FREE HYPERBOLIC NEURAL NETWORKS WITH LIMITED RADII

Anonymous authors
Paper under double-blind review

ABSTRACT

Non-Euclidean geometry with constant negative curvature, i.e., hyperbolic space, has attracted sustained attention in the community of machine learning. Hyperbolic space, owing to its ability to embed hierarchical structures continuously with low distortion, has been applied for learning data with tree-like structures. Hyperbolic Neural Networks (HNNs) that operate directly in hyperbolic space have also been proposed recently to further exploit the potential of hyperbolic representations. While HNNs have achieved better performance than Euclidean neural networks (ENNs) on datasets with implicit hierarchical structure, they still perform poorly on standard classification benchmarks such as CIFAR and ImageNet. The traditional wisdom is that it is critical for the data to respect the hyperbolic geometry when applying HNNs. In this paper, we first conduct an empirical study showing that the inferior performance of HNNs on standard recognition datasets can be attributed to the notorious *vanishing gradient problem*. We further discovered that this problem stems from the hybrid architecture of HNNs. Our analysis leads to a simple yet effective solution called *Feature Clipping*, which regularizes the hyperbolic embedding whenever its norm exceeding a given threshold. Our thorough experiments show that the proposed method can successfully avoid the vanishing gradient problem when training HNNs with backpropagation. The improved HNNs are able to achieve comparable performance with ENNs on standard image recognition datasets including MNIST, CIFAR10, CIFAR100 and ImageNet, while demonstrating more adversarial robustness and stronger out-of-distribution detection capability.

1 INTRODUCTION

Many public datasets exhibit hierarchical structures. For instance, the conceptual relations in WordNet (Miller, 1995) form a hierarchical structure, users in social networks such as Facebook or twitter form hierarchies based on different occupations and organizations (Gupte et al., 2011). Representing such hierarchical data in Euclidean space cannot capture and reflect their semantic or functional resemblance (Alanis-Lobato et al., 2016; Nickel & Kiela, 2017). Hyperbolic space, i.e., non-Euclidean space with constant negative curvature, has been leveraged to embed data with hierarchical structures with low distortion owing to the nature of exponential growth in volume with respect to its radius (Nickel & Kiela, 2017; Sarkar, 2011; Sala et al., 2018). For instance, hyperbolic space has been used for analyzing the hierarchical structure in single cell data (Klimovskaia et al., 2020), learning hierarchical word embedding (Nickel & Kiela, 2017), embedding complex networks (Alanis-Lobato et al., 2016), etc.

Recently, algorithms which operate directly on hyperbolic representations have also been derived to further exploit the potential of hyperbolic representations. For example, in Weber et al. (2020) the authors proposed *hyperbolic perceptron* (Weber et al., 2020) to perform perceptron algorithm directly on hyperbolic representations. Hyperbolic Support Vector Machine (Cho et al., 2019) was also proposed to perform large margin classification in hyperbolic space. Hyperbolic Neural Networks (HNNs) (Ganea et al., 2018) are proposed as an alternative to Euclidean neural networks (ENNs) to further exploit hyperbolic space and can be used for more complex problems. When HNNs are applied to image datasets, they employ a hybrid architecture (Khrulkov et al., 2020) as shown in Figure 1: an ENN is first used for extracting features from images, then the Euclidean embeddings are projected onto hyperbolic space as hyperbolic embeddings, finally the hyperbolic embeddings are

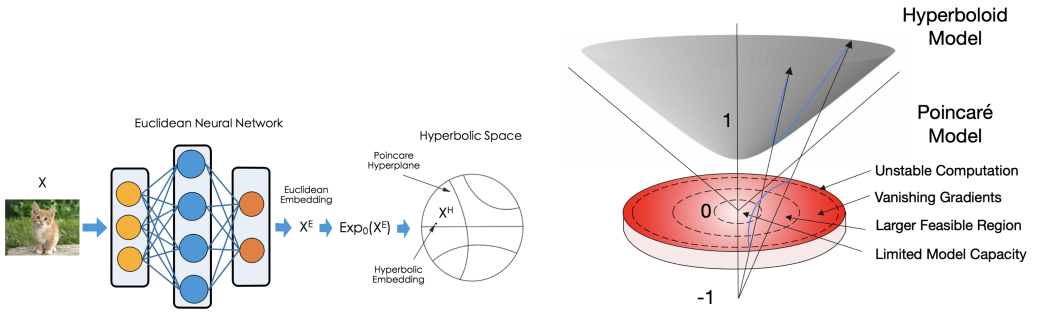


Figure 1: **Left:** HNNs employ a hybrid architecture. The Euclidean part converts an input into Euclidean embedding. Then the Euclidean embedding is projected onto the Poincaré model of hyperbolic space via exponential map $\text{Exp}_0(\cdot)$. Finally, the hyperbolic embeddings are classified with Poincaré hyperplanes. **Right:** Poincaré model can be derived using stereoscopic projection of the hyperboloid model. The distance grows exponentially fast as we move towards the boundary of the Poincaré ball. We identify that the Poincaré model can be partitioned into areas with unstable computation, vanishing gradients, larger feasible region and limited model capacity.

classified by a hyperbolic multiclass logistic regression (Ganea et al., 2018). While HNNs are able to achieve improvements over ENNs on several datasets with explicit hierarchical structure (Ganea et al., 2018), they perform poorly on standard image classification benchmarks. This, undesirably, causes severe limitations when applying HNNs. In Khrulkov et al. (2020), while the authors show that several image datasets possess latent hierarchical structure, there are no experimental results showing that HNNs can capture such structure or provide similar performance to ENNs. Existing improvements on HNNs mainly focus on reducing the number of parameters (Shimizu et al., 2020) or incorporating different types of neural network layers such as attention (Gulcehre et al., 2018) or convolution (Shimizu et al., 2020). Unfortunately, the reason behind the inferior performance of HNNs compared with ENNs on standard image datasets is not investigated or understood.

Our key insight is that the inferiority of HNNs on standard image datasets is not their intrinsic limitation but stems from the improper training procedures. We first conduct an empirical study showing that the hybrid nature of HNNs leads to *vanishing gradient problem* during training using backpropagation. In particular, the training dynamics of HNNs push the hyperbolic embeddings to the boundary of the Poincaré ball (Anderson, 2006) which causes the gradients of Euclidean parameters to vanish. Inspired by the above analysis, we propose a simple yet effective remedy to this problem, called *Feature Clipping*, by constraining the norm of the hyperbolic embedding during training. With the proposed technique, HNNs are on par with ENNs on several standard image classification benchmarks including MNIST, CIFAR10, CIFAR100 and ImageNet. This shows that HNNs can not only perform better than ENNs on hierarchical datasets but also achieve comparable performance to ENNs on standard image datasets. This undoubtedly broadens the application of HNNs for computer vision tasks. The improved HNNs are also more robust to ENNs and exhibit stronger out-of-distribution detection ability.

The contributions of the paper are as follows. (1) We conduct a detailed analysis to understand the underlying issues when applying HNNs on the standard image datasets. In Figure 1, we show that the Poincaré model of hyperbolic space can be partitioned into multiple non-overlapping areas with drastically different properties which can greatly affect the embedding quality. (2) We propose a simple yet effective solution to address the *vanishing gradient problem* during training HNNs by constraining the norm of the hyperbolic embedding. (3) We conduct extensive experiments on standard image datasets including MNIST, CIFAR10, CIFAR100 and ImageNet. The results show that by addressing the problem of vanishing gradients, the performance of HNNs on standard datasets has been greatly improved and matches ENNs. Meanwhile, the improved HNNs are also more robust to adversarial attacks and exhibits stronger out-of-distribution detection capability than their Euclidean counterparts.

2 Related Work

Supervised Learning In the seminal work of Hyperbolic Neural Networks (HNNs) (Ganea et al., 2018), the authors proposed different hyperbolic neural network layers including multinomial log-

itstic regression (MLR), fully connected layers and Recurrent Neural Networks which can operate directly on hyperbolic embeddings. The proposed HNNs outperform the Euclidean variants on text entailment and noisy-prefix prediction task. Recently, Hyperbolic Neural Networks++ (Shimizu et al., 2020) was proposed to reduce the number of parameters of HNNs and also introduced hyperbolic convolutional layers. In Gulcehre et al. (2018), the authors proposed Hyperbolic attention networks (Gulcehre et al., 2018) by rewriting the operations in the attention layers using gyrovector operations (Ungar, 2005) which leads to improvements on neural machine translation, learning on graphs and visual question answering. Hyperbolic graph neural network (Liu et al., 2019) was proposed by extending the representational geometry of Graph Neural Networks (GNNs) (Zhou et al., 2020) to hyperbolic space. Hyperbolic graph attention network (Zhang et al., 2019) further studied GNNs with attention mechanism in hyperbolic space. Recently, HNNs have been used for tasks such as few-shot classification and person re-identification (Khrulkov et al., 2020).

