Clipped Hyperbolic Classifiers Are Super-Hyperbolic Classifiers

Yunhui Guo[1] Xudong Wang[1] Yubei Chen[2] Stella X. Yu[1]
[1]UC Berkeley / ICSI [2]Facebook AI Research

Abstract

Hyperbolic space can naturally embed hierarchies, unlike Euclidean space. Hyperbolic Neural Networks (HNNs) exploit such representational power by lifting Euclidean features into hyperbolic space for classification, outperforming Euclidean neural networks (ENNs) on datasets with known semantic hierarchies. However, HNNs underperform ENNs on standard benchmarks without clear hierarchies, greatly restricting HNNs' applicability in practice.

Our key insight is that HNNs' poorer general classification performance results from vanishing gradients during backpropagation, caused by their hybrid architecture connecting Euclidean features to a hyperbolic classifier. We propose an effective solution by simply clipping the Euclidean feature magnitude while training HNNs.

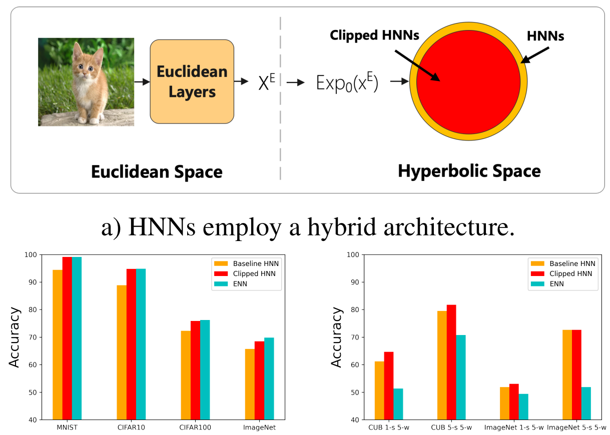
Our experiments demonstrate that clipped HNNs become super-hyperbolic classifiers: They are not only consistently better than HNNs which already outperform ENNs on hierarchical data, but also on-par with ENNs on MNIST, CIFAR10, CIFAR100 and ImageNet benchmarks, with better adversarial robustness and out-of-distribution detection.

1. Introduction

Many datasets are inherently hierarchical. WordNet [30] has a hierarchical conceptual structure, users in social networks such as Facebook or twitter form hierarchies based on different occupations and organizations [11].

Representing such hierarchical data in Euclidean space cannot capture and reflect their semantic or functional resemblance [1, 34]. 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 [34, 40, 41]. For instance, hyperbolic space has been used for analyzing the hierarchical structure in single cell data [20], learning hierarchical word embedding [34], embedding complex networks [1], etc.

Recent algorithms operate directly in hyperbolic space to exploit more representational power. Examples are Hyperbolic Perceptron [46], Hyperbolic Support Vector Machine [5], and Hyperbolic Neural Networks (HNNs) [8], an alternative to standard Euclidean neural networks (ENNs).



a) HNNs employ a hybrid architecture.

b) Standard benchmarks.

c) Few-shot learning tasks.

Figure 1. We propose an effective solution for training HNNs by clipping the Euclidean features. Clipped HNNs become super-hyperbolic classifiers: They are not only consistently better than HNNs which already outperform ENNs on hierarchical data, but also on-par with ENNs on standard benchmarks. a) 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. Clipped HNNs utilize a reduced region of hyperbolic space. b) Clipped HNNs outperform baseline HNNs on standard benchmarks. c) Clipped HNNs outperform both HNNs and ENNs on 1-s (shot) 1-w (way) and 5-s (shot) 5-w (way) few-shot learning tasks.

HNNs adopt a hybrid architecture [18] (Figure 1): An ENN is first used for extracting image features in Euclidean space; they are then projected onto hyperbolic space to be classified by a hyperbolic multiclass logistic regression [8].

While HNNs outperform ENNs on several datasets with explicit hierarchies [8], there are several serious limitations. **1)** HNNs underperform ENNs on standard classification benchmarks with flat or non-hierarchical semantic structures. **2)** Even for image datasets that possess latent hierarchical structures there are no experimental evidence that

HNNs can capture such structures or provide on-par performance with ENNs [18]. **3)** Existing improvements on HNNs mainly focus on reducing the number of parameters [42] or incorporating different types of neural network layers such as attention [10] or convolution [42]. Unfortunately, why HNNs are worse than ENNs on standard benchmarks has not been investigated or understood.

