Hyperbolic Vision Transformers

Yuriy Sosnin
HSE University

Abstract

This paper evaluates the hyperbolic embeddings methodology proposed by Ermolov et al. [14] within a metric learning framework, employing the vanilla Triplet loss. While we do not achieve state-of-the-art results, we demonstrate that Hyperbolic embeddings can outperform standard Euclidean embeddings in some scenarios even with minimal adjustments to default metric learning pipeline. Additionally, we propose a fully Hyperbolic Vision Transformer operating on the hyperbolic patch space. We do not obtain training results, leaving it as future work for exploration.

1. Introduction

Representation learning, also known as feature learning, aims to automatically discover meaningful and informative representations of raw data. Traditional machine learning approaches often require manually engineered features, which can be time-consuming and domain-specific. Representation learning, on the other hand, leverages the power of deep neural networks to extract high-level features from raw data, enabling the model to learn directly from the raw input. By learning representations that capture the underlying structure and patterns in the data, representation learning has shown great success in various tasks such as image recognition, natural language processing, and recommendation systems.

Inductive biases about the data and the space in which object representations are hypothesized to lie can greatly enhance model efficiency and yield superior results. One natural property observed in many real-world datasets is the presence of hierarchy. For instance, social networks exhibit power law distribution, small diameter, and high clustering – properties naturally emerging from hyperbolic geometry [10, 23]. Natural Language domain consists of data that have tree-like structures – words form dependency trees inside sentences, and hierarchical semantic trees in the whole language [36]. ImageNet is in essence hierarchical [11].

Most Neural Representation models are by default Euclidean, and Euclidean spaces are inherently limited in capturing hierarchy. It is proven that Euclidean space can not isometrically embed a tree, even with a bounded number of dimensions. In contrast, *Hyperbolic* spaces have properties that allow them to embed trees naturally. Hyperbolic space has negative curvature, it expands exponentially with distance from the origin, and therefore are able to embed trees with low distortion [34]. These characteristics make Hyperbolic spaces an appealing inductive bias to incorporate into representation models.

The field of Hyperbolic Neural Networks has been rapidly developed in recent years. Mostly it has been focused on NLP, but it has been demonstrated that Computer Vision applications can also benefit from these advancements [20]. In our work, we follow the approach of [14]. They propose to use pre-trained Vision Transformers (ViT) [13] as Euclidean feature extractors, and project the resulting representations into Poincaré Ball model of hyperbolic space. We evaluate ViT in hyperbolic space on a default pipeline with Triplet loss, and propose (without experiments) fully hyperbolic architecture. Additionally, we discuss the limitations associated with our implementation.

The rest of the paper is organized as follows. In section 2 we briefly introduce Poincaré Ball model of hyperbolic space, and how are necessary neural network operations defined in it. Section 3 describes our empirical procedure, introduces ViT and briefly discusses Metric Learning. We then present our results. In section 5 we propose a fully hyperbolic Vision Transformer and explain why we failed to achieve results with it. Sections 6 and 7 briefly mention related literature and conclude, respectively.

2. Neural Networks in Hyperbolic Space

2.1. Hyperbolic Space

Hyperbolic space is a smooth Riemannian manifold with constant negative sectional curvature. As Hyperbolic space can not be naturally embedded into Euclidean space, several isometric models are used to represent it, among which the most popular in ML applications is Poincaré Ball. Formally, Poincaré Ball is a pair of n-dimensional open ball $\mathbb{B}_c^n = \{\mathbf{x} \in \mathbb{R}^n : c\|\mathbf{x}\| < 1\}$ where c is space curvature, and

a Riemannian metric tensor $g^{\mathbb{D}} = \lambda_c^2 g^E$ with conformal factor $\lambda_c = \frac{2}{1-c\|x\|^2}$.

A framework of gyrovector spaces allows us to define hyperbolic Möbius addition and scalar multiplication [37, 38], given by

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

Distance between two points in hyperbolic space is therefore defined as follows:

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

A Riemannian manifold is locally represented by an n-dimensional Euclidean space called Tangent space. Logarithmic map defines a mapping $\log_{\mathbf{x}}^c : \mathcal{M} \to T_{\mathbf{x}}\mathcal{M}$ from the manifold to its tangent space at point x. Respectively, Exponential map at point $\mathbf{x}$ defines and inverse mapping from Tangent space to the manifold:

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

In most applications, $\mathbf{x}$ is usually set to $\mathbf{0}$, the origin of the space, simplifying the computation.