Unsupervised Learning Unsupervised learning methods based on variants of HNNs have also attracted a lot of attention. In Nagano et al. (2019), the authors proposed a wrapped normal distribution on hyperbolic space to construct hyperbolic variational autoencoders (VAEs) (Kingma & Welling, 2013). In a concurrent work (Mathieu et al., 2019), the authors proposed Gaussian generalizations on hyperbolic space to construct Poincaré VAEs. Recent work (Hsu et al., 2020) applied hyperbolic neural networks for unsupervised 3D segmentation based on complex volumetric data.

Compared with the above-mentioned methods which focus on the application of HNNs in data with natural tree structure, this paper attempts to extend the application of HNNs to standard image recognition datasets and improves the performance of HNNs on these datasets to the level of Euclidean counterparts, greatly enhancing the universality of HNNs.

3 Free Hyperbolic Neural Networks with Limited Radii

Our goal is to address the vanishing gradient problem when training HNNs. We propose an efficient solution to solve the problem and the improved HNNs are on par with ENNs on standard recognition datasets and show better performance in terms of few-shot learning, adversarial robustness and out-of-distribution detection. First, we review the basics of HNNs. Then, we analyze the vanishing gradient problem in training HNNs. Finally, we present the proposed method and show its effectiveness of addressing the issue of HNNs.

Riemannian Geometry An n-dimensional topological manifold $\mathcal{M}$ is a topological space that is locally Euclidean of dimension n: every point $\mathbf{x} \in \mathcal{M}$ has a neighborhood that is homeomorphic to an open subset of $\mathbb{R}^n$. A smooth manifold is a topological manifold with additional smooth structure which is a maximal smooth atlas. A Riemannian manifold $(\mathcal{M}, \mathfrak{g})$ is a real smooth manifold with a Riemannian metric $\mathfrak{g}$. The Riemannian metric $\mathfrak{g}$ is defined on the tangent space $T_{\mathbf{x}}\mathcal{M}$ of $\mathcal{M}$ which is a smoothly varying inner product. For $\mathbf{x} \in \mathcal{M}$ and any two vectors $\mathbf{v}, \mathbf{w} \in T_{\mathbf{x}}\mathcal{M}$, the inner product $\langle \mathbf{v}, \mathbf{w} \rangle_{\mathbf{x}}$ is defined as $\mathfrak{g}(\mathbf{v}, \mathbf{w})$. With the definition of inner product, for $\mathbf{v} \in T_{\mathbf{x}}\mathcal{M}$, the norm is defined as $\|\mathbf{v}\|_{\mathbf{x}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{\mathbf{x}}}$. A geodesic is a curve $\gamma : [0, 1] \to \mathcal{M}$ of unit speed that is locally minimizing the distance between two points on the manifold. Given $\mathbf{x}, \mathbf{y} \in \mathcal{M}, \mathbf{v} \in T_{\mathbf{x}}\mathcal{M}$, and a geodesic γ of length $\|\mathbf{v}\|$ such that $\gamma(0) = \mathbf{x}, \gamma(1) = \mathbf{y}, \gamma'(0) = \mathbf{v}/\|\mathbf{v}\|$, the exponential map $\text{Exp}_{\mathbf{x}} : T_{\mathbf{x}}\mathcal{M} \to \mathcal{M}$ satisfies $\text{Exp}_{\mathbf{x}}(\mathbf{v}) = \mathbf{y}$ and the inverse exponential map $\text{Exp}_{\mathbf{x}}^{-1} : \mathcal{M} \to T_{\mathbf{x}}\mathcal{M}$ satisfies $\text{Exp}_{\mathbf{x}}^{-1}(\mathbf{y}) = \mathbf{v}$. For more details please refer to Carmo (1992); Lee (2018)

Poincaré Ball Model for Hyperbolic Space A hyperbolic space is a Riemannian manifold with constant negative curvature. There are several isometric models for hyperbolic space, one of the commonly used models is Poincaré ball model (Nickel & Kiela, 2017; Ganea et al., 2018) which can be derived using stereoscopic projection of the hyperboloid model (Anderson, 2006). The n-dimensional Poincaré ball model of constant negative curvature $-c$ is defined as $(\mathbb{B}_c^n, \mathfrak{g}^c)$, where $\mathbb{B}_c^n = \{\mathbf{x} \in \mathbb{R}^m : c\|\mathbf{x}\| < 1\}$ and $\mathfrak{g}^c = (\gamma_{\mathbf{x}}^c)^2 I_n$ is the Riemannian metric tensor. $\gamma_{\mathbf{x}}^c = \frac{2}{1-c\|\mathbf{x}\|^2}$ is the conformal factor and I_n is the Euclidean metric tensor. The conformal factor induces the inner product $\langle \mathbf{u}, \mathbf{v} \rangle_{\mathbf{x}}^c = (\gamma_{\mathbf{x}}^c)^2 \langle \mathbf{u}, \mathbf{v} \rangle$ and norm $\|\mathbf{v}\|_{\mathbf{x}}^c = \gamma_{\mathbf{x}}^c \|v\|$ for all $\mathbf{u}, \mathbf{v} \in T_{\mathbf{x}}\mathbb{B}_c^n$. The exponential map of Poincaré ball model can be written analytically with the operations of gyrovector space which will be introduced in Section 3.

Gyrovector Space A gyrovector space (Ungar, 2005; 2008) is an algebraic structure that provides an analytic way to operate in hyperbolic space. Each point in hyperbolic space is endowed with vector-like properties similar to the point in Euclidean space.

The basic operation in gyrovector space is called Möbius addition $\oplus_c$. With Möbius addition $\oplus_c$, we can define vector addition of two points in Poincaré ball model as,

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

for all $\mathbf{u}, \mathbf{v} \in \mathbb{B}_c^n$. Particularly, $\lim_{c \to 0} \oplus_c$ converges to the standard $+$ in the Euclidean space. Similarly, we can define various operations such as scalar multiplication, subtraction, exponential map, inverse exponential map in Poincaré ball model with the operations of gyrovector space. Those operations form the basis for constructing hyperbolic neural network layers as shown in (Ganea et al., 2018). For more details, please refer to Appendix A.1.

Hyperbolic Neural Networks In Ganea et al. (2018), the authors derived different hyperbolic neural network layers based on the algebra of gyrovector space. When applying hyperbolic neural networks to image datasets (Khrulkov et al., 2020), they consist of an Euclidean sub-network and a hyperbolic classifier as shown in Figure 1. The Euclidean sub-network $E(\mathbf{x})$ converts an input $\mathbf{x}$ such as an image into a representation $\mathbf{x}^E$ in Euclidean space. $\mathbf{x}^E$ is then projected onto hyperbolic space $\mathbb{B}_c^n$ via an exponential map $\text{Exp}_{\mathbf{0}}^c(\cdot)$ as $\mathbf{x}^H \in \mathbb{B}_c^n$. The hyperbolic classifier $H(\mathbf{x}^H)$ performs classification based on $\mathbf{x}^H$ with the standard cross-entropy loss ℓ.

Let the parameters of the Euclidean sub-network be $\mathbf{w}^E$ and the parameters of the hyperbolic classifier be $\mathbf{w}^H$. Given the loss function ℓ, the optimization problem can be formalized as,

$$\min_{\mathbf{w}^E, \mathbf{w}^H} \ell(H(\text{Exp}_{\mathbf{0}}^c((E(\mathbf{x}; \mathbf{w}^E)); \mathbf{w}^H), y) \tag{2}$$

where the outer and inner functions are $H : \mathbb{B}_c^m \to \mathbb{R}$ and $E : \mathbb{R}^n \to \mathbb{R}^m$. As shown in (Ganea et al., 2018), the exponential map is defined as,

$$\text{Exp}_{\mathbf{0}}^c(\mathbf{v}) = \tanh(\sqrt{c}\|\mathbf{v}\|)\frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|} \tag{3}$$

The construction of hyperbolic classifier relies on the following definition of Poincaré hyperplanes,

Definition 3.1 (Poincaré hyperplanes (Ganea et al., 2018)) *For $\mathbf{p} \in \mathbb{B}_c^n$, $\mathbf{a} \in T_{\mathbf{p}}\mathbb{B}_c^n \setminus \{\mathbf{0}\}$, the Poincaré hyperplane is defined as,*

$$\tilde{H}_{a,p}^c \coloneqq \{\mathbf{x} \in \mathbb{B}_c^n : \langle -\mathbf{p} \oplus_c \mathbf{x}, \mathbf{a} \rangle = 0\} \tag{4}$$

where $\mathbf{a}$ is the normal vector and $\langle \mathbf{a}, \mathbf{p} \rangle$ defines the bias of the Poincaré hyperplane.

As shown in Ganea et al. (2018), in hyperbolic space the probability that a given $\mathbf{x} \in \mathbb{B}_c^n$ is classified as class k is,

$$p(y = k|\mathbf{x}) \propto \exp(\text{sign}(\langle -\mathbf{p}_k \oplus_c \mathbf{x}, \mathbf{a}_k \rangle))\sqrt{\mathfrak{g}_{\mathbf{p_k}}^c(\mathbf{a}_k, \mathbf{a}_k)}d_c(\mathbf{x}, \tilde{H}_{\mathbf{a}_k, \mathbf{p}}^c) \tag{5}$$

where $d_c(\mathbf{x}, \tilde{H}_{\mathbf{a}_k, \mathbf{p}}^c)$ is the distance of the embedding $\mathbf{x}$ to the Poincaré hyperplane of class k. In hyperbolic classifier the parameters are the vectors $\{\mathbf{p}_k\}$ for each class k.