Our key insight is that HNNs' poorer general classification performance is caused by their hybrid architecture connecting Euclidean features to a hyperbolic classifier. It leads to *vanishing gradients* during training. In particular, the training dynamics of HNNs push the hyperbolic embeddings to the boundary of the Poincaré ball [2] which causes the gradients of Euclidean parameters to vanish.

We propose a simple yet effective solution to this problem by simply clipping the Euclidean feature magnitude during training, thereby preventing the hyperbolic embedding from approaching the boundary during training. Our experiments demonstrate that clipped HNNs become super-hyperbolic classifiers: They are not only consistently better than HNNs which already outperform ENNs on hierarchical data, but also on-par with ENNs on MNIST, CIFAR10, CIFAR100 and ImageNet benchmarks, with better adversarial robustness and out-of-distribution detection.

Our paper makes the following contributions. **1)** Our detailed analysis reveals the underlying issue of vanishing gradients that makes HNNs worse than ENNs on standard classification benchmarks. **2)** We propose a simple yet effective feature clipping solution. **3)** Our extensive experimentation demonstrates that clipped HNNs outperform standard HNNs and become on-par with ENNs on standard benchmarks. They are also more robust to adversarial attacks and exhibit stronger out-of-distribution detection capability than their Euclidean counterparts.

2. Related Work

Supervised Learning in Hyperbolic Space. Several hyperbolic neural networks were proposed in the seminal work of HNNs [8], including multinomial logitstic regression (MLR), fully connected and recurrent neural networks which can operate directly on hyperbolic embeddings, outperforming Euclidean variants on text entailment and noisy-prefix prediction tasks. Hyperbolic Neural Networks++ [42] reduces the number of parameters of HNNs and also introduces hyperbolic convolutional layers. Hyperbolic attention networks [10] rewrite the operations in the attention layers using gyrovector operations [44], delivering gains on neural machine translation, learning on graphs and visual question answering. Hyperbolic graph neural network [25] extends the representational geometry of Graph Neural Networks (GNNs) [54] to hyperbolic space. Hyperbolic graph attention networks [52] further studies GNNs with attention mechanisms in hyperbolic space. HNNs have been used for few-shot classification and person re-identification [18].

Unsupervised Learning in Hyperbolic Space. [32] uses a wrapped normal distribution in hyperbolic space to construct hyperbolic variational autoencoders (VAEs) [19], whereas [29] uses Gaussian generalizations in hyperbolic space to construct Poincaré VAEs. [17] applies hyperbolic neural networks to unsupervised 3D segmentation of complex volumetric data.

Our work differs from all the above-mentioned methods which focus on the application of HNNs to data with natural tree structures. We extend HNNs to standard classification benchmarks which may not have hierarchies. By improving HNNs to the level of ENNs in these scenarios, we greatly enhance the general applicability of HNNs.

3. Super-Hyperbolic Classifiers from Clipping

Our goal is to understand why HNNs underperform ENNs on standard image classification benchmarks and propose corresponding solutions. First, we review the basics of Riemannian geometry and HNNs. Then, we analyze the vanishing gradient problem in training HNNs. Finally, we present the proposed method and discuss its connections to existing methods.

3.1. Preliminaries

Smooth Manifold. 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.

Riemannian Manifold. 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.

Inner Product and Norm on Riemannian Manifold. 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 $\|\mathbf{v}\|_{\mathbf{x}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{\mathbf{x}}}$.

Geodesics on Riemannian Manifold. A geodesic is a curve $\gamma : [0, 1] \to \mathcal{M}$ of constant speed that is locally minimizing the distance between two points on the manifold.

Exponential Map on Riemannian 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 [4, 24]

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

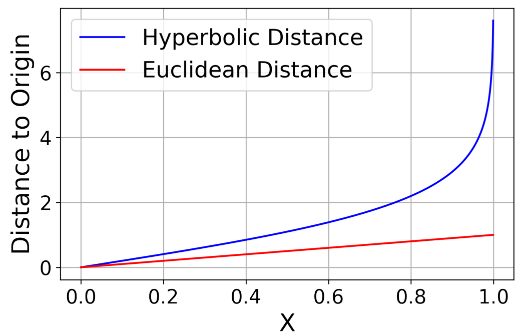


Figure 2. Hyperbolic distance grows exponentially as we move towards the boundary of the Poincaré ball.

