. In this work, we propose to consider the hyperbolic spaces. Their distinctive property is the exponential volume growth with respect to the radius, unlike Euclidean spaces with polynomial growth. This feature makes hyperbolic space especially suitable for embedding tree-like data due to increased representation power. The paper [42] shows that a tree can be embedded to Poincaré disk with an arbitrarily low distortion. Most of the natural data is intrinsically hierarchical, and hyperbolic spaces suit well for its representation. Another desirable property of hyperbolic spaces is the ability to use low-dimensional manifolds for embeddings without sacrificing the model accuracy and its representation power [34].

The goal of the loss function is straightforward: we want to group the representations of similar objects in the embedding space while pulling away representations of dissimilar objects. Most loss functions can be divided into two categories: proxy-based and pair-based [23]. Additionally to the network parameters, the first type of losses trains proxies, which represent subsets of the dataset [32]. This proce-

\*Skolkovo Institute of Science and Technology

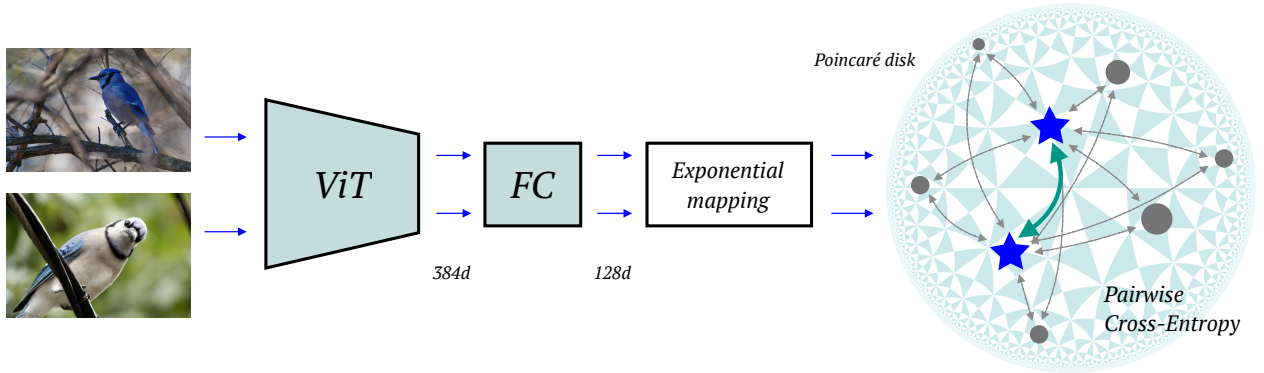


Figure 1. Overview of the proposed method. Two images representing one class (positives) are encoded with the vision transformer, projected into a space of a lower dimension with a fully connected (FC) layer, and then mapped to a hyperbolic space. Blue stars depict the resulting embeddings. Poincaré disk is shown with uniform triangle tiling on the background to illustrate the manifold curvature. Gray circles represent other samples from the batch (negatives). Finally, arrows in the disk represent distances used in the pairwise cross-entropy loss. Positives are pushed closer to each other, negative are pulled far apart.

cedure can be seen from a perspective of a simple classification task: we train matching embeddings, which would classify each subset [33]. At the same time, pair-based losses operate directly on the embeddings. The advantage of pair-based losses is that they can account for the fine-grained interactions of individual samples. Such losses do not require data labels: it is sufficient to have pair-based relationships. This property is crucial for a widely used pairwise cross-entropy loss in self-supervised learning scenario [4, 17, 55]. Instead of labels, the supervision comes from a pretext task, which defines positive and negative pairs. Inspired by these works, we adopt pairwise cross-entropy loss for our experiments.

The main contributions of our paper are the following:

- We propose to project embeddings to the Poincaré ball and to use the pairwise cross-entropy loss with hyperbolic distances. Through extensive experiments, we demonstrate that the hyperbolic counterpart outperforms the Euclidean setting.
- We show that the joint usage of vision transformers, hyperbolic embeddings, and pairwise cross-entropy loss provides the best performance for the image retrieval task.

## 2. Method

We propose a new metric learning loss that combines representative expressiveness of the hyperbolic space and the simplicity and generality of the cross-entropy loss. The suggested loss operates in the hyperbolic space encouraging the representatives of one class (*positives*) to be closer while pushing the samples from other categories (*negatives*) away.

The schematic overview of the proposed method is depicted at Figure 1. The remainder of the section is organized as follows. We start with providing the necessary preliminaries on hyperbolic spaces in Section 2.1, then we discuss the loss function in Section 2.2 and, finally, we briefly describe the architecture and discuss pretraining schemes in Section 2.5.

### 2.1. Hyperbolic Embeddings

Formally, the  $n$ -dimensional hyperbolic space  $\mathbb{H}^n$  is a Riemannian manifold of constant negative curvature. There exist several isometric models of hyperbolic space, in our work we stick to the Poincaré ball model  $(\mathbb{D}_c^n, g^{\mathbb{D}})$  with the curvature parameter  $c$  (the actual curvature value is then  $-c^2$ ). This model is realized as a pair of an  $n$ -dimensional ball  $\mathbb{D}^n = \{x \in \mathbb{R}^n : c\|x\|^2 < 1, c \geq 0\}$  equipped with the Riemannian metric  $g^{\mathbb{D}} = \lambda_c^2 g^E$ , where  $\lambda_c = \frac{2}{1-c\|x\|^2}$  is the *conformal factor* and  $g^E = \mathbf{I}_n$  is Euclidean metric tensor. This means, that local distances are scaled by the factor  $\lambda_c$  approaching infinity near the boundary of the ball. This gives rise to the ‘space expansion’ property of hyperbolic spaces. While, in the Euclidean spaces, the volume of an object of a diameter  $r$  scales polynomially in  $r$ , in the hyperbolic space, such volumes scale *exponentially* with  $r$ . Intuitively, this is a continuous analogue of trees: for a tree with a branching factor  $k$ , we obtain  $O(k^d)$  nodes on the level  $d$ , which in this case serves as a discrete analogue of the radius. This property allows us to efficiently embed hierarchical data even in low dimensions, which is made precise by embedding theorems for trees and complex networks [42].