Training Hyperbolic Neural Networks with Backpropagation The standard backpropagation algorithm (Rumelhart et al., 1986) is used for training HNNs (Ganea et al., 2018; Khrulkov et al., 2020). During backpropagation, the gradient of the Euclidean parameters $\mathbf{w}^E$ can be computed as,

$$\frac{\partial \ell}{\partial \mathbf{w}^E} = (\frac{\partial \mathbf{x}^H}{\partial \mathbf{w}^E})^T \frac{\partial \ell}{\partial \mathbf{x}^H} \tag{6}$$

where $\mathbf{x}^H$ is the hyperbolic embedding of the input $\mathbf{x}$, $\frac{\partial \mathbf{x}^H}{\partial \mathbf{w}^E}$ is the Jacobian matrix and $\frac{\partial \ell}{\partial \mathbf{x}^H}$ is the gradient of the loss function with respect to the hyperbolic embedding $\mathbf{x}^H$. It is noteworthy that since $\mathbf{x}^H$ is an embedding in hyperbolic space, $\frac{\partial \ell}{\partial \mathbf{x}^H} \in T_{\mathbf{x}^H}\mathbb{B}_c^n$ is the Riemannian gradient (Bonnabel, 2013) and

$$\frac{\partial \ell}{\partial \mathbf{x}^H} = \frac{(1 - \|\mathbf{x}^H\|^2)^2}{4}\nabla \ell(\mathbf{x}^H) \tag{7}$$

where $\nabla \ell(\mathbf{x}^H)$ is Euclidean gradient and $\frac{(1-\|\mathbf{x}^H\|^2)^2}{4}$ is the inverse of the Riemannian metric tensor.

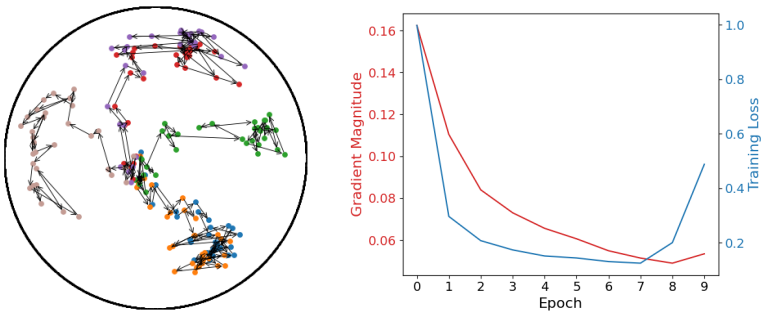


Figure 2: Hyperbolic neural networks suffer from vanishing gradient problem during training with backpropagation. **Left**: The trajectories of the hyperbolic embeddings of six randomly sampled inputs during training in a 2-dimensional Poincaré ball. The arrows indicate the change of location of each embedding with each gradient update. The embeddings move to the boundary of the ball during optimization which causes vanishing gradient problem. **Right**: The gradient vanishes while the training loss goes up at the end of training.

Vanishing Gradient Problem At initialization, all the hyperbolic embeddings of the inputs locate in the center of the Poincaré ball. From Equation 5, we can see that in order to maximize the probability of the correct prediction, we need to increase the distance of the hyperbolic embedding to the corresponding Poincaré hyperplane, i.e., $d_c(\mathbf{x}^H, \tilde{H}^c_{\mathbf{a}_k,\mathbf{p}})$. The training dynamics of HNNs thus push the hyperbolic embeddings to the boundary of the Poincaré ball in which case $\|\mathbf{x}^H\|^2$ approaches one. The inverse of the Riemannian metric tensor becomes zero which causes $\|\frac{\partial \ell}{\partial \mathbf{x}^H}\|^2$ to be small. From Equation 6, it is easy to see that if $\|\frac{\partial \ell}{\partial \mathbf{x}^H}\|^2$ is small, then $\|\frac{\partial \ell}{\partial \mathbf{w}^E}\|^2$ is small and the optimization makes no progress on $\mathbf{w}^E$.

To obtain further understanding, we conduct an experiment to show the vanishing gradient problem during training hyperbolic neural networks. We train a LeNet-like convolutional neural network (LeCun et al., 1998) with hyperbolic classifier on the MNIST data. We use a two-dimensional Poincaré ball for visualization. In Figure 2, we show the trajectories of the hyperbolic embeddings of six randomly sampled inputs during training. The arrows indicate the movement of each embedding after one gradient update step. It can be observed that at initialization all the hyperbolic embeddings are close to the center of the Poincaré ball. During training, the hyperbolic embeddings gradually move to the boundary of the ball. The magnitude of the gradient diminishes during training as the training loss decays. However, at the end of training, while the training loss slightly increases, the gradient vanishes due to the issue that the hyperbolic embeddings approach the boundary of the ball. *Vanishing gradient problem* (Hochreiter, 1998; Pennington et al., 2017; 2018; Hanin, 2018) is one of the difficulties in training deep neural networks using backpropagation. Vanishing gradient problem occurs when the magnitude of the gradient is too small for the optimization to make progress. For the standard Euclidean neural networks, vanishing gradient problem can be alleviated by architecture designs (Hochreiter & Schmidhuber, 1997; He et al., 2016), proper weight initialization (Mishkin & Matas, 2015) and carefully chosen activation functions (Xu et al., 2015a). However, the vanishing gradient problem in training HNNs is not exploited in existing literature.

The Effect of Gradient Update of Euclidean Parameters on the Hyperbolic Embedding We derive the effect of a single gradient update of the Euclidean parameters on the hyperbolic embedding, for more details please refer to Appendix A.2. For the Euclidean sub-network $E : \mathbb{R}^n \to \mathbb{R}^m$, consider the first-order Taylor-expansion with a single gradient update,

$$\begin{aligned} E(\mathbf{w}^E_{t+1}) &= E(\mathbf{w}^E_t + \eta \frac{\partial \ell}{\partial \mathbf{w}^E}) \\ &\approx E(\mathbf{w}^E_t) + \eta (\frac{\partial E(\mathbf{w}^E_t)}{\partial \mathbf{w}^E_t})^T \frac{\partial \ell}{\partial \mathbf{w}^E} \end{aligned} \tag{8}$$

where η is the learning rate. The gradient of the exponential map can be computed as,

$$\begin{aligned} \nabla \exp^c_{\mathbf{0}}(\mathbf{v}) &= \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|} \nabla \tanh(\sqrt{c}\|\mathbf{v}\|) + \tanh(\sqrt{c}\|\mathbf{v}\|) \nabla \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|} \\ &= (1 - \tanh(\sqrt{c}\|\mathbf{v}\|)^2 + \tanh(\sqrt{c}\|\mathbf{v}\|) \frac{1}{\sqrt{c}} \frac{2}{\|\mathbf{v}\|} \end{aligned} \tag{9}$$

Let $\mathbf{x}_{t+1}^H$ be the projected point in hyperbolic space, i.e,

$$\mathbf{x}_{t+1}^H = \exp_{\mathbf{0}}^c(E(\mathbf{w}_{t+1}^E)) \tag{10}$$

By applying the first-order Taylor-expansion on the exponential map and following standard derivations, we can find that,

$$\begin{aligned} \mathbf{x}_{t+1}^H &= \mathbf{x}_t^H + C(E(\mathbf{w}_t^E)^T \frac{\partial \ell}{\partial \mathbf{w}^E} \\ &= \mathbf{x}_t^H + C(E(\mathbf{w}_t^E)^T \frac{(1 - \|\mathbf{x}_t^H\|^2)^2}{4} \frac{\partial \ell}{\partial \mathbf{w}^E} \end{aligned} \tag{11}$$

where $C(E(\mathbf{w}_t^E)) = (\nabla \exp_{\mathbf{0}}^c(E(\mathbf{w}_t^E))^T \eta(\frac{\partial E(\mathbf{w}_t^E)}{\partial \mathbf{w}_t^E})^T$. Thus once $\|\mathbf{x}_t^H\|$ approaches one, the hyperbolic embedding stagnates no matter how large the training loss is.

Feature clipping for training hyperbolic neural networks There are several possible solutions to address the vanishing gradient problem for training HNN. One tentative solution is to replace all the Euclidean layers with hyperbolic layers, however it is not clear how to directly map the original input images onto hyperbolic space. Another solution is to use normalized gradient descent (Hazan et al., 2015) for optimizing the Euclidean parameters to reduce the effect of gradient magnitude. However we observed that this introduces instability during training and makes it harder to tune the learning rate for optimizing Euclidean parameters.