2.2. Hyperbolic Neural Networks

Exploration of hyperbolic spaces in deep learning started from the work of Ganea et al., where they first formulated most neural network operations on Poincaré ball [15].

Matrix-vector multiplication for linear layer is performed in tangent space with Euclidean parameters: first, hyperbolic vector is logarithmically mapped to tangent space; there, matrix multiplication is performed; the result is mapped back on the manifold with exponential map and bias is added using gyrovector addition [15].

$$h^c(x) = \exp_{\mathbf{0}}^c(\mathbf{W}(\log_{\mathbf{0}}^c(x))) \oplus_c \mathbf{b} \quad (4)$$

Activation functions are calculated in tangent space as well:

$$f^c(x) = \exp_{\mathbf{0}}^c(f(\log_{\mathbf{0}}^c(x))) \quad (5)$$

Some authors hypothesized if activation functions are even needed in hyperbolic setting as matrix multiplications are already non-linear, and there is no need to avoid collapsing several "linear" layers into one.

Those formulations are sufficient for default fully-connected networks, however, to define Transformers in hyperbolic space we need more, namely, aggregation operation for self-attention and concatenation-split for making multi-head self-attention.

Self-attention layer consists of two essential parts: *relation* stage, i.e. calculation of attention weights between queries and keys, and *aggregation* of values with attention weights [39]. In standard Transformer architecture attention weight is SoftMax over scalar products between queries and keys; in hyperbolic case, natural metric of similarity is hyperbolic distance (2) [16]. Therefore, relation between query q and key k is calculated as follows:

$$f^c(\mathbf{q}, \mathbf{k}) = -D^c(\mathbf{q}, \mathbf{k}) - \beta \quad (6)$$

where D is hyperbolic distance from equation 2 and β is a parameter.

Defining aggregation (weighted addition) is more difficult as it does not have a natural close form in Poincaré ball. Some papers use Fréchet mean [27] or even perform calculation in tangent space [29]. However, as Klein model has a close-form, it is possible to redefine the operation for Poincaré ball as proposed in [35].

$$h^c(\mathbf{v}, \alpha) = \frac{1}{2} \otimes_c \left(\frac{(\sum_{i=1}^N \alpha_i)\lambda_{\mathbf{v}_i}^c \mathbf{v}_i}{\sum_{i=1}^N |\alpha_i| \left(\lambda_{\mathbf{v}_i}^c - 1\right)} \right) \quad (7)$$

where $\mathbf{v_i}$ are value vectors to be aggregated, α_i are scalar weights.

Split and concatenation are defined accordingly in [35]. With these operations we can extend self-attention to multiple heads.

2.3. Optimization

Having parameters in non-Euclidean spaces requires special optimization techniques. Gradient Descent update on Riemannian manifolds is defined the following way (H is Riemannian gradient of the loss) [5, 6]:

$$w_{t+1} = \exp_{w_t}(-\gamma_t H(z_t, w_t)) \quad (8)$$

It has been noticed that Gradient Descent on parameters in Hyperbolic space exhibits numerical instability and often requires extra 64-bit precision [30]. A parallel idea is using feature clipping each time a Euclidean vector is exponentially mapped into Poincaré Ball, i.e. restrict the norm to some max radius r [17].

3. Method

We evaluate hyperbolic embeddings methodology from [14] in metric learning framework with vanilla Triplet loss, instead of their proposed cross-entropy-like loss. The idea is to extract representations from images using a normal Euclidean model, then map those representations into a hyperbolic space, and evaluate similarity for metric learning