Hyperbolic spaces are not vector spaces; to be able to perform operations such as addition, we need to introduce a

so-called gyrovector formalism [54]. For a pair  $\mathbf{x}, \mathbf{y} \in \mathbb{D}_c^n$ , their addition is defined as

$$\mathbf{x} \oplus_c \mathbf{y} = \frac{(1 + 2c\langle \mathbf{x}, \mathbf{y} \rangle + c\|\mathbf{y}\|^2)\mathbf{x} + (1 - c\|\mathbf{x}\|^2)\mathbf{y}}{1 + 2c\langle \mathbf{x}, \mathbf{y} \rangle + c^2\|\mathbf{x}\|^2\|\mathbf{y}\|^2}. \quad (1)$$

The hyperbolic distance between  $\mathbf{x}, \mathbf{y} \in \mathbb{D}_c^n$  is defined in the following manner:

$$D_{hyp}(\mathbf{x}, \mathbf{y}) = \frac{2}{\sqrt{c}} \operatorname{arctanh}(\sqrt{c}\|\mathbf{x} \oplus_c \mathbf{y}\|). \quad (2)$$

Note that with  $c \rightarrow 0$  the distance function (2) reduces to Euclidean:  $\lim_{c \rightarrow 0} D_{hyp}(\mathbf{x}, \mathbf{y}) = 2\|\mathbf{x} - \mathbf{y}\|$ .

We also need to define a bijection from Euclidean space to the Poincaré model of hyperbolic geometry. This mapping is termed *exponential* while its inverse mapping from hyperbolic space to Euclidean is called *logarithmic*.

For some fixed base point  $\mathbf{x} \in \mathbb{D}_c^n$ , the exponential mapping is a function  $\exp_{\mathbf{x}}^c: \mathbb{R}^n \rightarrow \mathbb{D}_c^n$  defined as:

$$\exp_{\mathbf{x}}^c(\mathbf{v}) = \mathbf{x} \oplus_c \left( \tanh \left( \sqrt{c} \frac{\lambda_{\mathbf{x}}^c \|\mathbf{v}\|}{2} \right) \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|} \right). \quad (3)$$

The base point  $\mathbf{x}$  is usually set to  $\mathbf{0}$  which makes formulas less cumbersome and empirically has little impact on the obtained results.

To train our model, we take a sample  $\mathbf{x}_i$ , pass it through the encoder and project the output to hyperbolic space; the resulted representation in hyperbolic space is denoted as  $\mathbf{z}_i$ . Since our pairwise cross-entropy loss is based on hyperbolic distances, we do not project  $\mathbf{z}_i$  back to Euclidean space and use only the exponential mapping.

## 2.2. Pairwise Cross-Entropy Loss

At each iteration, we sample  $N$  different categories of images and two samples per category. In this case, the total number of samples (batch size) is  $K = 2N$  consisting of  $N$  positive pairs.

Additionally to hyperbolic distance, we define the distance with the cosine similarity, implemented with a squared Euclidean distance between normalized vectors:

$$D_{cos}(\mathbf{z}_i, \mathbf{z}_j) = \left\| \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|_2} - \frac{\mathbf{z}_j}{\|\mathbf{z}_j\|_2} \right\|_2^2 = 2 - 2 \frac{\langle \mathbf{z}_i, \mathbf{z}_j \rangle}{\|\mathbf{z}_i\|_2 \cdot \|\mathbf{z}_j\|_2} \quad (4)$$

The loss function for a positive pair  $(i, j)$  is defined as

$$l_{i,j} = -\log \frac{\exp(-D(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1, k \neq i}^K \exp(-D(\mathbf{z}_i, \mathbf{z}_k)/\tau)}, \quad (5)$$

where  $D$  is a distance ( $D_{hyp}$  or  $D_{cos}$ ) and  $\tau$  is a temperature hyperparameter. The total loss is computed for all positive pairs, both  $(i, j)$  and  $(j, i)$ , in a batch.

	CUB-200	Cars-196	SOP	In-Shop
ViT-S	0.280	0.339	0.271	0.313
DeiT-S	0.294	0.343	0.270	0.323
DINO	0.315	0.327	0.301	0.318

Table 1.  $\delta$ -hyperbolicity values calculated for the embeddings obtained from different encoders. We can see that the  $\delta$  values are fairly consistent with respect to different feature extractors. Lower  $\delta$  values indicate a higher degree of data hyperbolicity.

If the total number of categories is small and a larger batch size is more suitable from the optimization perspective, it is possible to sample more than two samples per category. In this case, we sample  $d$  images per each category,  $d \geq 2$ . We divide the batch  $K = dN$  into  $d$  subsets with each subset consisting of  $N$  samples from different categories. Next, we obtain the loss value for each pair of subsets, as defined Equation (5), summing them up for the final value.