We address the vanishing gradient problem by first reformulating the optimization problem in Equation 2 with a regularization term which controls the magnitude of hyperbolic embeddings,

$$\min_{\mathbf{w}^E, \mathbf{w}^H} \ell(H(\mathbf{x}^H; \mathbf{w}^H), y) + \beta \|\mathbf{x}^H\|^2 \tag{12}$$

where $\mathbf{x}^H = \text{Exp}_{\mathbf{0}}^c((E(\mathbf{x}; \mathbf{w}^E)$ and $\beta > 0$ is a hyperparameter. By minimizing the training loss, the hyperbolic embeddings tend to move to the boundary of the Poincaré ball which causes the vanishing gradient problem. The additional regularization term is used to prevent the hyperbolic embeddings from approaching the boundary.

While the soft constraint introduced in Equation 12 is effective, it introduces additional complexity to the optimization process as shown in Appendix A.9. We instead employ the following hard constraint which regularizes the Euclidean embedding before the exponential map whenever its norm exceeding a given threshold,

$$C(\mathbf{x}^E; r) = \min\{1, \frac{r}{\|\mathbf{x}^E\|}\} \cdot \mathbf{x}^E \tag{13}$$

where $\mathbf{x}^E = E(\mathbf{x}; \mathbf{w}^E)$ and $r > 0$ is a hyperparameter. Using the relation between the hyperbolic distance and Euclidean distance,

$$d_c(0, r) = s \ln(\frac{s + r}{s - r}) \tag{14}$$

where $s = 1/\sqrt{|c|}$ and c is the curvature, r can be converted into the effective radius $d_c(0, r)$ of the Poincaré ball.

The proposed *Feature Clipping* imposes a hard constraint on the maximum norm of the hyperbolic embedding to prevent the inverse of the Riemannian metric tensor from approaching zero. Therefore there is always a gradient signal for optimizing the hyperbolic embedding. Although decreasing the norm of the hyperbolic embedding shrinks the effective radius of the embedding space, we found that it does no harm to accuracy while alleviating the vanishing gradient problem.

A radius limited hyperbolic classifier is a super-hyperbolic classifier, not a nearly Euclidean classifier. In Appendix A.8, we show that hyperbolic space with *Feature Clipping* well maintains the hyperbolic property and delivers better results for learning hierarchical word embeddings.

In Liu et al. (2019); Nickel & Kiela (2018), a similar clipping strategy (with a much larger clipping value) is used to overcome the numerical issue when training hyperbolic graph neural networks and learning word embeddings with the hyperboloid model. We need to point out that our work is focused on the hyperbolic neural network for supervised image classification and its unique vanishing gradient issue, which is drastically different from Liu et al. (2019); Nickel & Kiela (2018) in terms of model architecture and the focused problem.

4 Experimental Settings and Evaluation Protocol

We conduct extensive experiments on standard recognition datasets to show the effectiveness of the proposed *feature clipping* for HNNs. The results show that HNNs with feature clipping are on par

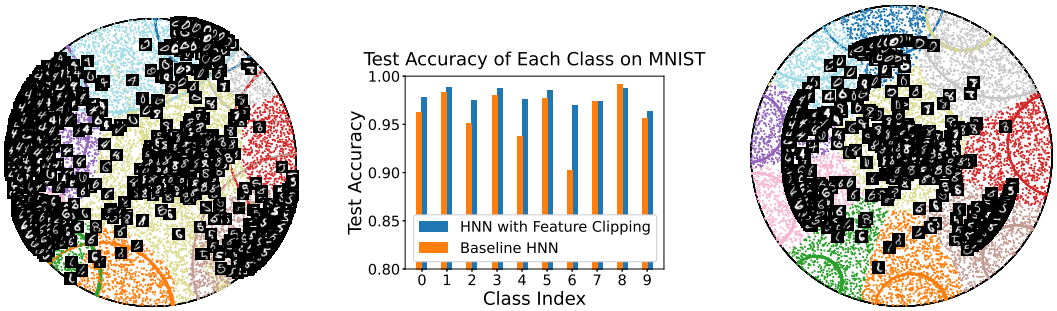


Figure 3: HNNs with feature clipping learn more discriminative feature in hyperbolic space. The per class accuracy in the center figure indicates that the baseline HNNs learn biased feature space which hurts the performance of certain classes. **Left**: the Poincaré decision hyperplanes and the hyperbolic embeddings of sampled test images of baseline HNNs. **Center**: The per class test accuracy of baseline HNNs and HNNs with feature clipping. **Right**: the Poincaré decision hyperplanes and the hyperbolic embeddings of sampled test images of HNNs with feature clipping.

with ENNs on standard recognition datasets while demonstrating better performance in terms of few-shot classification, adversarial robustness and out-of-distribution detection.

Datasets We conduct experiments on various commonly used image classification datasets: MNIST (LeCun), CIFAR10 (Krizhevsky et al., 2009), CIFAR100 (Krizhevsky et al., 2009) and ImageNet (Deng et al., 2009). The details of these datasets can be found in Appendix A.3. To our best knowledge, this paper is the first attempt to extensively evaluate hyperbolic neural networks on the standard image classification datasets for supervised classification.

Baselines and Networks We compare the performance of HNNs training with/without the proposed feature clipping method (Ganea et al., 2018; Khrulkov et al., 2020) and their Euclidean counterparts. For MNIST, we use a LeNet-like convolutional neural network (LeCun et al., 1998) which has two convolutional layers with max pooling layers in between and three fully connected layers. For CIFAR10 and CIFAR100, we use WideResNet (Zagoruyko & Komodakis, 2016). For ImageNet, we use a standard ResNet18 (He et al., 2016).

Training Setups For training ENNs, we use SGD with momentum. For training HNNs, the Euclidean parameters of HNNs are trained using SGD, and the hyperbolic parameters of HNNs are optimized using stochastic Riemann gradient descent (Bonnabel, 2013), just like the previous method. For training networks on MNIST, we train the network for 10 epochs with a learning rate of 0.1. The batch size is 64. For training networks on CIFAR10 and CIFAR100, we train the network for 100 epochs with an initial learning rate of 0.1 and use cosine learning rate scheduler (Loshchilov & Hutter, 2016). The batch size is 128. For training networks on ImageNet, we train the network for 100 epochs with an initial learning rate of 0.1 and the learning rate decays by 10 every 30 epochs. The batch size is 256. We find the HNNs are robust to the choice of the hyperparameter r, thus we fix r to be 1.0 in all the experiments. For more discussions and results on the effect of r, please see Appendix A.4. For the baseline HNNs, we use a clipping value of 15 similar to Liu et al. (2019); Nickel & Kiela (2018) to address the numerical issue. The experiments on MNIST, CIFAR10 and CIFAR100 are repeated for 5 times and we report both average accuracy and standard deviation. All the experiments are done on 8 NVIDIA TITAN RTX GPUs.

Results on the standard benchmarks In Table 1, we show the results of different networks on the considered benchmarks. On MNIST, we can observe that the accuracy of the improved HNNs with feature clipping is about 5% higher than the baseline HNNs and match the performance of Euclidean neural networks. On CIFAR10, CIFAR100 and ImageNet, the improved HNNs achieve 6%, 3% and 3% improvement over baseline HNNs. The results show that HNNs can perform well even on datasets which lack explicit hierarchical structure.

In Figure 3 we show the Poincaré hyperplanes of all the classes and the hyperbolic embeddings of 1000 sampled test images extracted by the baseline HNNs and HNNs with feature clipping. Note that the Poincaré hyperplanes consist of arcs of Euclidean circles that are orthogonal to the boundary of the ball. We also color the points in the ball based on the classification results. It can be observed that by regularizing the magnitude of the hyperbolic embedding, all the embeddings locate in a

Methods	MNIST	CIFAR10	CIFAR100	ImageNet
Euclidean (He et al., 2016)	99.12 ± 0.34 %	94.81 ± 0.42%	76.24 ± 0.35%	69.82%
Hyperbolic (Ganea et al., 2018)	94.42 ± 0.29 %	88.82 ± 0.51%	72.26 ± 0.41%	65.74%
Hyperbolic w/ Feature Clipping	99.08 ± 0.31 %	94.76 ± 0.44%	75.88 ± 0.38%	68.45%
Δ	↑ 4.66%	↑ 5.94%	↑ 3.62%	↑ 2.79%

Table 1: By incorporating the proposed method, the performance gap between Hyperbolic Neural Network (HNN) and Euclidean Neural Network (ENN) can be greatly closed on all experimented benchmarks. **Top-1 accuracies on standard image classification datasets** are compared here. Top-1 accuracy gains to the vanilla HNNs (Ganea et al., 2018; Khrulkov et al., 2020) are shown in the last row.

restricted region of the whole Poincaré ball and the network learns more regular and discriminative feature in hyperbolic space.

Methods	Embedding Net	1-Shot 5-Way	5-Shot 5-Way
ProtoNet (Snell et al., 2017)	4 Conv	51.31 ± 0.91%	70.77 ± 0.69%
Hyperbolic ProtoNet (Khrulkov et al., 2020)	4 Conv	61.18 ± 0.24%	79.51 ± 0.16%
Hyperbolic ProtoNet w/ Feature Clipping	4 Conv	**64.66 ± 0.24%**	**81.76 ± 0.15%**

Table 2: Hyperbolic embeddings provide a better alternative to Euclidean embeddings on few-shot learning task, and further improvements can be obtained through the proposed feature clipping method. Here are comparisons of **few-shot classification results on fine-grained CUB dataset** on 1-shot 5-way and 5-shot 5-way tasks. All accuracies are reported with 95% confidence intervals.