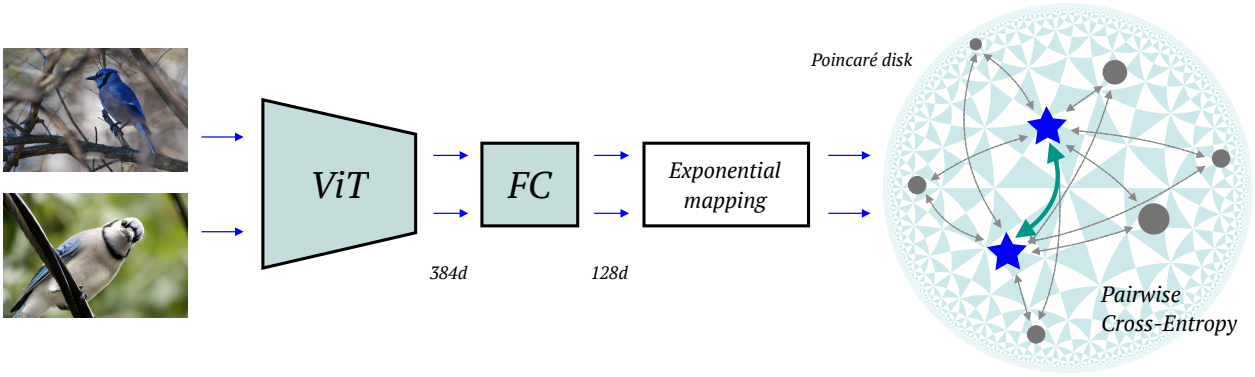

Figure 1. A pipeline for CV metric learning proposed by [14]. First, ViT extractor produces embeddings for images, then they are mapped to hyperbolic space, and distance loss calculation is performed there.

there. This approach offers several advantages: first, all model parameters are Euclidean, eliminating the need for Riemannian optimization and mitigating associated challenges; second, pre-trained model weights can be utilized. Still, in section 5 we propose a fully hyperbolic approach, where the model itself operates on points in hyperbolic space.

3.1. Vision Transformer

In this section, we introduce model architecture that we employ for our experiments. Vision Transformer (ViT) [13, 39] is an attention-base architecture extends the successful Transformer Encoder to the domain of Computer Vision. To represent image as tokens, it is split into small non-overlapping patches (16x16 or 32x32 are popular choices), and each patch is linearly projected into a n-dimensional model space. The resulting vectors are augmented with positional encoding and fed into standard BERT-like Transformer Encoder architecture [12]. To achieve the whole image representations, special [CLS] token is prepended at the bottom of the model, and its representation from the last layer is as image representation.

ViTs come in different varieties, depending on representation dimension size, number of layers and heads (base, small, large), as well as pretraining procedures. Default ViT is pretrained on classification task on ImageNet [11]. Another prominent approach, DINO, is am unsupervised contrasive learning framework that trains a model to correctly distinguish two augmented versions of one image from random other images [7, 32]. We reference them later as ViT and DINO respectively.

3.2. Metric Learning

Metric Learning is a Deep Learning approach which focuses on learning a geometrically convenient embedding space for a given domain. Unlike of classification task, striving to infer a hyperplane which would separate classes in model representation space, metric learning directly optimizes this space so that similar instances are located closer to each other than dissimilar ones, according to some distance metric. It allows to easily expand models to new unseen classes, as well as solving general learning-to-rank problems. Inference is usually performed with a simple kNN in the embedding space.

Usual similarity metric is either standard Euclidean distance or so-called cosine similarity, a dot-product of normalized embedding vectors (which is equivalent to cosine of distance on a sphere). Following [14], we instead use hyperbolic distance (2) as similarity metric between embeddings, preliminarily exponentially mapped into hyperbolic space. It is hypothesized that many real-life datasets, and especially image datasets, are hierarchical in nature, and that employing a more appropriate manifold as inductive bias would make models more efficient and yield better results [20].

While [14] use a novel cross-entropy based loss, we strive to evaluate the hyperbolic space together with a classic Triplet loss pipeline [18]. The triplet loss is designed to minimize the distance between an anchor point and a positive sample, while maximizing the distance between the anchor point and a negative sample. It considers a tuple of samples (a, p, n), where a is the anchor instance, p is the positive instance associated with the same class as the anchor, and n is the negative instance from a different class. The triplet loss is formulated as:

$$L(a, p, n) = ReLU(\alpha + D(a, p) - D(a, n)) \quad (9)$$

where $D(a, p)$ and $D(a, n)$ are the similarities between anchor-positive and anchor-negative pairs (in our case,

Model	Dim	CUB-200-2011				Stanford Cars			
		1	2	4	8	1	2	4	8
DINO-Hyperbolic	128	74.6	84.0	90.2	94.5	78.1	87.8	93.0	96.6
DINO-Sphere	128	77.9	86.7	91.7	95.1	79.5	87.4	92.2	95.6
DINO-Hyperbolic	384	78.0	86.2	92.1	95.6	79.6	88.0	93.2	94.3
DINO-Sphere	384	77.6	86.6	91.7	95.1	79.6	87.6	93.4	96.2
ViT-Hyperbolic	128	58.6	70.1	81.1	89.0	64.1	74.9	84.7	92.1
ViT-Sphere	128	67.2	79.0	86.4	91.5	63.4	75.6	85.0	91.5
ViT-Hyperbolic	384	60.3	71.8	82.9	89.7	65.4	76.2	85.8	92.5
ViT-Sphere	384	67.9	79.0	87.2	93.0	65.7	75.9	85.2	92.3