## 2.3. $\delta$ -hyperbolicity

While the curvature value of an underlying manifold for embedding is often neglected, a more efficient way is to estimate it for each dataset specifically. Following the analysis in [22], we estimate a ‘measure’ of the data hyperbolicity. This evaluation is made through the computation of the so-called Gromov  $\delta$ . Its calculation requires first computing *Gromov product* for points  $x, y, z \in \mathcal{X}$ :

$$(y, z)_x = \frac{1}{2}(d(x, y) + d(x, z) - d(y, z)), \quad (6)$$

where  $(\mathcal{X}, d)$  is an arbitrary metric space. For a set of points, we compute the matrix  $M$  of pairwise Gromov products (6). The  $\delta$  value is then defined as the largest entry in the matrix  $(M \otimes M) - M$ . Here,  $\otimes$  denotes the min-max matrix product defined as  $(A \otimes B)_{ij} = \max_k \min\{A_{ik}, B_{kj}\}$  [10].

Being rescaled between 0 and 1, the relative  $\delta$ -hyperbolicity reflects how close to the hyperbolic the hidden structure is: values tending to 0 show the higher degree of intrinsic data hyperbolicity. The  $\delta$  value is related to the optimal radius of the Poincaré ball for embeddings through the following expression  $c(X) = (\frac{0.144}{\delta})^2$ . We adopt the procedure described in [22] and evaluate  $\delta$  for image embeddings extracted using three encoders: ViT-S, DeiT-S and DINO (described in Section 2.5). Tab. 1 highlights the obtained relative  $\delta$  values for CUB-200, Cars-196, SOP and In-Shop datasets.

## 2.4. Feature Clipping

The paper [15] empirically shows that a hyperbolic neural network tends to have vanishing gradients since it pushes

the embeddings close to the boundary of Poincaré ball, making the gradients of Euclidean parameters vanish. To avoid numerical errors when dealing with hyperbolic neural networks, the common approach is to perform clipping by norm on the points in the Poincaré ball; the standard norm value is  $\frac{1}{\sqrt{c}}(1 - 10^{-5})$ . Instead, the paper [15] proposes to augment this procedure with an additional technique called *feature clipping*:

$$\mathbf{x}_C^E = \min \left\{ 1, \frac{r}{\|\mathbf{x}^E\|} \right\} \cdot \mathbf{x}^E, \quad (7)$$

where  $\mathbf{x}^E$  lies in the Euclidean space,  $\mathbf{x}_C^E$  is its clipped counterpart and  $r$  is a new *effective radius* of the Poincaré ball. Intuitively, this allows us to push embeddings further away from the boundary and avoid the vanishing gradients problem; in the experiments of [15] it led to a consistent improvement over baselines.

## 2.5. Vision Transformers

In our experiments, we use ViT architecture introduced by [7]. The input image is sliced into patches of size  $16 \times 16$  pixels. Each patch is flattened and then linearly projected into an embedding. The resulting vectors are concatenated with position embeddings. Also, this set of vectors includes an additional “classification” token. Note that in our case, this token is used to obtain the image embedding, but we do not train a standard classifier as in [7]. For consistency with previous literature, we name this token [class]. The set of resulting vectors is fed into a standard transformer encoder [56]. It consists of several layers with multiheaded self-attention (MSA) and MLP blocks, with a LayerNorm before and a residual connection after each block. The output for the transformer encoder for the [class] token is used as the final image representation. For more details, we refer to [7].

ViT-S [48] is a smaller version of ViT with 6 heads in MSA (base version uses 12 heads). This architecture is similar to ResNet-50 [18] in terms of number of parameters (22M for ViT-S and 23M for ResNet-50) and computational requirements (8.4 FLOPS for ViT-S and 8.3 FLOPS for ResNet-50). This similarity makes it possible to fairly compare with previous works based on ResNet-50 encoder, for this reason, we employ this configuration for our experiments. A more thorough description is available in [48].

Vision transformers, compared to CNNs, require more training signal. One solution, as proposed in [7], is to use a large dataset. ImageNet-21k [6] contains approximately 14M images classified into 21K categories. ViT-S, pretrained on ImageNet-21k, is publicly available [48]; we include it in our experiments. Another solution, DeiT-S [53], is based on the same (ViT-S) architecture and is trained on a smaller ImageNet-1k dataset [41] (a subset of ImageNet-21k consisting of about 1.3M training im-

ages and 1K categories). An additional training signal is provided by teacher-student distillation, with a CNN-based teacher [53].

The third solution used in our experiments, DINO [3], is based on self-supervised training. In this case, the model ViT-S is trained on the ImageNet-1k dataset [41] without labels. The encoder must produce consistent output for different parts of an image, obtained using augmentations (random crop, color jitter, and others). This training scheme is in line with the image retrieval task; in both cases, the encoder is explicitly trained to produce similar output for semantically similar input. However, the goal of these tasks is different: self-supervised learning provides pretrained features, which are then used for other downstream tasks, while for image retrieval resulting features are directly used for the evaluation.

## 3. Experiments

We follow a widely adopted training and evaluation protocol [23] and compare several versions of our method with current state-of-the-art on four benchmark datasets for category-level retrieval. We include technical details of datasets, our implementation and training details, and finally, present empirical results. There are two types of experiments, first, we compare with the state-of-the-art, and then we investigate the impact of hyperparameters (encoder patch size, manifold curvature, embedding size and batch size).

### 3.1. Datasets

**CUB-200-2011** (CUB) [61] includes 11,788 images with 200 categories of bird breeds. The training set corresponds to the first 100 classes with 5,864 images, and the remaining 100 classes with 5,924 images are used for testing. The images are very similar; some breeds can only be distinguished by minor details, making this dataset challenging and, at the same time, informative for the image retrieval task. **Cars-196** (Cars) [25] consists of 16,185 images representing 196 car models. First 98 classes (8,054 images) are used for training and the other 98 classes (8,131 images) are held out for testing. **Stanford Online Product** (SOP) [47] consists of 120,053 images of 22,634 products downloaded from eBay.com. We use the standard split: 11,318 classes (59,551 images) for training and remaining 11,316 classes (60,502 images) for testing. **In-shop Clothes Retrieval** (In-Shop) [28] consists of 7,986 categories of clothing items. First 3,997 categories (25,882 images) are for training, the remaining 3,985 categories are used for testing partitioned into a query set (14,218 images) and a gallery set (12,612 images).