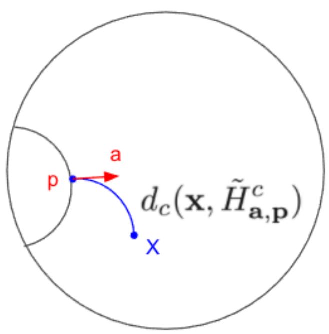


Figure 3. Poincaré hyperplane defined by $\mathbf{a}$ and $\mathbf{p}$. Blue line is the orthogonal projection of $\mathbf{x}$ to the Poincaré hyperplane.

model [8, 34] which can be derived using stereoscopic projection of the hyperboloid model [2]. The n-dimensional Poincaré ball model of constant negative curvature $-c$ is defined as $(\mathbb{B}^n_c, \mathfrak{g}^c_{\mathbf{x}})$, where $\mathbb{B}^n_c = \{\mathbf{x} \in \mathbb{R}^n : c\|\mathbf{x}\| < 1\}$ and $\mathfrak{g}^c_{\mathbf{x}} = (\gamma^c_{\mathbf{x}})^2 I_n$ is the Riemannian metric tensor. I_n is the Euclidean metric tensor. The conformal factor is defined as,

$$\gamma^c_{\mathbf{x}} = \frac{2}{1 - c\|\mathbf{x}\|^2} \tag{1}$$

The conformal factor induces the inner product $\langle \mathbf{u}, \mathbf{v} \rangle^c_{\mathbf{x}} = (\gamma^c_{\mathbf{x}})^2 \langle \mathbf{u}, \mathbf{v} \rangle$ and norm $\|\mathbf{v}\|^c_{\mathbf{x}} = \gamma^c_{\mathbf{x}} \|\mathbf{v}\|$ for all $\mathbf{u}, \mathbf{v} \in T_{\mathbf{x}} \mathbb{B}^n_c$. The exponential map of Poincaré ball model can be written analytically with the operations of gyrovector space.

Distance in Poincaré Ball Model. Figure 2 shows that hyperbolic distance grows much faster than Euclidean distance as we move towards the boundary of the Poincaré ball. As we will show later, this fundamental fact would lead to an optimization issue when we construct a neural network consisting of both Euclidean and hyperbolic layers.

Gyrovector Space. A gyrovector space [44, 45] is an algebraic structure that provides an analytic way to operate in hyperbolic space. Gyrovector space can be used to define various operations such as scalar multiplication, subtraction, exponential map, inverse exponential map in Poincaré ball model.

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,

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

for all $\mathbf{u}, \mathbf{v} \in \mathbb{B}^n_c$. Particularly, $\lim_{c \to 0} \oplus_c$ converges to the standard $+$ in the Euclidean space. For more details, please refer to the Supplementary.

Hyperbolic Neural Networks. Hyperbolic neural networks consist of an Euclidean sub-network and a hyperbolic classifier (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 $\mathbb{B}^n_c$ via an exponential map $\mathrm{Exp}^c_{\mathbf{0}}(\cdot)$ as $\mathbf{x}^H \in \mathbb{B}^n_c$. 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,

$$\min_{\mathbf{w}^E, \mathbf{w}^H} \ell(H(\mathrm{Exp}^c_{\mathbf{0}}((E(\mathbf{x}; \mathbf{w}^E)); \mathbf{w}^H), y) \tag{3}$$

where the outer and inner functions are $H : \mathbb{B}^n_c \to \mathbb{R}$ and $E : \mathbb{R}^m \to \mathbb{R}^n$. As shown in [8], the exponential map is defined as,

$$\mathrm{Exp}^c_{\mathbf{0}}(\mathbf{v}) = \tanh(\sqrt{c}\|\mathbf{v}\|)\frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|} \tag{4}$$

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

Definition 3.1 (Poincaré hyperplanes [8]) *For* $\mathbf{p} \in \mathbb{B}^n_c$, $\mathbf{a} \in T_{\mathbf{p}}\mathbb{B}^n_c \setminus \{\mathbf{0}\}$, *the Poincaré hyperplane is defined as,*

$$\tilde{H}^c_{\mathbf{a},\mathbf{p}} := \{\mathbf{x} \in \mathbb{B}^n_c : \langle -\mathbf{p} \oplus_c \mathbf{x}, \mathbf{a} \rangle = 0\} \tag{5}$$

where $\mathbf{a}$ *is the normal vector. Figure 3 shows the Poincaré hyperplane defined by* $\mathbf{a}$ *and* $\mathbf{p}$.