Table 1. Experimental results. Columns represent k values 1, 2, 4, 8 for Recall@k.

hyperbolic distances), and α is the margin. The margin enforces that negative samples should be at least farther away from positive samples by α distance. In standard "all triplets" scenario, all combinations of a, p, n are mined from batch and processed at each iteration.

4. Empirical Results

4.1. Data

We evaluate the performance of our models on two popular metric learning benchmark datasets.

CUB-200-2011: The CUB dataset [40] consists of 11,788 images from 200 bird species. Each class contains approximately 30 to 60 images per species. Following the standard procedure, we split the dataset in half by classes, setting classes 1-100 as training set and 101-200 as validation set.

Stanford Cars: The Stanford Cars dataset [22] comprises 16,185 images from 196 car classes, with each class containing about 50 to 80 images. Similarly, we set first 96 classes as training subset and keep the rest for validation.

4.2. implementation Details

We utilize ViT-Small as an encoder and compare two types of pretraining: standard classification (ViT) and contrasive learning (DINO). Unlike [14], we do not freeze patch embeddings. The resulting 384-dimensional representation is projected into 128, 256 and 384-dimensional space, and then mapped either onto a sphere (by normalizing them) or into hyperbolic space. In hyperbolic space, we use hyperbolic distance as discussed earlier; on a sphere, we use cosine similarity (10).

$$D_{sph}(\mathbf{x}, \mathbf{y}) = -\langle \mathbf{x}, \mathbf{y} \rangle \tag{10}$$

We use a curvature parameter $c = 0.1$, and clipping radius $r = 2.3$.

To evaluate the performance of metric learning algorithms, we report Recall@k. Recall@k represents the proportion of instances in the top k retrieved neighbors that are of the same class as the query instance. The neighbors are selected according to appripriate distance D – D_{hyp} in case of hyperbolic projection and D_{sph} in case of sphere.

We use the AdamW optimizer [26] with a learning rate of $1e{-}5$ for DINO and $3e{-}5$ for ViT-Small, with the weight decay value of 0.01. We sample 8 labels and 20 instances for each label and form every batch this way. We run training for 30 epochs on both datasets. Data augmentations include random crop resizing of the image to 224x224 using bicubic interpolation with a random horizontal flip; test images are resized to 224 on the smaller side and centered cropped to 224x224. As we are not dealing with any non-Euclidean parameters, we use default 32-bit precision.

We perform our experiments in PyTorch [33], using Open Metric Learning framework [1] for metric learning parts of our code, and geoopt [21] as geometric optimization implementation.

4.3. Results

Table 1 presents experimental results for our approach. Rows represent different model variations and columns encode values for Recall@k.

We did not manage to achieve competitive results comparable to those reported in [14]. Most likely, vanilla triplet loss, comparing at each instance only three objects, lacks the expressiveness of N-pair cross entropy loss, comparing each object with all objects of different classes from the batch at once. Moreover, our vanilla ViT results substantially underperform results obtained with the DINO backbone. We can hypothesize that contrasive pretraining yields a more favorable embedding space in context of distance-based triplet loss, and this difference is less pronounces in method of [14].

Despite this, we can still compare Spherical representation and Hyperbolic representations. As we can see, it is not clear from our results that hyperbolic space is substantially better across all cases. Hyperbolic representations achieve better results in case of CUB dataset in higher embedding dimension.

5. Fully-Hyperbolic Vision Transformer

The idea of mapping image representations into hyperbolic space is that those representations exhibit hierarchical structure [20]. However, we can hypothesize that the space of *patches* that Vision Transformer operates on, which are of course images as well, is even more hyperbolic. Say, a background depicting grass is not really discriminative of the image class, and should intuitively reside near the origin of the space. Furthermore, some features can be associated with several distinct classes, and should be located further. Finally, specific features that allow to distinguish a specific class are the outermost branches in this hierarchy.

Indeed, one of the first applications of hyperbolic Machine Learning was embedding *words*, which later prompted fully hyperbolic NLP Transformers [15, 31].