### 3.2. Implementation Details

We use ViT-S [48] as an encoder with three types of pre-training (ViT-S, DeiT-S and DINO), details are presented in Section 2.5. The linear projection for patch embeddings as a first basic operation presumably corresponds to low-level feature extraction, so we freeze it during fine-tuning. The encoder outputs a representation of dimensionality 384, which is further plugged into a head linearly projecting the features to the space of dimension 128. We initialize the biases of the head with constant 0 and weights with a (semi) orthogonal matrix [43]. We include two versions of the head: with a projection to a hyperbolic space (“Hyp-”) and with projection to a unit hypersphere (“Sph-”). In the first case, we use curvature parameter  $c = 0.1$  (in Section 3.4 we investigate how it affects the method’s performance), temperature  $\tau = 0.2$  and clipping radius (defined in Section 2.4)  $r = 2.3$ . For spherical embeddings, we use temperature  $\tau = 0.1$ .

To evaluate the model performance, for the encoder, we compute the Recall@K metric for the output with distance  $D_{cos}$  (Eq. (4)); for the head, we use  $D_{cos}$  for “Sph-” version and hyperbolic distance  $D_{hyp}$  (Eq. (2)) for “Hyp-” version. We resize the test images to 224 (256 for CUB) on the smaller side and take one  $224 \times 224$  center crop. Note that some methods use images of higher resolution for training and evaluations, *e.g.*, ProxyNCA++ [52] use  $256 \times 256$  crops indicating that smaller  $227 \times 227$  crops degrade the performance by 4.3% on CUB. However,  $224 \times 224$  is the default size for encoders considered in our work; moreover, some recent methods, such as IRT<sub>R</sub> [8], use this size for experiments.

We use the AdamW optimizer [29] with a learning rate value  $1 \times 10^{-5}$  for DINO and  $3 \times 10^{-5}$  for ViT-S and DeiT-S. The weight decay value is 0.01, and the batch size equals 900. The number of optimizer steps depends on the dataset: 200 for CUB, 600 for Cars, 25000 for SOP, 2200 for In-Shop. The gradient is clipped by norm 3 for a greater stability. We apply commonly used data augmentations: random crop resizing the image to  $224 \times 224$  using bicubic interpolation combined with a random horizontal flip. We train with Automatic Mixed Precision in O2 mode<sup>1</sup>. All experiments are performed on one NVIDIA A100 GPU.

### 3.3. Results

Tab. 2 highlights the experimental results for the 128-dimensional head embedding and the results for 384-dimensional encoder embedding are shown in Tab. 3. We include evaluation of the pretrained encoders without training on the target dataset in Tab. 3 for reference. On the CUB dataset, we can observe the solid performance of methods with ViT encoder; the gap between the second-

<sup>1</sup><https://github.com/NVIDIA/apex>

best method IRT<sub>R</sub> and Hyp-ViT is 9%. However, the main improvement comes from the dataset used for pretraining (ImageNet-21k), since Hyp-DINO and Hyp-DeiT demonstrate a smaller improvement, while baseline ViT-S without finetuning shows strong performance. We hypothesize that this is due to the presence of several bird classes in the ImageNet-21k dataset encouraging the encoder to separate them during the pretraining phase.

For the SOP and In-Shop datasets, the difference between Hyp-ViT and Hyp-DINO is minor, while, for Cars-196, Hyp-DINO outperforms Hyp-ViT with a significant margin. These results confirm that both pretraining schemes are suitable for the considered task. The versions with DeiT perform worse compared to ViT- and DINO-based encoders while outperforming CNN-based models. This observation confirms the significance of vision transformers in our architecture. The experimental results suggest that hyperbolic space embeddings consistently improve the performance compared to spherical versions. Hyperbolic space seems to be beneficial for the embeddings, and the distance in hyperbolic space suits well for the pairwise cross-entropy loss function. At the same time, our sphere-based versions perform well compared to other methods with CNN encoders.

Figure 2 illustrates how learned embeddings are arranged on the Poincaré disk. We use UMAP [31] method with the “hyperboloid” distance metric to reduce the dimensionality to 2D for visualization. For the training part, we can see that samples are clustered according to labels, and each cluster is pushed closer to the border of the disk, indicating that the encoder separates classes well. However, for the testing part, the structure is more complex. We observe that some of the samples tend to move towards the center and intermix, while others stay in clusters, showing possible hierarchical relationships. We can see that car images are grouped by several properties: pose, color, shape, etc.

### 3.4. Impact of Hyperparameters

In this section, we investigate the impact of the values of the hyperparameters on the model performance.