[8] shows that in hyperbolic space the probability that a given $\mathbf{x} \in \mathbb{B}^n_c$ is classified as class k is,

$$p(y = k|\mathbf{x}) \propto \exp(\langle -\mathbf{p}_k \oplus_c \mathbf{x}, \mathbf{a}_k \rangle) \sqrt{\mathfrak{g}^c_{\mathbf{p_k}}(\mathbf{a}_k, \mathbf{a}_k)} d_c(\mathbf{x}, \tilde{H}^c_{\mathbf{a}_k,\mathbf{p}}) \tag{6}$$

where $d_c(\mathbf{x}, \tilde{H}^c_{\mathbf{a}_k,\mathbf{p}})$ is the distance of the embedding $\mathbf{x}$ to the Poincaré hyperplane of class k as shown in Figure 3. In hyperbolic classifier, the parameters are the vectors $\{\mathbf{p}_k\}$ for each class k.

3.2. Vanishing Gradient Problem in Training Hyperbolic Neural Networks

Training Hyperbolic Neural Networks with Backpropagation. The standard backpropagation algorithm [38] is used for training HNNs [8, 18]. During backpropagation, the gradient of the Euclidean parameters $\mathbf{w}^E$ can be computed as,

$$\frac{\partial \ell}{\partial \mathbf{w}^E} = (\frac{\partial \mathbf{x}^H}{\partial \mathbf{w}^E})^T \frac{\partial \ell}{\partial \mathbf{x}^H} \tag{7}$$

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 [3] and

$$\frac{\partial \ell}{\partial \mathbf{x}^H} = \frac{(1 - \|\mathbf{x}^H\|^2)^2}{4} \nabla \ell(\mathbf{x}^H) \tag{8}$$

where $\nabla \ell(\mathbf{x}^H)$ is the Euclidean gradient.

Vanishing Gradient Problem. We conduct an experiment to show the vanishing gradient problem during training hyperbolic neural networks. We train a LeNet-like convolutional neural network [23] with hyperbolic classifier on the MNIST data. We use a two-dimensional Poincaré ball for visualization. Figure 4 shows 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.

From Equation 6, 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 7, 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 $\mathbf{w}^E$.

Vanishing gradient problem [12, 15, 36, 37] 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 Euclidean neural networks, vanishing gradient problem can be alleviated by architecture designs [14, 16], proper weight initialization [31] and carefully chosen activation functions [48]. However, the vanishing gradient problem in training HNNs is not exploited in existing literature.

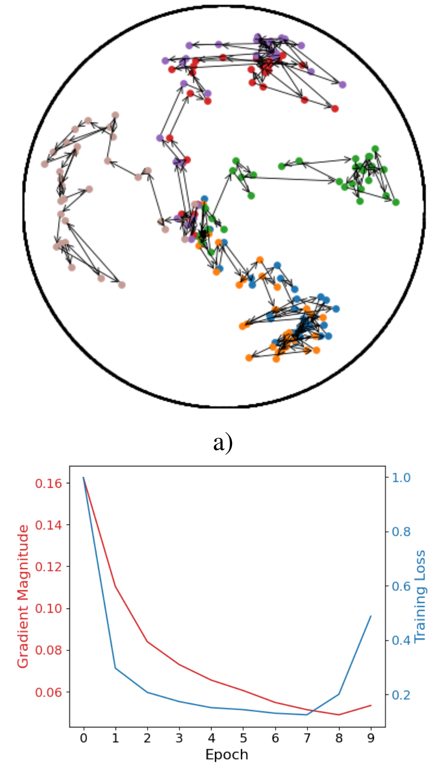


Figure 4. Hyperbolic neural networks suffer from vanishing gradient problem during training with backpropagation. a) The trajectories of the hyperbolic embeddings of six randomly sampled inputs during training in a 2-dimensional Poincaré ball. The arrows indicate the change of location of each embedding with each gradient update. The embeddings move to the boundary of the ball during optimization which causes vanishing gradient problem. b) The gradient vanishes while the training loss goes up at the end of training.