Methods	Embedding Net	1-Shot 5-Way	5-Shot 5-Way
ProtoNet (Snell et al., 2017)	4 Conv	49.42 ± 0.78%	68.20 ± 0.66%
Hyperbolic ProtoNet (Khrulkov et al., 2020)	4 Conv	51.88 ± 0.20%	**72.63 ± 0.16%**
Hyperbolic ProtoNet w/ Feature Clipping	4 Conv	**53.01 ± 0.22%**	**72.66 ± 0.15%**

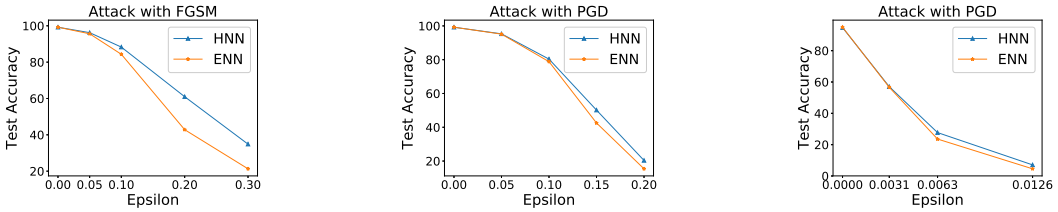
Table 3: **Few-shot classification results on miniImageNet** on 1-shot 5-way and 5-shot 5-way tasks. All accuracies are reported with 95% confidence intervals.

Few-shot Learning We show that the proposed feature clipping can also improve the performance of Hyperbolic ProtoNet (Khrulkov et al., 2020) for few-shot learning. Different from the standard ProtoNet (Snell et al., 2017) which computes the prototype of each class in Euclidean space, Hyperbolic ProtoNet computes the class prototype in hyperbolic space using hyperbolic averaging. Hyperbolic geometry has been shown to learn more accurate embeddings than Euclidean geometry for few-shot learning (Khrulkov et al., 2020).

We follow the experimental settings in Khrulkov et al. (2020) and conduct experiments on CUB dataset (Welinder et al., 2010) and miniImageNet dataset (Russakovsky et al., 2015). We consider 1-shot 5-way and 5-shot 5-way tasks as in (Khrulkov et al., 2020). The evaluation is repeated for 10000 times and we report the average performance and the 95% confidence interval. Table 2 and Table 3 show that the proposed feature clipping further improves the accuracy of Hyperbolic ProtoNet for few-shot classification by as much as 3%.

Adversarial Robustness We show that HNNs are more robust to adversarial attacks including FGSM (Goodfellow et al., 2014) and PGD (Madry et al., 2017) than ENNs. We train the networks regularly without adversarial training with the setups described in Section 4. For attacking networks trained on MNIST using FGSM, we consider the perturbation $\epsilon = 0.05, 0.1, 0.2, 0.3$. For attacking networks trained on MNIST using PGD, we consider the perturbation $\epsilon = 0.05, 0.1, 0.15, 0.2$. The number of steps is 40. For attacking networks trained on CIFAR10 using PGD, we consider the perturbation $\epsilon = 0.8/255, 1.6/255, 3.2/255$. The number of steps is 7. From Figure 4 we can see that across all the cases hyperbolic neural networks show more robustness than ENNs to adversarial attacks. More results and discussions can be found in Appendix A.5.

Out-of-distribution Detection We conduct experiments to show that HNNs have stronger out-of-distribution detection capability than ENNs. Out-of-distribution detection aims at determining



(a) On MNIST (b) On MNIST (c) On CIFAR10

Figure 4: **Adversarial robustness** of hyperbolic neural networks (HNNs) and Euclidean neural networks (ENNs) to different attack methods and perturbations.

whether or not an given input is from the same distribution as the training data. We follow the experimental settings in Liu et al. (2020). The in-distribution datasets are CIFAR10 and CIFAR100. The out-of-distribution datasets are ISUN (Xu et al., 2015b), Place365 (Zhou et al., 2017), Texture (Cimpoi et al., 2014), SVHN (Netzer et al., 2011), LSUN-Crop (Yu et al., 2015) and LSUN-Resize (Yu et al., 2015). We use the same network and training setups as described in Section 4 for training models on CIFAR10 and CIFAR100. For detecting out-of-distribution data, we use both softmax score and energy score as described in Liu et al. (2020). For metrics, we consider FPR95, AUROC and AUPR (Liu et al., 2020). In Table 4 and Table 5 we show the results of using softmax score on CIFAR10 and CIFAR100 respectively. We can see that HNNs and ENNs achieve similar AUPR, however HNNs achieve much better performance in terms of FPR95 and AUROC. In particular, HNNs reduce FPR95 by 5.82% and 9.55% on CIFAR10 and CIFAR100 respectively. For results using energy score, please see Appendix A.6

Table 4: **The results of out-of-distribution detection** on CIFAR10 with ***softmax score***

Network / OOD Dataset	Euclidean Neural Network			Hyperbolic Neural Network		
	FPR95 ↓	**AUROC** ↑	**AUPR** ↑	**FPR95** ↓	**AUROC** ↑	**AUPR** ↑
ISUN	46.30 ± 0.78	91.50 ± 0.16	98.16 ± 0.05	45.28 ±0.65	91.61 ± 0.21	98.09 ± 0.06
Place365	51.09 ±0.92	87.56 ±0.37	96.76± 0.15	54.77 ±0.76	86.82 ± 0.41	96.17 ±0.20
Texture	65.04± 0.91	82.80 ± 0.35	94.59 ±0.20	47.12± 0.62	89.91± 0.20	97.39± 0.09
SVHN	71.66 ±0.84	86.58± 0.21	97.06± 0.06	49.89± 1.03	91.34± 0.22	98.13± 0.06
LSUN-Crop	22.22± 0.78	96.05± 0.10	99.16 ±0.03	23.87 ± 0.73	95.65± 0.22	98.98 ±0.07
LSUN-Resize	41.06 ±1.07	92.67 ±0.16	98.42 ±0.04	41.49 ±1.24	92.97± 0.24	98.46 ±0.07
Mean	49.56	89.53	97.36	**43.74**	**91.38**	**97.87**

Table 5: **The results of out-of-distribution detection** on CIFAR100 with ***softmax score***

Network / OOD Dataset	Euclidean Neural Network			Hyperbolic Neural Network		
	FPR95 ↓	**AUROC** ↑	**AUPR** ↑	**FPR95** ↓	**AUROC** ↑	**AUPR** ↑
ISUN	74.07 ± 0.87	82.51 ± 0.39	95.83 ± 0.11	68.37 ± 0.90	81.31 ± 0.43	94.96 ± 0.20
Place365	81.01 ± 1.07	76.90 ± 0.45	94.02 ± 0.15	79.66 ± 0.69	76.94 ± 0.28	93.91 ± 0.18
Texture	83.67 ± 0.68	77.52 ± 0.32	94.47 ± 0.10	64.91 ± 0.80	83.26 ± 0.25	95.77 ± 0.08
SVHN	84.56 ± 0.78	84.32 ± 0.22	96.69 ± 0.07	53.11 ± 1.04	89.53 ± 0.26	97.71 ± 0.07
LSUN-Crop	43.46 ± 0.79	93.09 ± 0.23	98.58 ± 0.05	51.08 ± 1.17	87.21 ± 0.39	96.83 ± 0.13
LSUN-Resize	71.50 ± 0.73	82.12 ± 0.40	95.69 ± 0.13	63.86 ± 1.10	82.36 ± 0.42	95.16 ± 0.13
Mean	73.05	82.74	**95.88**	**63.50**	**83.43**	95.72

5 Conclusion

We address one important issue when training HNNs which is ignored in previous literature. We identify the vanishing gradient problem when training hyperbolic neural networks and propose a simple yet effective solution which does not need to modify the current optimizer or architecture. We conduct extensive experiments on commonly used image dataset benchmarks including MNIST, CIFAR10, CIFAR100 and ImageNet. Hyperbolic neural networks with feature clipping show significant improvement over baseline HNNs and match the performance of ENNs. The proposed method also improves the performance of hyperbolic neural networks for few-shot learning. Further studies reveal that hyperbolic neural networks are more robust to adversarial attacks and have stronger out-of-distribution detection capability.

REPRODUCIBILITY STATEMENT

We have listed all used parameters and training details in Section 4 for ease of reproducibility. Our code is based on several publically available github repositories: `https://github.com/leymir/hyperbolic-image-embeddings`, `https://github.com/dalab/hyperbolic_nn`, `https://github.com/facebookresearch/poincare-embeddings` and `https://github.com/wetliu/energy_ood`). We will make it publically available during review process and afterwards.

REFERENCES

Gregorio Alanis-Lobato, Pablo Mier, and Miguel A Andrade-Navarro. Efficient embedding of complex networks to hyperbolic space via their laplacian. *Scientific reports*, 6(1):1–10, 2016.

James W Anderson. *Hyperbolic geometry*. Springer Science & Business Media, 2006.