**Encoder patch size.** ViT architecture does not process each pixel independently; for computational feasibility, the input image is sliced into patches projected into the initial embeddings. The default size of the patch is  $16 \times 16$ , although considering other values is also possible. The experiments in [3] have demonstrated a significant performance gain from smaller  $8 \times 8$  patches for self-supervised learning. In this case, the number of parameters of the encoder does not change; however, it requires processing  $4 \times$  more embeddings, which allows the encoder to learn more complex dependencies between patches. We add an experiment with this setup in Tab. 4 demonstrating a substantial performance improvement (+4.4%) compared to the default configuration. In this case, we use the same training procedure,

Method	CUB-200-2011 (K)				Cars-196 (K)				SOP (K)				In-Shop (K)			
	1	2	4	8	1	2	4	8	1	10	100	1000	1	10	20	30
Margin [62]	63.9	75.3	84.4	90.6	79.6	86.5	91.9	95.1	72.7	86.2	93.8	98.0	-	-	-	-
FastAP [2]	-	-	-	-	-	-	-	-	73.8	88.0	94.9	98.3	-	-	-	-
NSoftmax [64]	56.5	69.6	79.9	87.6	81.6	88.7	93.4	96.3	75.2	88.7	95.2	-	86.6	96.8	97.8	98.3
MIC [40]	66.1	76.8	85.6	-	82.6	89.1	93.2	-	77.2	89.4	94.6	-	88.2	97.0	-	98.0
XBM [59]	-	-	-	-	-	-	-	-	80.6	91.6	96.2	98.7	91.3	97.8	98.4	98.7
IRT <sub>R</sub> [8]	72.6	81.9	88.7	92.8	-	-	-	-	83.4	93.0	97.0	99.0	91.1	98.1	98.6	99.0
Sph-DeiT	73.3	82.4	88.7	93.0	77.3	85.4	91.1	94.4	82.5	93.1	97.3	99.2	89.3	97.0	97.9	98.4
Sph-DINO	76.0	84.7	90.3	94.1	81.9	88.7	92.8	95.8	82.0	92.3	96.9	99.1	90.4	97.3	98.1	98.5
Sph-ViT §	83.2	89.7	93.6	95.8	78.5	86.0	90.9	94.3	82.5	92.9	97.4	99.3	90.8	97.8	98.5	98.8
Hyp-DeiT	74.7	84.5	90.1	94.1	82.1	89.1	93.4	96.3	83.0	93.4	97.5	99.2	90.9	97.9	98.6	98.9
Hyp-DINO	78.3	86.0	91.2	94.7	<b>86.0</b>	<b>91.9</b>	<b>95.2</b>	<b>97.2</b>	84.6	94.1	97.7	99.3	92.6	<b>98.4</b>	<b>99.0</b>	<b>99.2</b>
Hyp-ViT §	<b>84.0</b>	<b>90.2</b>	<b>94.2</b>	<b>96.4</b>	82.7	89.7	93.9	96.2	<b>85.5</b>	<b>94.9</b>	<b>98.1</b>	<b>99.4</b>	<b>92.7</b>	<b>98.4</b>	98.9	99.1

Table 2. Recall@K metric for four datasets for 128-dimensional embeddings. The 6 versions of our method are listed in the bottom section, evaluated for head embeddings. ‘‘Sph-’’ are versions with hypersphere embeddings optimised using  $D_{cos}$  (Eq. (4)), ‘‘Hyp-’’ are versions with hyperbolic embeddings optimised using  $D_{hyp}$  (Eq. (2)). ‘‘DeiT’’, ‘‘DINO’’ and ‘‘ViT’’ indicate type of pretraining for the vision transformer encoder. Margin, FastAP, MIC, XBM, NSoftmax are based on ResNet-50 [18] encoder, IRT<sub>R</sub> is based on DeiT [53].

§ pretrained on the larger ImageNet-21k [6].

Method	Dim	CUB-200-2011 (K)				Cars-196 (K)				SOP (K)				In-Shop (K)			
		1	2	4	8	1	2	4	8	1	10	100	1000	1	10	20	30
A-BIER [36]	512	57.5	68.7	78.3	86.2	82.0	89.0	93.2	96.1	74.2	86.9	94.0	97.8	83.1	95.1	96.9	97.5
ABE [24]	512	60.6	71.5	79.8	87.4	85.2	90.5	94.0	96.1	76.3	88.4	94.8	98.2	87.3	96.7	97.9	98.2
SM [49]	512	56.0	68.3	78.2	86.3	83.4	89.9	93.9	96.5	75.3	87.5	93.7	97.4	90.7	97.8	98.5	98.8
XBM [59]	512	65.8	75.9	84.0	89.9	82.0	88.7	93.1	96.1	79.5	90.8	96.1	98.7	89.9	97.6	98.4	98.6
HTL [13]	512	57.1	68.8	78.7	86.5	81.4	88.0	92.7	95.7	74.8	88.3	94.8	98.4	80.9	94.3	95.8	97.2
MS [58]	512	65.7	77.0	86.3	91.2	84.1	90.4	94.0	96.5	78.2	90.5	96.0	98.7	89.7	97.9	98.5	98.8
SoftTriple [37]	512	65.4	76.4	84.5	90.4	84.5	90.7	94.5	96.9	78.6	86.6	91.8	95.4	-	-	-	-
HORDE [20]	512	66.8	77.4	85.1	91.0	86.2	91.9	95.1	97.2	80.1	91.3	96.2	98.7	90.4	97.8	98.4	98.7
Proxy-Anchor [23]	512	68.4	79.2	86.8	91.6	86.1	91.7	95.0	97.3	79.1	90.8	96.2	98.7	91.5	98.1	98.8	<b>99.1</b>
NSoftmax [64]	512	61.3	73.9	83.5	90.0	84.2	90.4	94.4	96.9	78.2	90.6	96.2	-	86.6	97.5	98.4	98.8
ProxyNCA++ [52]	512	69.0	79.8	87.3	92.7	86.5	92.5	95.7	97.7	80.7	92.0	96.7	98.9	90.4	98.1	98.8	99.0
IRT <sub>R</sub> [8]	384	76.6	85.0	91.1	94.3	-	-	-	-	84.2	93.7	97.3	99.1	91.9	98.1	98.7	98.9
ResNet-50 [18] †	2048	41.2	53.8	66.3	77.5	41.4	53.6	66.1	76.6	50.6	66.7	80.7	93.0	25.8	49.1	56.4	60.5
DeiT-S [53] †	384	70.6	81.3	88.7	93.5	52.8	65.1	76.2	85.3	58.3	73.9	85.9	95.4	37.9	64.7	72.1	75.9
DINO [3] †	384	70.8	81.1	88.8	93.5	42.9	53.9	64.2	74.4	63.4	78.1	88.3	96.0	46.1	71.1	77.5	81.1
ViT-S [48] † §	384	83.1	90.4	94.4	96.5	47.8	60.2	72.2	82.6	62.1	77.7	89.0	96.8	43.2	70.2	76.7	80.5
Sph-DeiT	384	76.2	84.5	90.2	94.3	81.7	88.6	93.4	96.2	82.5	92.9	97.2	99.1	89.6	97.2	98.0	98.4
Sph-DINO	384	78.7	86.7	91.4	94.9	86.6	91.8	95.2	97.4	82.2	92.1	96.8	98.9	90.1	97.1	98.0	98.4
Sph-ViT §	384	85.1	90.7	94.3	96.4	81.7	89.0	93.0	95.8	82.1	92.5	97.1	99.1	90.4	97.4	98.2	98.6
Hyp-DeiT	384	77.8	86.6	91.9	95.1	86.4	92.2	95.5	97.5	83.3	93.5	97.4	99.1	90.5	97.8	98.5	98.9
Hyp-DINO	384	80.9	87.6	92.4	95.6	<b>89.2</b>	<b>94.1</b>	<b>96.7</b>	<b>98.1</b>	85.1	94.4	97.8	99.3	92.4	<b>98.4</b>	<b>98.9</b>	<b>99.1</b>
Hyp-ViT §	384	<b>85.6</b>	<b>91.4</b>	<b>94.8</b>	<b>96.7</b>	86.5	92.1	95.3	97.3	<b>85.9</b>	<b>94.9</b>	<b>98.1</b>	<b>99.5</b>	<b>92.5</b>	98.3	98.8	<b>99.1</b>