We hypothesize that including this inductive bias into a Vision Transformer architecture itself can improve the parameter efficiency and therefore achieve even better results. In other words, instead of mapping the whole image representation, we can map *patch* representations at the very beginning of the model.

As we discussed in section 2, we have all the necessary parts to implement fully hyperbolic Transformers, operating with inputs that lie in hyperbolic space. The only part lacking is Layer Normalization [3], but we decide to implement our model without it.

As discussed earlier, fully-hyperbolic models are subject to computational instability and essentially require 64-bit precision to train. We try to train our model with 64-bit precision, but Tesla V100 GPU with 80 GB RAM goes out of memory. In an attempt to fix these huge memory requirements, we derive gradients for some hyperbolic operations explicitly. However, that yields even worse results – it turns out PyTorch default autograd is more efficient.

We reduce precision to 32 bit and manage to run a training procedure from section 3. However, another important difference from those experiments that we can not use pre-trained weights anymore. Indeed, it turns out that initializing the model with pre-trained weights yields no better results than initializing with random weights. This highlights the need to employ an extensive pretraining akin to [7, 13, 32]. We decide to leave this to future work.

6. Related Literature

Recently hyperbolic Neural Networks have become a prominent stream of literature. Despite early applications focused on NLP tasks [15], Computer Vision have caught up as well. As mentioned earlier, image datasets tend to exhibit a high degree of hyperbolicity, making them a good option to achieve state of the art results [14, 20]. Formulations of convolutional operator in hyperbolic spaces have also been proposed [2, 4, 24]. [8] used distance from the origin in hyperbolic space as a zero-shot measure of uncertainty for semantic segmentation. [41] proposed to use procedure of mapping high-level representations for contrasive learning.

Despite fully hyperbolic Vision Transformers have not been considered, application of hyperbolic Transformer architecture to NLP and graph data is pretty established. Several papers have proposed to use hyperbolic distance instead of Euclidean dot product as a measure of similarity between keys and values in self-attention [16, 28, 35, 42]. Formulations of Transformers in Lorenz, Hyperboloid and Klein models, the other than Poincaré Ball models of hyperbolic space are common as well [9, 25, 31]. Finally, it has even been proposed to generate music [19].

7. Conclusion

In this paper we evaluate Hyperbolic Vision Transformers in Metric Learning setup on two Computer Vision benchmark datasets. We do not achieve state of the art results, but show that hyperbolic embeddings are capable of outperforming standard Euclidean ones in certain cases. We also propose a fully Hyperbolic Vision Transformer operating on hyperbolic patch space, but fail to train it and achieve any results. We leave it for future work.

References

[1] OML-Team/open-metric-learning, June 2023. original-date: 2022-06-04T13:12:25Z. 4

[2] Mina Ghadimi Atigh, Julian Schoep, Erman Acar, Nanne van Noord, and Pascal Mettes. Hyperbolic Image Segmentation. pages 4453–4462, 2022. 5

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer Normalization, July 2016. arXiv:1607.06450 [cs, stat]. 5

[4] Ahmad Bdeir, Kristian Schwethelm, and Niels Landwehr. Hyperbolic Geometry in Computer Vision: A Novel Framework for Convolutional Neural Networks, May 2023. arXiv:2303.15919 [cs]. 5

[5] Silvère Bonnabel. Stochastic Gradient Descent on Riemannian Manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, Sept. 2013. 2

[6] Gary Bécigneul and Octavian-Eugen Ganea. Riemannian Adaptive Optimization Methods, Feb. 2019. arXiv:1810.00760 [cs, stat] version: 2. 2

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. pages 9650–9660, 2021. 3, 5

[8] Bike Chen, Wei Peng, Xiaofeng Cao, and Juha Röning. Hyperbolic Uncertainty Aware Semantic Segmentation, Mar. 2022. arXiv:2203.08881 [cs] version: 1. 5

[9] Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully Hyperbolic Neural Networks, Mar. 2022. arXiv:2105.14686 [cs]. 5

[10] Aaron Clauset, Cristopher Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, May 2008. 1

[11] 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, June 2009. ISSN: 1063-6919. 1, 3

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. arXiv:1810.04805 [cs]. 3

[13] 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, June 2021. arXiv:2010.11929 [cs]. 1, 3, 5