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

Manfredo Perdigao do Carmo. *Riemannian geometry*. Birkhäuser, 1992.

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*, pp. 1832–1840. PMLR, 2019.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613, 2014.

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*, pp. 248–255. Ieee, 2009.

Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *arXiv preprint arXiv:1805.09112*, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. Hyperbolic attention networks. *arXiv preprint arXiv:1805.09786*, 2018.

Mangesh Gupte, Pravin Shankar, Jing Li, Shanmugauelayut Muthukrishnan, and Liviu Iftode. Finding hierarchy in directed online social networks. In *Proceedings of the 20th international conference on World wide web*, pp. 557–566, 2011.

Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *arXiv preprint arXiv:1801.03744*, 2018.

Elad Hazan, Kfir Y Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *arXiv preprint arXiv:1507.02030*, 2015.

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*, pp. 770–778, 2016.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Joy Hsu, Jeffrey Gu, Gong-Her Wu, Wah Chiu, and Serena Yeung. Learning hyperbolic representations for unsupervised 3d segmentation. *arXiv preprint arXiv:2012.01644*, 2020.

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*, pp. 6418–6428, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

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

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

John M Lee. *Introduction to Riemannian manifolds*. Springer, 2018.

Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *arXiv preprint arXiv:1910.12892*, 2019.

Weitang Liu, Xiaoyun Wang, John D Owens, and Yixuan Li. Energy-based out-of-distribution detection. *arXiv preprint arXiv:2010.03759*, 2020.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with poincar\'e variational auto-encoders. *arXiv preprint arXiv:1901.06033*, 2019.

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

Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

Yoshihiro Nagano, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama. A wrapped normal distribution on hyperbolic space for gradient-based learning. In *International Conference on Machine Learning*, pp. 4693–4702. PMLR, 2019.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Maximilian Nickel and Douwe Kiela. Poincar\'e embeddings for learning hierarchical representations. *arXiv preprint arXiv:1705.08039*, 2017.

Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pp. 3779–3788. PMLR, 2018.

Jeffrey Pennington, Samuel S Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *arXiv preprint arXiv:1711.04735*, 2017.

Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pp. 1924–1932. PMLR, 2018.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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

Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pp. 355–366. Springer, 2011.

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

Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

Abraham A Ungar. Hyperbolic trigonometry and its application in the poincaré ball model of hyperbolic geometry. *Computers & Mathematics with Applications*, 41(1-2):135–147, 2001.

Abraham A Ungar. *Analytic hyperbolic geometry: Mathematical foundations and applications*. World Scientific, 2005.

Abraham Albert Ungar. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194, 2008.

Melanie Weber, Manzil Zaheer, Ankit Singh Rawat, Aditya Menon, and Sanjiv Kumar. Robust large-margin learning in hyperbolic space. *arXiv preprint arXiv:2004.05465*, 2020.

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.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015a.

Pingmei Xu, Krista A Ehinger, Yinda Zhang, Adam Finkelstein, Sanjeev R Kulkarni, and Jianxiong Xiao. Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*, 2015b.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Yiding Zhang, Xiao Wang, Xunqiang Jiang, Chuan Shi, and Yanfang Ye. Hyperbolic graph attention network. *arXiv preprint arXiv:1912.03046*, 2019.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

A APPENDIX

A.1 GYROVECTOR SPACE

We give more details on gyrovector space, for a more systematic treatment, please refer to (Ungar, 2005; 2008; 2001).

Gyrovector space provides a way to operate in hyperbolic space with vector algebra. Gyrovector space to hyperbolic geometry is similar to standard vector space to Euclidean geometry. The geometric objects in gyrovector space are called gyroevectors which are equivalent classes of directed gyrosegments. Similar to the vectors in Euclidean space which are added according to parallelogram law, gyrovectors are added according to gyroarallelogram law. Technically, gyrovector spaces are gyrocommutative gyrogroups of gyrovectors that admit scalar multiplications.

We start from the introduction of gyrogroups which give rise to gyrovector spaces.

Definition A.1 (Gyrogroups) *A groupoid $(G, \oplus)$ is a gyrogroup if it satisfies the follow axioms,*

1. *There exist one element $0 \in G$ satisfies $0 \oplus a = a$ for all $a \in G$.*
2. *For each $a \in G$, there exist an element $\ominus a \in G$ which satisfies $\ominus a \oplus a = 0$*
3. *For every $a, b, c \in G$, there exist a unique element $gry[a, b]c \in G$ such that $\oplus$ satisfies the left gyroassociative law $a \oplus (b \oplus c) = (a \oplus b) \oplus gry[a, b]c$.*
4. *The map $gry[a, b]c$: $G \to G$ given by $c \mapsto gry[a, b]c$ is an automorphism of the groupoid $(G, \oplus)$: $gyr[a, b] \in Aut(G, \oplus)$. The automorphism $gyr[a, b]$ of G is called the gyroautomorphism of G generated by $a, b \in G$.*
5. *The operation gry: $G \times G \to Aut(G, \oplus)$ is called gyrator of G. The gyroautomorphism $gyr[a, b]$ generated by any $a, b \in G$ has the left loop property: $gyr[a, b]$ = $gyr[a \oplus b, b]$.*

In particular, Möbius complex disk groupoid $(\mathbb{D}, \oplus_M)$ is a gyrocommunicative gyrogroup, where $\mathbb{D} = \{z \in \mathbb{C} : |z| < 1\}$ and $\oplus_M$ is the Möbius addition. The same applies to the s-ball $\mathbb{V}_s$ which is defined as,

$$\mathbb{V}_s = \{\mathbf{v} \in \mathbb{V} : \|\mathbf{v}\| < s\} \tag{15}$$

Gyrocommutative gyrogroups which admit scalar multiplication $\oplus$ become gyrovector space $(G, \oplus, \otimes)$. Möbius gyrogroups $(\mathbb{V}, \oplus_M)$ admit scalar multiplication $\oplus_M$ become Möbius gyrovector space $(\mathbb{V}, \oplus_M, \otimes_M)$.

Definition A.2 (Möbius Scalar Multiplication) *Let $(\mathbb{V}_s, \oplus_M)$ be a Möbius gyrogroup, the Möbius scalar multiplication $\otimes_M$ is defined as,*

$$r \otimes_M \mathbf{v} = s \frac{(1 + \frac{\|\mathbf{v}\|}{s})^r - (1 - \frac{\|\mathbf{v}\|}{s})^r}{(1 + \frac{\|\mathbf{v}\|}{s})^r + (1 - \frac{\|\mathbf{v}\|}{s})^r} \frac{\mathbf{v}}{\|\mathbf{v}\|} \tag{16}$$

where $r \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{V}_s$, $\mathbf{v} \neq \mathbf{0}$.

Definition A.3 (Gyrolines) *Let $\mathbf{a}, \mathbf{b}$ be two distinct points in the gyrovector space $(G, \oplus, \otimes)$. The gyroline in G which passes through $\mathbf{a}, \mathbf{b}$ is the set of points:*

$$L = \mathbf{a} \oplus (\ominus \mathbf{a} \oplus \mathbf{b}) \otimes t \tag{17}$$

where $t \in \mathbb{R}$.

It can be proven that gyrolines in a Möbius gyrovector space coincide with the geodesics of the Poincaré ball model of hyperbolic geometry.

With the aid of operations in gyrovector spaces, we can define important properties of the Poincaré ball model in closed-form expressions.

Definition A.4 (Exponential Map and Logarithmic Map) *As shown in (Ganea et al., 2018), the exponential map $\exp_{\mathbf{x}}^c : T_{\mathbf{x}}\mathbb{B}_c^n \to \mathbb{B}_c^n$ is defined as,*

$$\exp_{\mathbf{x}}^c(\mathbf{v}) = \mathbf{x} \oplus_c (\tanh(\frac{\sqrt{c}\lambda_{\mathbf{x}}^c \|\mathbf{v}\|}{2}) \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|}), \qquad \forall \mathbf{x} \in \mathbb{B}_c^n, \mathbf{v} \in T_{\mathbf{x}}\mathbb{B}_c^n. \tag{18}$$

The logarithmic map $\log_{\mathbf{x}}^c : \mathbb{B}_c^n \to T_{\mathbf{x}}\mathbb{B}_c^n$ is defined as,

$$\log_{\mathbf{x}}^c = \frac{2}{\sqrt{c}\lambda_{\mathbf{x}}^c} \tanh^{-1}(\sqrt{c}\|\ominus_c \mathbf{x} \oplus_c \mathbf{y}\|) \frac{\ominus_c \mathbf{x} \oplus_c \mathbf{y}}{\|\ominus_c \mathbf{x} \oplus_c \mathbf{y}\|}, \qquad \mathbf{x}, \mathbf{y} \in \mathbb{B}_c^n \tag{19}$$

The distance between two points in the Poincaré ball can be defined as,

Definition A.5 *(Poincaré Distance between Two Points)*

$$d_c(\mathbf{x}, \mathbf{y}) = \frac{2}{\sqrt{c}} \tanh^{-1}(\sqrt{c}\|\ominus_c \mathbf{x} \oplus_c \mathbf{y}\|) \tag{20}$$