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 the Supplementary. For the Euclidean sub-network $E : \mathbb{R}^m \to \mathbb{R}^n$, consider the first-order Taylor-expansion with a single gradient update,

$$\begin{aligned} E(\mathbf{w}^E_{t+1}) &= E(\mathbf{w}^E_t + \eta \frac{\partial \ell}{\partial \mathbf{w}^E}) \\ &\approx E(\mathbf{w}^E_t) + \eta (\frac{\partial E(\mathbf{w}^E_t)}{\partial \mathbf{w}^E_t})^T \frac{\partial \ell}{\partial \mathbf{w}^E} \end{aligned} \tag{9}$$

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

$$\nabla \text{Exp}_{\mathbf{0}}^c(\mathbf{v}) = \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|}\nabla \tanh(\sqrt{c}\|\mathbf{v}\|) + \tanh(\sqrt{c}\|\mathbf{v}\|)\nabla \frac{\mathbf{v}}{\sqrt{c}\|\mathbf{v}\|}$$
$$= 1 - \tanh^2(\sqrt{c}\|\mathbf{v}\|) + \tanh(\sqrt{c}\|\mathbf{v}\|)\frac{1}{\sqrt{c}}\frac{2}{\|\mathbf{v}\|} \tag{10}$$

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

$$\mathbf{x}_{t+1}^H = \text{Exp}_{\mathbf{0}}^c(E(\mathbf{w}_{t+1}^E)) \tag{11}$$

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

$$\mathbf{x}_{t+1}^H = \mathbf{x}_t^H + C(E(\mathbf{w}_t^E)^T)\frac{\partial \ell}{\partial \mathbf{w}^E} \tag{12}$$

where $C(E(\mathbf{w}_t^E)) = \nabla \text{Exp}_{\mathbf{0}}^c(E(\mathbf{w}_t^E))^T \eta(\frac{\partial E(\mathbf{w}_t^E)}{\partial \mathbf{w}_t^E})^T$. Thus once $\|\mathbf{x}_t^H\|$ approaches one, from Equation 7 and Equation 8 we can find that the hyperbolic embedding stagnates no matter how large the training loss is.

3.3. Clipped Hyperbolic Neural Networks

Euclidean Feature Clipping. There are several possible solutions to address the vanishing gradient problem for training HNNs. 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 [13] 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 3 with a regularization term which controls the magnitude of hyperbolic embeddings,

$$\min_{\mathbf{w}^E, \mathbf{w}^H} \ell(H(\mathbf{x}^H; \mathbf{w}^H), y) + \beta\|\mathbf{x}^H\|^2 \tag{13}$$

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 13 is effective, it introduces additional complexity to the optimization process and has worse performance. We instead employ the following hard constraint which regularizes the Euclidean embedding before the exponential map whenever its norm exceeds a given threshold,

$$\text{CLIP}(\mathbf{x}^E; r) = \min\{1, \frac{r}{\|\mathbf{x}^E\|}\} \cdot \mathbf{x}^E \tag{14}$$

where $\mathbf{x}^E = E(\mathbf{x}; \mathbf{w}^E)$ and $r > 0$ is a hyperparameter. The clipped Euclidean embedding is projected via exponential map as $\text{CLIP}(\mathbf{x}^H; r) = \text{Exp}_{\mathbf{0}}^c(\text{CLIP}(\mathbf{x}^E; r))$. The relation between the clipping value and the effective radius of Poincaré ball is depicted in Figure 5.

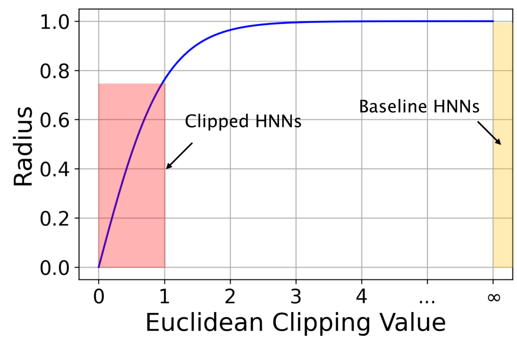


Figure 5. The relation between the clipping value r and the effective radius of the Poincaré ball, i.e., $\text{Exp}_{\mathbf{0}}^c(\text{CLIP}(\mathbf{x}^E; r))$. Clipped HNNs utilize a reduced region of Poincaré ball.