Table 3. Recall@K metric for four datasets, ‘‘Dim’’ column shows the dimensionality of embeddings. The 6 versions of our method are listed in the bottom section, evaluated for encoder embeddings, titles are described in Table 2. Encoders by method: A-BIER, ABE, SM: GoogleNet [51]; XBM, HTL, MS, SoftTriple, HORDE, Proxy-Anchor: Inception with batch normalization [19]; NSoftmax, ProxyNCA++: ResNet-50 [18]; IRT<sub>R</sub>: DeiT [53]. † pretrained encoders without training on the target dataset. § pretrained on the larger ImageNet-21k [6].

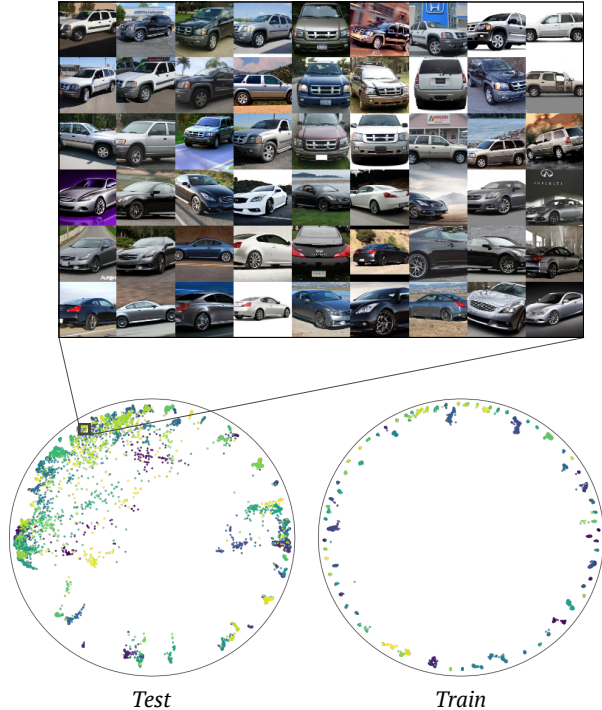


Figure 2. Hyp-DINO embeddings for Cars-196 dataset (training and evaluation sets) on the Poincaré disk. Each point inside the disk corresponds to a sample, different colors indicate different classes. Images of cars are plotted preserving neighborhood relations of samples.

Method	Dim	Recall@K			
		1	2	4	8
NSoftmax [64]	2048	89.3	94.1	96.4	98.0
ProxyNCA++ [52]	2048	90.1	94.5	97.0	98.4
Hyp-DINO 16 × 16	128	86.0	91.9	95.2	97.2
Hyp-DINO 8 × 8	128	90.4	94.7	97.0	98.2
Hyp-DINO 16 × 16	384	89.2	94.1	96.7	98.1
Hyp-DINO 8 × 8	384	92.8	96.2	97.8	98.8

Table 4. First two rows represent current best overall result for Cars-196 dataset with ResNet-50 encoder. Our method (Hyp-DINO) is presented with 8 × 8 and 16 × 16 patch sizes.

described in Section 3.2, with the batch size equal to 120.