[14] Aleksandr Ermolov, Leyla Mirvakhabova, Valentin Khrulkov, Nicu Sebe, and Ivan Oseledets. Hyperbolic Vision Transformers: Combining Improvements in Metric Learning. pages 7409–7419, 2022. 1, 2, 3, 4, 5

[15] Octavian Ganea, Gary Becigneul, and Thomas Hofmann. Hyperbolic Neural Networks. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 2, 5

[16] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. Hyperbolic Attention Networks, May 2018. arXiv:1805.09786 [cs]. 2, 5

[17] Yunhui Guo, Xudong Wang, Yubei Chen, and Stella X. Yu. Clipped Hyperbolic Classifiers Are Super-Hyperbolic Classifiers. pages 11–20, 2022. 2

[18] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In Defense of the Triplet Loss for Person Re-Identification, Nov. 2017. arXiv:1703.07737 [cs]. 3

[19] Wenkai Huang, Yujia Yu, Haizhou Xu, Zhiwen Su, and Yu Wu. Hyperbolic Music Transformer for Structured Music Generation. *IEEE Access*, 11:26893–26905, 2023. 5

[20] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic Image Embeddings. pages 6418–6428, 2020. 1, 3, 5

[21] Max Kochurov, Rasul Karimov, and Serge Kozlukov. Geoopt: Riemannian Optimization in PyTorch, July 2020. arXiv:2005.02819 [cs]. 4

[22] 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, Dec. 2013. 4

[23] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguñá. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, Sept. 2010. 1

[24] Keegan Lensink, Bas Peters, and Eldad Haber. Fully hyperbolic convolutional neural networks. *Research in the Mathematical Sciences*, 9(4):60, Sept. 2022. 5

[25] Zhe Liu and Yibin Xu. THG: Transformer with Hyperbolic Geometry, June 2021. arXiv:2106.07350 [cs]. 5

[26] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, Jan. 2019. arXiv:1711.05101 [cs, math]. 4

[27] Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa. Differentiating through the Fréchet Mean, July 2021. arXiv:2003.00335 [cs, stat]. 2

[28] Federico López and Michael Strube. A Fully Hyperbolic Neural Model for Hierarchical Multi-Class Classification, Oct. 2020. arXiv:2010.02053 [cs]. 5

[29] Marko Valentin Micic and Hugo Chu. Hyperbolic Deep Learning for Chinese Natural Language Understanding, Dec. 2018. arXiv:1812.10408 [cs]. 2

[30] Gal Mishne, Zhengchao Wan, Yusu Wang, and Sheng Yang. The Numerical Stability of Hyperbolic Representation Learning, June 2023. arXiv:2211.00181 [cs, math]. 2

[31] Maximillian Nickel and Douwe Kiela. Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3779–3788. PMLR, July 2018. 5

[32] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, Apr. 2023. arXiv:2304.07193 [cs]. 3, 5

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 4

[34] Rik Sarkar. Low Distortion Delaunay Embedding of Trees in Hyperbolic Plane. In Marc van Kreveld and Bettina Speckmann, editors, *Graph Drawing*, Lecture Notes in Computer Science, pages 355–366, Berlin, Heidelberg, 2012. Springer. 1

[35] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. Hyperbolic Neural Networks++, Mar. 2021. arXiv:2006.08210 [cs, stat]. 2, 5

[36] Alexandru Tifrea, Gary Bécigneul, and Octavian-Eugen Ganea. Poincaré GloVe: Hyperbolic Word Embeddings, Nov. 2018. arXiv:1810.06546 [cs]. 1

[37] Abraham Albert Ungar. *Analytic Hyperbolic Geometry and Albert Einstein's Special Theory of Relativity*. WORLD SCIENTIFIC, Feb. 2008. 2

[38] Abraham Albert Ungar. *A Gyrovector Space Approach to Hyperbolic Geometry*. Synthesis Lectures on Mathematics & Statistics. Springer International Publishing, Cham, 2009. 2

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 2, 3

[40] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset, July 2011. 4

[41] Yun Yue, Fangzhou Lin, Kazunori D. Yamada, and Ziming Zhang. Hyperbolic Contrastive Learning, Feb. 2023. arXiv:2302.01409 [cs]. 5

[42] Yiding Zhang, Xiao Wang, Chuan Shi, Xunqiang Jiang, and Yanfang Ye. Hyperbolic Graph Attention Network. *IEEE Transactions on Big Data*, 8(6):1690–1701, Dec. 2022. 5