A.2 The Effect of Gradient Update of Euclidean Parameters on the Hyperbolic Embedding

We derive the effect of the a single gradient update of the Euclidean parameters on the hyperbolic embedding. For the Euclidean sub-network $E : \mathbb{R}^n \to \mathbb{R}^m$. Consider the first-order Taylor-expansion of the Euclidean network with a single gradient update,

$$\begin{aligned} E(\mathbf{w}_{t+1}^E) &= E(\mathbf{w}_t^E + \eta \frac{\partial \ell}{\partial \mathbf{w}^E}) \\ &\approx E(\mathbf{w}_t^E) + \eta (\frac{\partial E(\mathbf{w}_t^E)}{\partial \mathbf{w}_t^E})^T \frac{\partial \ell}{\partial \mathbf{w}^E} \end{aligned} \tag{21}$$

Meanwhile, the exponentional map of the Poincaré ball is,

$$\exp_{\mathbf{0}}^c(\mathbf{v}) = \tanh(\sqrt{c}\|\mathbf{v}\|) \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|} \tag{22}$$

The gradient of the exponential map can be computed as,

$$\begin{aligned} \nabla \exp_{\mathbf{0}}^c(\mathbf{v}) &= \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|} \nabla \tanh(\sqrt{c}\|\mathbf{v}\|) + \tanh(\sqrt{c}\|\mathbf{v}\|) \nabla \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|} \\ &= (1 - \tanh(\sqrt{c}\|\mathbf{v}\|)^2 + \tanh(\sqrt{c}\|\mathbf{v}\|) \frac{1}{\sqrt{c}} \frac{2}{\|\mathbf{v}\|} \end{aligned} \tag{23}$$

Let $\mathbf{x}_{t+1}^H$ be the projected point in hyperbolic space, i.e,

$$\mathbf{x}_{t+1}^H = \exp_{\mathbf{0}}^c(E(\mathbf{w}_{t+1}^E)) \tag{24}$$

Again we can apply the first-order Taylor-expansion on the exponential map,

$$\begin{aligned} \mathbf{x}_{t+1}^H &= \exp_0^c(E(\mathbf{w}_{t+1}^E)) \\ &\approx \exp_0^c(E(\mathbf{w}_t^E) + \eta (\frac{\partial E(\mathbf{w}_t^E)}{\partial \mathbf{w}_t^E})^T \frac{\partial \ell}{\partial \mathbf{w}^E}) \end{aligned} \tag{25}$$

Denote $\eta \frac{\partial E(\mathbf{w}_t^E)}{\partial \mathbf{w}_t^E})^T \frac{\partial \ell}{\partial \mathbf{w}^E}$ by $J_{\mathbf{w}_t^E}$, we have

$$\begin{aligned} \mathbf{x}_{t+1}^H &= \exp_0^c(E(\mathbf{w}_{t+1}^E)) \\ &\approx \exp_0^c(E(\mathbf{w}_t^E) + J_{\mathbf{w}_t^E}) \\ &\approx \exp_0^c(E(\mathbf{w}_t^E)) + \frac{\partial \exp_0^c(E(\mathbf{w}_t^E)}{\partial E(\mathbf{w}_t^E)}^T J_{\mathbf{w}_t^E} \\ &= \mathbf{x}_t^H + (\frac{\partial \exp_0^c(E(\mathbf{w}_t^E)}{\partial E(\mathbf{w}_t^E)})^T J_{\mathbf{w}_t^E} \end{aligned} \tag{26}$$

Denote $(\frac{\partial \exp_{\mathbf{0}}^c(E(\mathbf{w}_t^E)}{\partial E(\mathbf{w}_t^E)})^T \eta (\frac{\partial E(\mathbf{w}_t^E)}{\partial \mathbf{w}_t^E})^T$ by $C(E(\mathbf{w}_t^E))$,

$$\begin{aligned} \mathbf{x}_{t+1}^H &= \mathbf{x}_t^H + C(E(\mathbf{w}_t^E)^T \frac{\partial \ell}{\partial \mathbf{w}^E} \\ &= \mathbf{x}_t^H + C(E(\mathbf{w}_t^E)^T \frac{(1 - \|\mathbf{x}_t^H\|^2)^2}{4} \frac{\partial \ell}{\partial \mathbf{w}^E} \end{aligned} \tag{27}$$

A.3 DATASETS

	MNIST	**CIFAR10**	**CIFAR100**	**ImageNet**
# of Training Examples	60,000	50,000	50,000	1,281,167
# of Test Examples	10,000	10,000	10,000	50,000

Table 6: The statistics of the datasets.

A.4 THE EFFECT OF HYPERPARAMETER R

We conduct ablation studies to show the effect of the hyperparameter r which is the maximum norm of the Euclidean embedding. In Figure 5 we show the change of test accuracy as we vary the hyperparameter r on MNIST, CIFAR10 and CIFAR100. We repeat the experiments for each choice of r five times and report both average accuracy and standard deviation. On the one hand, it can be observed that a larger r leads to a drop in test accuracy. As we point out, this is caused by the vanishing gradient problem in training hyperbolic neural networks. On the other hand, a small r can also lead to a drop in test accuracy especially on more complex tasks such as CIFAR10 and CIFAR100. The plausible reason is that a small r reduces the capacity of the embedding space which is detrimental for learning discriminative features.

To conclude, there is a sweet spot in terms of choosing r which is neither too large (causing vanishing gradient problem) nor too small (not enough capacity). The performance of hyperbolic neural network is also robust to the choice of the hyperparameter r if it is around the sweet spot. In Figure 1, it can be observed that hyperbolic space can be partitioned into multiple non-overlapping areas with drastically different embedding quality.

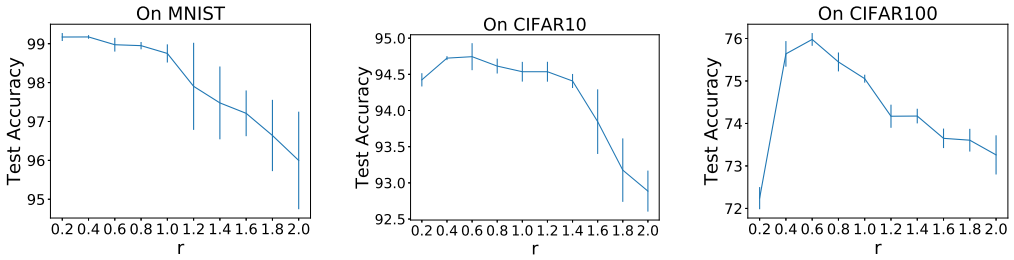


Figure 5: We show the change of the test accuracy as we vary the hyperparameter r. A large r leads to *vanishing gradient problem* and a small r causes insufficient capacity. Both lead to a drop in test accuracy.

A.5 MORE RESULTS ON ADVERSARIAL ROBUSTNESS

Although we observe that with adversarial training, hyperbolic neural networks achieve similar robust accuracy to Euclidean neural networks, in a further study, we consider training models using a small ϵ but attacking with a larger ϵ with FGSM on MNIST. In Table 7 we show the results of training the networks using $\epsilon = 0.05$ and attacking with ϵ = 0.1, 0.2 and 0.3. We can observe that for attacking with larger ϵ such as 0.2 and 0.3, hyperbolic neural networks show more robustness to Euclidean neural networks. The possible explanation is that the proposed feature clipping reduces the adversarial noises in the forward pass. One of the future directions is to systematically understand and analyze the reason behind the robustness of hyperbolic neural networks. In Figure 6, we show the clean and adversarial images generated by FGSM with hyperbolic neural networks and Euclidean neural networks respectively. The predictions of the networks are shown above the image. It can be observed that hyperbolic neural networks show more adversarial robustness compared with Euclidean neural networks.

Network \ Perturbation	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.3$
Euclidean Network	94.51%	67.85%	42.18%
Hyperbolic Network	93.58%	74.97%	46.27%

Table 7: Adversarial training with FGSM ($\epsilon = 0.05$) on MNIST.

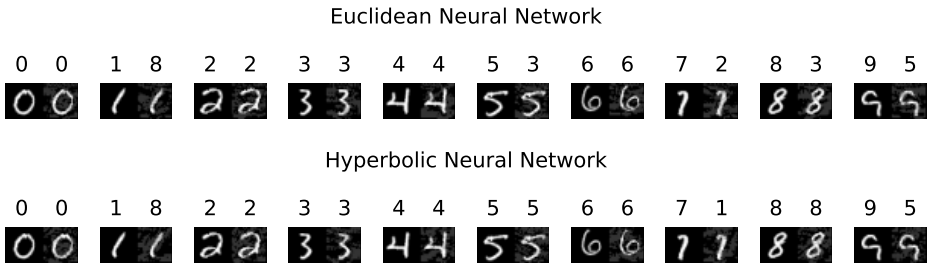


Figure 6: Hyperbolic neural networks show more adversarial robustness compared with Euclidean neural networks. We show the clean image and the corresponding adversarial image and the predictions of the network of 10 randomly sampled images. In several cases, hyperbolic neural networks make correct predictions on the adversarial images while Euclidean neural networks make wrong predictions.