**Manifold curvature.** Tab. 5 shows the model performance depending on the curvature value  $c$ . We observe that the method is robust in the range (0.01, 0.3) while larger values lead to degradation. Notably, the accuracy of the head degrades faster since the hyperbolic distance is also used in the evaluation and the imprecision in this parameter immediately affects the output. The radius of the ball is inversely proportional to the  $c$  value. Intuitively, if the  $c$  value tends to 0, the radius tends to infinity, making the ball as

Parameter	Encoder(384)	Head
Default	92.4	92.6
$c = 0.01$	92.3	92.6
$c = 0.05$	92.4	92.6
$c = 0.3$	92.3	92.0
$c = 0.5$	91.8	91.0
$c = 1.0$	90.0	89.2
Head dim. 16	88.6	83.3
Head dim. 32	90.2	89.6
Head dim. 64	91.6	91.7
Batch size 200	92.0	91.9
Batch size 400	92.5	92.5
Batch size 1600	92.4	92.6

Table 5. Recall@1 metric for various hyperparameters for Hyp-ViT configuration on In-Shop dataset. Default configuration is  $c = 0.1$ , head dimensionality 128, batch size 900.

flat as the Euclidean space; in contrast, larger  $c$  values correspond to a steeper configuration. Note that according to  $\delta$  values (Tab. 1), the estimated value of  $c$  is close to 0.2, depending on the dataset and encoder. However, smaller values tend to provide better stability; we believe this is due to an optimisation process that can be improved for the hyperbolic space. For this reason, we adjusted the default value towards a smaller 0.1 (Section 3.2).

**Embedding size and batch size.** As expected, lower output dimensionality leads to lower recall values. However, taking into account a high data variability (3,985 categories in the test set), the experimental results suggest that the method has a reasonable representation power even in the case of lower dimensions.

The batch size directly influences the number of negative examples during the training phase; thus, intuitively, larger values have to be more profitable for the model performance. However, as the experiments show (Tab. 5), the method is robust for batch size  $\geq 400$ , having a minor accuracy degradation for batch size equal to 200. Therefore, for considered datasets, the method does not require distributed training with a large number of GPUs [4] or specific solutions with a momentum network [17].

## 4. Related Work

**Hyperbolic embeddings.** Learning embeddings in hyperbolic spaces have emerged since this approach was proposed for NLP tasks [34, 35]. Shortly after that, hyperbolic neural networks were presented as a generalization of standard Euclidean operations allowing to learn the data representations directly in hyperbolic spaces [11]. The authors generalized standard linear layers to hyperbolic counterparts, defined multinomial logistic regression and recurrent neural networks. Several studies showed the benefits of

hyperbolic embeddings of visual data when applied to few-shot [9, 12, 22] and zero-shot learning [9, 26]. In [22], the authors proposed a hybrid architecture with the main bulk of the layers operating in Euclidean space and only final layers operating in hyperbolic space. In [9], the authors instead focus on kernelization widely used in Euclidean space and generalize them for hyperbolic representations. The paper [26] proposes a method directly incorporating the hierarchical relations for hyperbolic embeddings in application to zero-shot learning.

**Vision transformers in metric learning.** The paper [8] has recently demonstrated beneficial properties of vision transformers for category-level and object retrieval tasks. The proposed  $\text{IRT}_R$  employs the architecture and pretraining scheme of DeiT [53]. The method is trained using the contrastive loss with cross-batch memory [59] with momentum encoder [17] in several experiments. Moreover, the method requires a sophisticated entropy regularization to spread the embeddings more uniformly on a hypersphere. The study performed in [57] has shown that pairwise cross-entropy loss, considered in our work, already possesses this property. Asymptotically, this loss can be decoupled into two components: one optimizes the alignment of positive pairs, while the second preserves overall uniformity. Furthermore, the exponential expansion of the volume in the hyperbolic space can facilitate uniform feature alignment.

Self-supervised learning is similar in spirit to metric learning: in both cases, the encoder is trained to produce similar representations for semantically similar images. Consequently, there are various relevant approaches in these domains. DINO [3] is a recently proposed method, where vision transformer is trained in the self-supervised learning setting. This method shows a high k-NN classification accuracy for obtained representations while also performing well in the image retrieval task. These results suggest that both vision transformers and self-supervised pre-training are advantageous for metric learning and our experiments confirm this.

**Metric learning loss functions.** A contrastive loss [16] and its popular variation triplet margin loss [60] are classic metric learning loss functions. In the first case, the distance between positive pairs is optimized to be lower than some predefined threshold and larger for the negative pairs. The triplet margin loss penalizes the cases where negative examples are closer to each other than positives plus a margin  $m$ . Another variation of the contrastive loss is the lifted structure loss [47] with LogSumExp applied to all negative pair distances. Similarly, NCA loss [14] minimizes the distance between positives with respect to a set of negatives using exponential weighting. In essence, pairwise cross-entropy loss (Equation (5)) equals NCA when all batch samples are used as negatives.

A cross-entropy loss in the form of pairwise-distance

loss for the metric learning was introduced by [46] as N-pair loss. In addition, the paper [1] established a connection between the standard cross-entropy loss for classification and metric learning losses, proposing their own version of the pairwise cross-entropy loss. Recently, this loss function has seen overwhelming success in the self-supervised learning field [4, 17, 55]. Popular implementations are InfoNCE [55] and NT-Xent [4]. However, most of these works only consider Euclidean distances between embeddings (generally  $L_2$ -normalized). Our method extends this widely used loss function to the hyperbolic space.

## 5. Conclusion

In this paper, we have combined several improvements for the metric learning task: pairwise cross-entropy loss with the hyperbolic distance function, vision transformers with several pretraining schemes. We empirically verified that each proposed component is crucial for the best performance. In deep learning tasks, it is often tricky to empirically distinguish the source of improvement between methodological contribution and a technical solution [33]. To address this obstacle, we have presented several versions, comparing elements of our method in the equal setup. The *Hyp-DINO* version is of particular interest: it demonstrates that self-supervised learning and metric learning complement each other perfectly, resulting in a powerful metric with minimal supervision.