Using the relation between the hyperbolic distance and the Euclidean distance to the origin,

$$d_c(0, x) = s \ln(\frac{s + x}{s - x}) \tag{15}$$

where $s = 1/\sqrt{|c|}$, c is the curvature and x is Euclidean distance to the origin. c is usually set to -1. The clipping value r can be further converted into hyperbolic radius $r_{\mathbb{B}_c^n}$ as below,

$$r_{\mathbb{B}_c^n} = d_c(0, \text{CLIP}(\mathbf{x}^H; r)) = 2r \tag{16}$$

The hyperbolic radius $r_{\mathbb{B}_c^n}$ is measured in hyperbolic distance. The hyperbolic embeddings are within a hyperbolic ball of radius of $r_{\mathbb{B}_c^n}$.

Discussion on Feature Clipping. 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 the Supplementary, we show that clipped hyperbolic space well maintains the hyperbolic property and delivers better results for learning hierarchical word embeddings.

Discussion on Hyperbolic Embedding Literature. Similar regularization approaches have been used in the hyperbolic embedding literature to prevent numerical issues when optimizing hyperbolic embeddings [25,35]. In contrast, our work is focused on the *hyperbolic neural networks* for image classification and its unique *vanishing gradient issue*, which is drastically different from [25,35] in terms of model architecture and the focused problem. In hyperbolic neural networks, the gradients are backpropagated through the hyperbolic layers to the Euclidean layers which causes the gradients to vanish. The vanishing gradient issue will not occur in [25, 35] since Euclidean layers are not adopted.

Lorentz model is used recently to overcome the numerical issues of Poincaré ball model for learning word embeddings [35]. However, it is most effective only in low dimensions [25]. For image datasets of ImageNet-scale, hyperbolic neural networks with high-dimensional embeddings are necessary for enough model capacity.

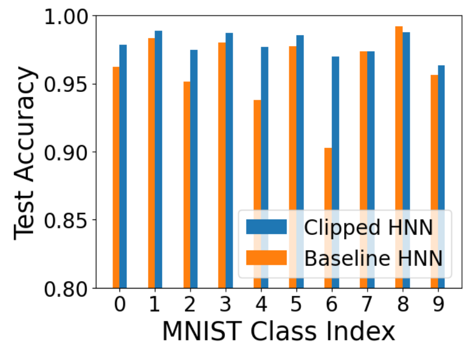
4. Experimental Results

We conduct four types of experiments: standard balanced classification tasks, few-shot learning tasks, adversarial robustness and out-of-distribution detection. The results show that clipped HNNs are on par with ENNs on standard recognition datasets while demonstrating better performance in terms of few-shot classification, adversarial robustness and out-of-distribution detection.

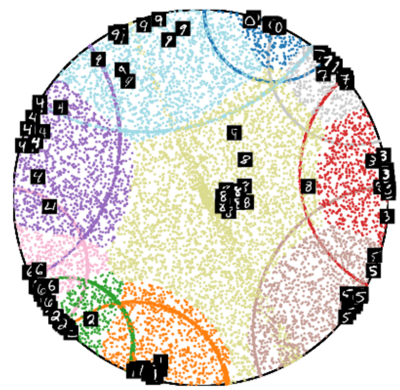
Datasets. We consider four commonly used image classification datasets: MNIST [22], CIFAR10 [21], CIFAR100 [21] and ImageNet [7]. See details in the Supplementary. To our best knowledge, this paper is the first attempt to extensively evaluate hyperbolic neural networks on the standard image classification datasets for supervised classification.

Baselines and Networks. We compare the performance of HNNs training with/without the proposed feature clipping method [8, 18] and their Euclidean counterparts. For MNIST, we use a LeNet-like convolutional neural network [23] which has two convolutional layers with max pooling layers in between and three fully connected layers. For CIFAR10 and CIFAR100, we use WideResNet [51]. For ImageNet, we use a standard ResNet18 [14].