A.6 More results on out-of-distribution detection

In Table 8 and 9 we show the results of using energy score (Liu et al., 2020) on CIFAR10 and CIFAR100 for out-of-distribution detection. Similar to the case of using softmax score, we can observe that on both datasets hyperbolic neural networks achieve similar performance in terms of one metric and perform better in terms other two metrics compared with Euclidean neural networks.

Table 8: **The results of out-of-distribution detection** on CIFAR10 with ***energy score***

OOD Dataset \ Network	Euclidean Neural Network			Hyperbolic Neural Network		
	FPR95 ↓	**AUROC** ↑	**AUPR** ↑	**FPR95** ↓	**AUROC** ↑	**AUPR** ↑
ISUN	34.19 ± 0.97	93.07 ± 0.24	98.42 ± 0.07	25.39 ± 0.32	95.48 ± 0.09	99.01 ± 0.04
Place365	43.34 ± 1.22	88.50 ± 0.48	96.76 ± 0.17	45.17 ± 1.19	89.61 ± 0.28	97.20 ± 0.14
Texture	58.51 ± 0.77	82.98 ± 0.20	94.55 ± 0.14	49.70 ± 0.94	90.66 ± 0.20	97.98 ± 0.04
SVHN	49.04 ± 1.05	91.57 ± 0.13	98.12 ± 0.05	57.33 ± 1.34	88.45 ± 0.20	97.44 ± 0.06
LSUN-Crop	9.48 ± 0.60	98.21 ± 0.07	99.63 ± 0.02	24.78 ± 0.73	95.06 ± 0.15	98.92 ± 0.05
LSUN-Resize	28.28 ± 0.66	94.31 ± 0.14	98.72 ± 0.04	22.52 ± 0.67	96.15 ± 0.09	99.18 ± 0.02
Mean	**37.14**	91.44	97.70	37.48	**92.57**	**98.29**

Table 9: **The results of out-of-distribution detection** on CIFAR100 with ***energy score***

OOD Dataset \ Network	Euclidean Neural Network			Hyperbolic Neural Network		
	FPR95 ↓	**AUROC** ↑	**AUPR** ↑	**FPR95** ↓	**AUROC** ↑	**AUPR** ↑
ISUN	74.49 ± 0.60	82.45 ± 0.33	95.84 ± 0.12	68.75 ± 0.93	81.33 ± 0.31	94.93 ± 0.16
Place365	81.20 ± 0.86	77.02 ± 0.34	94.13 ± 0.13	79.51 ± 0.69	77.23 ± 0.37	93.97 ± 0.17
Texture	83.19 ± 0.31	77.74 ± 0.35	94.54 ± 0.11	65.03 ± 0.52	83.38 ± 0.29	95.85 ± 0.10
SVHN	84.12 ± 0.59	84.41 ± 0.16	96.72 ± 0.04	55.44 ± 1.00	89.43 ± 0.25	97.69 ± 0.06
LSUN-Crop	43.80 ± 1.29	93.04 ± 0.22	98.56 ± 0.05	74.89 ± 0.73	84.98 ± 0.18	96.46 ± 0.08
LSUN-Resize	71.86 ± 0.69	81.86 ± 0.27	95.60 ± 0.09	64.35 ± 0.62	82.64 ± 0.36	95.27 ± 0.14
Mean	73.11	82.75	**95.90**	**67.99**	**83.17**	95.70

A.7 Softmax with temperature scaling

We consider softmax with temperature scaling as an alternative for addressing the vanishing gradient problem in training hyperbolic neural networks. Softmax with temperature scaling introduces

an additional temperature parameter T to adjust the logits before applying the softmax function. Softmax with temperature scaling can be formulated as,

$$\text{Softmax}(\boldsymbol{Z}/T)_i = \frac{e^{Z_i/T}}{\sum_{j=1}^{K} e^{Z_j/T}} \qquad \text{for} \quad i = 1, ..., K \quad \text{and} \quad \boldsymbol{Z} = (Z_1, ..., Z_K) \tag{28}$$

In hyperbolic neural networks, $\boldsymbol{Z}$ is the output of the hyperbolic fully-connected layer and K is the number of classes. If the additional temperature parameter T is smaller than 1, the magnitude (in the Euclidean sense) of the hyperbolic embedding will be scaled up which prevents it from approaching the boundary of the ball.

In Figure 7, we show the performance of training hyperbolic neural networks with temperature scaling compared with the proposed feature clipping. We consider feature dimensions of 2 and 64 respectively. Different temperature parameters are considered and the experiments are repeated for 10 times with different random seeds. We show both the average accuracy and the standard deviation. We can observe that softmax with temperature scaling and a carefully tuned temperature parameter can approach the performance of the proposed feature clipping when the feature dimension is 2. However, the feature dimension is 64, softmax with temperature scaling severely underperforms the proposed feature clipping. The results again confirm the effectiveness of the proposed approach.

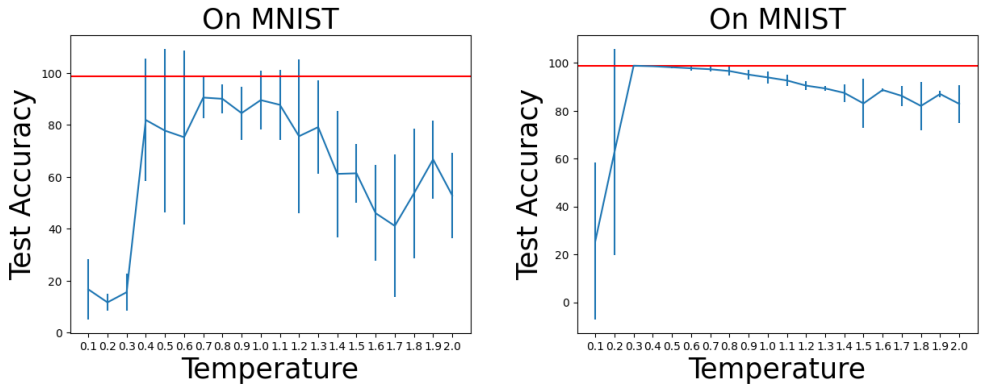


Figure 7: We show the change of the test accuracy as we vary the temperature parameter T. The red horizontal line is the result of the hyperbolic neural networks with the proposed feature clipping. Softmax with temperature scaling with a carefully tuned temperature can approach the performance of the proposed feature clipping. However, it is sensitive to the feature dimension and the temperature parameter. **Left**: the embedding dimension is 2. **Right**: the embedding dimension is 64.

A.8 A MAGNITUDE-CLIPPED HYPERBOLIC SPACE IS STILL HYPERBOLIC

The metric in the hyperbolic space with the clipping strategy is still drastically different from that in the Euclidean space, even with magnitude clipping. For the first example, consider two points: a = [0.5, 0.55], b = [0.3, -0.6], the magnitude of both points is smaller than 0.76. The hyperbolic distance between the two points is 3.1822 while the Euclidean distance is 1.1673. This is a two-dimensional example, with a larger embedding dimension, the difference will be much more significant.

A magnitude-clipped hyperbolic space is still hyperbolic, as the hyperbolic geometry still holds: unlike Euclidean triangles, where the angles always add up to π radians (180°, a straight angle), in hyperbolic geometry the sum of the angles of a hyperbolic triangle is always strictly less than π radians (180°, a straight angle). The difference is referred to as the defect. For a second example, consider three points: A = [0.5, 0.55], B = [0.3, -0.6], C = [-0.1, 0.1]. Their magnitude are all smaller than 0.76. For the triangle ABC, the defect is 58.21° in hyperbolic space and 0° in Euclidean space. This again shows that the clipped hyperbolic space still well maintains the hyperbolic property.

We apply the proposed clipping strategy to learn word embedding as in Nickel & Kiela (2017). We perform the reconstruction task on the transitive closure of the WordNet noun hierarchy. We

compare the embedding quality of the Euclidean space, the hyperbolic space, the hyperbolic space with the proposed clipping using mean average precision (mAP). The embedding dimension is 10. The results are summarized in Table 10.

We have two conclusions here. First, learning word embeddings with hyperbolic space provides better results than learning in Euclidean space. Second, using hyperbolic space with clipping is slightly better than using hyperbolic space without clipping.

Method	**mAP**
Euclidean space	0.059
Hyperbolic space with clipping	0.860
Hyperbolic space	0.851

Table 10: Learning with word embeddings with clipped hyperbolic space outperforms both with Euclidean space and vanilla hyperbolic space.

A.9 Additional regularization to minimize the norm of the Euclidean embedding during training

The results of using the regularization term are shown in Table 11.

Method	**On CIFR10**	**On CIFAR100**
vanilla HNN	88.82	72.26
w/ regularization	92.71	73.34
w/ clipping	94.76	75.88

Table 11: The proposed feature clipping outperforms vanilla HNNs and HNNs with regularization.

We can see that the clipping strategy outperforms the regularization approach. The reason is that with regularization, the loss function consists of two terms: one is the cross-entropy loss and the other is the regularization loss. It is difficult to balance the two terms . During training, if the cross-entropy loss becomes small, the optimization focuses on minimizing the embedding norm, however a small embedding norm is also detrimental to the performance.