**Limitations.** In this work, we have considered only a vision domain with a category-level retrieval task. However, the proposed method is not limited to such applications. Hyperbolic embeddings [34] and transformers [56] were initially proposed for the natural language processing. The recently proposed method [39] shows that combining visual and language domains gives rise to learning universal representation. At the same time, papers [21, 30] demonstrate that one transformer architecture is suitable for many domains at once. Hence, combining multiple domains with a common semantic metric can be an interesting development of this work.

**Broader Impact.** Common applications in the field of metric learning are face recognition [27, 44] and person re-identification [5, 63]. Such systems can improve people’s safety and quality of life, but there are also notorious cases based on gathering personal information. Another possible risk is learned social biases. While this topic is commonly studied in the NLP field, the situation with vision transformers is more subtle and far less explored.

**Acknowledgments.** This work was supported by the EU H2020 AI4Media Project under Grant 951911, by the Analytical center under the RF Government (subsidy agreement 000000D730321P5Q0002, Grant No. 70-2021-00145 02.11.2021). We thank Google Cloud Platform (GCP) for providing computing support.



## References

- [1] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. In *European conference on computer vision*, pages 548–564. Springer, 2020. 8
- [2] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1861–1870, 2019. 6
- [3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, October 2021. 1, 4, 5, 6, 8
- [4] 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. 2, 7, 8
- [5] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 403–412, 2017. 1, 8
- [6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 4, 6
- [7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 1, 4
- [8] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021. 1, 5, 6, 8
- [9] Pengfei Fang, Mehrtash Harandi, and Lars Petersson. Kernel methods in hyperbolic spaces. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10665–10674, 2021. 8
- [10] Hervé Fournier, Anas Ismail, and Antoine Vigneron. Computing the Gromov hyperbolicity of a discrete metric space. *Information Processing Letters*, 115(6-8):576–579, 2015. 3
- [11] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *Advances in Neural Information Processing Systems 31*, pages 5345–5355, 2019. 7
- [12] Zhi Gao, Yuwei Wu, Yunde Jia, and Mehrtash Harandi. Curvature generation in curved spaces for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8691–8700, 2021. 8
- [13] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–285, 2018. 6
- [14] Jacob Goldberger, Geoffrey E Hinton, Sam Roweis, and Russ R Salakhutdinov. Neighbourhood components analysis. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005. 8
- [15] Yunhui Guo, Xudong Wang, Yubei Chen, and Stella X Yu. Free hyperbolic neural networks with limited radii. *arXiv preprint arXiv:2107.11472*, 2021. 3, 4
- [16] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006. 8
- [17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 2, 7, 8
- [18] 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 (CVPR)*, June 2016. 4, 6
- [19] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 6
- [20] Pierre Jacob, David Picard, Aymeric Histace, and Edouard Klein. Metric learning with horde: High-order regularizer for deep embeddings. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6539–6548, 2019. 6
- [21] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021. 8
- [22] 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. 3, 8
- [23] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Proxy anchor loss for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 4, 6
- [24] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 736–751, 2018. 6
- [25] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *2013 IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 4
- [26] 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. 8
- [27] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphreface: Deep hypersphere embedding

- for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017. 1, 8
- [28] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 4
- [29] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 5
- [30] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021. 8
- [31] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018. 5
- [32] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017. 1
- [33] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020. 2, 8
- [34] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30:6338–6347, 2017. 1, 7, 8
- [35] 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. 7
- [36] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE transactions on pattern analysis and machine intelligence*, 42(2):276–290, 2018. 6
- [37] Qi Qian, Lei Shang, Baigui Sun, Juhua Hu, Hao Li, and Rong Jin. Softtriple loss: Deep metric learning without triplet sampling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6450–6458, 2019. 6
- [38] Limeng Qiao, Yemin Shi, Jia Li, Yaowei Wang, Tiejun Huang, and Yonghong Tian. Transductive episodic-wise adaptive metric for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3603–3612, 2019. 1
- [39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 8
- [40] Karsten Roth, Biagio Brattoli, and Bjorn Ommer. Mic: Mining interclass characteristics for improved metric learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8000–8009, 2019. 6
- [41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 4
- [42] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pages 355–366. Springer, 2011. 1, 2
- [43] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings*, 2014. 5
- [44] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015. 1, 8
- [45] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 1
- [46] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 1, 8
- [47] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 4, 8
- [48] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021. 4, 5, 6
- [49] Yumin Suh, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. Stochastic class-based hard example mining for deep metric learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7244–7252, 2019. 6
- [50] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018. 1
- [51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 6
- [52] Eu Wern Teh, Terrance DeVries, and Graham W Taylor. Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis. In *European Conference on Computer Vision*, pages 448–464. Springer, 2020. 5, 6, 7
- [53] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training

- data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 1, 4, 6, 8
- [54] Abraham Albert Ungar. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194, 2008. 3
- [55] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807, 2018. 2, 8
- [56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 4, 8
- [57] 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. 8
- [58] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019. 6
- [59] Xun Wang, Haozhi Zhang, Weilin Huang, and Matthew R Scott. Cross-batch memory for embedding learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6388–6397, 2020. 6, 8
- [60] Kilian Q Weinberger, John Blitzer, and Lawrence Saul. Distance metric learning for large margin nearest neighbor classification. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2006. 8
- [61] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 4
- [62] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017. 6
- [63] Tong Xiao, Shuang Li, Bochao Wang, Liang Lin, and Xiaogang Wang. Joint detection and identification feature learning for person search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3415–3424, 2017. 1, 8
- [64] Andrew Zhai and Hao-Yu Wu. Classification is a strong baseline for deep metric learning. *arXiv preprint arXiv:1811.12649*, 2018. 6, 7
- [65] Stephan Zheng, Yang Song, Thomas Leung, and Ian Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4480–4488, 2016. 1