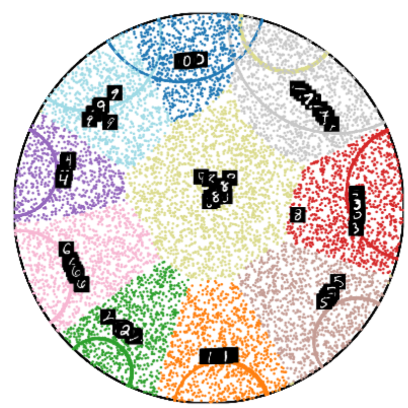
Training Setups. For training ENNs, we use SGD with momentum. For training HNNs, the Euclidean parameters of HNNs are trained using SGD, and the hyperbolic parameters of HNNs are optimized using stochastic Riemann gradient descent [3], just like the previous method. For training networks on MNIST, we 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 [27]. 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 the Supplementary. For baseline HNNs, we use a clipping value of 15 similar to [25, 35] 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.



a) Accuracy of each class on MNIST.



b) Baseline HNNs



c) Clipped HNNs

Figure 6. Clipped HNNs greatly outperform baseline HNNs. a) The per class test accuracy of baseline HNNs and Clipped HNNs. b) The Poincaré decision hyperplanes and the hyperbolic embeddings of sampled test images of baseline HNNs. c) The Poincaré decision hyperplanes and the hyperbolic embeddings of sampled test images of clipped HNNs. Clipped HNNs learn more discriminative feature in hyperbolic space. The per class accuracy indicates that baseline HNNs learn biased feature space which hurts the performance on certain classes.

Results on Standard Benchmarks. Table 1 shows the results of different networks on the considered benchmarks. On MNIST, we can observe that the accuracy of the im-

Standard Classification			
Task	Euclidean [14]	Hyperbolic [8]	C-Hyperbolic
MNIST	99.12±0.34	94.42±0.29	99.08±0.31
CIFAR10	94.81±0.42	88.82±0.51	94.76±0.44
CIFAR100	76.24±0.35	72.26±0.41	75.88±0.38
ImageNet	69.82	65.74	68.45
Few-Shot Classification on CUB Dataset			
1-Shot 5-Way	51.31±0.91	61.18±0.24	64.66±0.24
5-Shot 5-Way	70.77±0.69	79.51±0.16	81.76±0.15
Few-Shot Classification on MiniImageNet Dataset			
1-Shot 5-Way	49.42±0.78	51.88±0.20	53.01±0.22
5-Shot 5-Way	51.88±0.20	72.63±0.16	72.66±0.15

Table 1. Clipped HNN approaches ENN on standard classification benchmarks. Clipped hyperbolic ProtoNet (C-Hyperbolic) greatly outperforms standard hyperbolic ProtoNet (Hyperbolic) and Euclidean ProtoNet (Euclidean) on few-shot learning tasks.

proved clipped HNNs is about 5% higher than the baseline HNNs and match the performance of ENNs. 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.

Figure 6 shows the Poincaré hyperplanes of all the classes and the hyperbolic embeddings of 1000 sampled test images extracted by the baseline HNNs and clipped HNNs. 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 restricted region of the whole Poincaré ball and the network learns more regular and discriminative features in hyperbolic space.

Few-Shot Learning. We show that the proposed feature clipping can also improve the performance of Hyperbolic ProtoNet [18] for few-shot learning. Different from the standard ProtoNet [43] which computes the prototype of each class in Euclidean space, Hyperbolic ProtoNet computes the class prototype in hyperbolic space using hyperbolic averaging. Hyperbolic features are shown to be more effective than Euclidean features for few-shot learning [18].

We follow the experimental settings in [18] and conduct experiments on CUB [47] and miniImageNet dataset [39]. We consider 1-shot 5-way and 5-shot 5-way tasks as in [18]. The evaluation is repeated for 10000 times and we report the average performance and the 95% confidence interval. Table 1 shows that the proposed feature clipping further improves the accuracy of Hyperbolic ProtoNet for few-shot classification by as much as 3%.

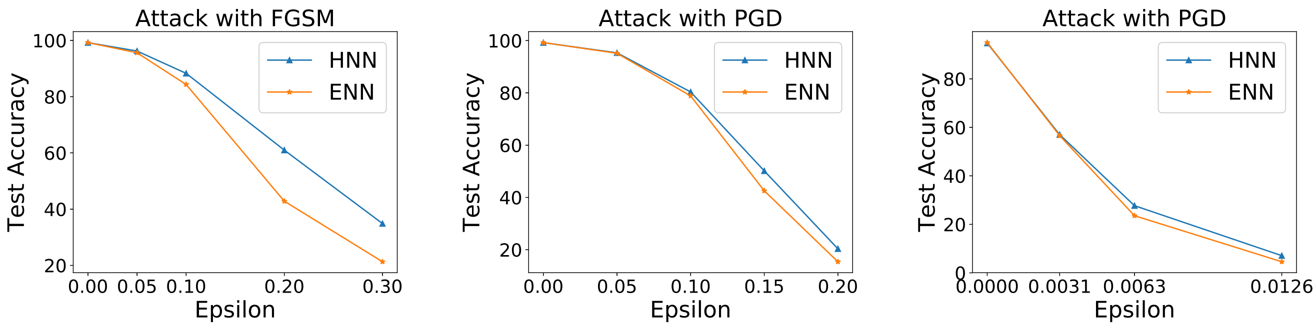
Table 2. **Clipped HNNs consistently outperform ENNs (shaded in gray) on out-of-distribution (OOD) detection with *softmax scores* when trained on CIFAR10 and tested on OOD datasets.**

OOD Dataset	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 3. **Clipped HNNs consistently outperform ENNs (shaded in gray) on out-of-distribution (OOD) detection with *softmax scores* when trained on CIFAR100 and tested on OOD datasets. On average, they are on par with ENNs by AUPR and far better by PRR95 and AUROC.**

OOD Dataset	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

Adversarial Robustness. We show that clipped HNNs are more robust to adversarial attacks including FGSM [9] and PGD [28] than ENNs. For attacking networks



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

Figure 7. Clipped HNNs are more robust to ENNs against adversarial attacks. We show the results of **adversarial robustness** of clipped HNNs and ENNs to different attack methods and perturbations. Clipped HNNs are consistently better than ENNs.

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 7 we can see that across all the cases clipped HNNs show more robustness than ENNs to adversarial attacks. For more discussions and results using vanilla HNNs, please see the Supplementary.

Out-of-Distribution Detection. We conduct experiments to show that clipped HNNs have stronger out-of-distribution detection capability than ENNs. Out-of-distribution detection aims at determining whether or not a given input is from the same distribution as the training data. We follow the experimental settings in [26]. The in-distribution datasets are CIFAR10 and CIFAR100. The out-of-distribution datasets are ISUN [49], Place365 [53], Texture [6], SVHN [33], LSUN-Crop [50] and LSUN-Resize [50]. For detecting out-of-distribution data, we use both softmax score and energy score as described in [26]. For metrics, we consider FPR95, AUROC and AUPR [26]. Table 2 and 3 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 and vanilla HNNs, please see the Supplementary.

The Effect of Feature Dimension. Figure 8 shows the change of test accuracy as we vary the feature dimension on CIFAR10 and CIFAR100. Clipped HNNs are much better than ENNs when the feature dimension is low. One possible reason is that when the dimension is low in the Euclidean case, the data are hard to be linearly separated. However in hyperbolic space, since the Poincaré hyperplanes are "curved", the data are more likely to be linearly separated even in two dimensions.

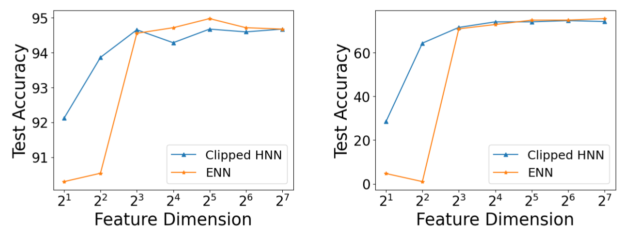


Figure 8. Clipped HNNs are better than ENNs when the feature dimension is low. **Left**: Test accuracy on CIFAR10. **Right**: Test accuracy on CIFAR100. We change the feature dimension from 2 to 128.

5. Summary

We propose a simple yet effective solution called *Feature Clipping* to address the vanishing gradient problem in training HNNs. We conduct extensive experiments on commonly used image dataset benchmarks. To the best of our knowledge, this is the first time that HNNs can be applied to image datasets of ImageNet-scale. Clipped HNNs show significant improvement over baseline HNNs and match the performance of ENNs. The proposed feature clipping also improves the performance of HNNs for few-shot learning. Further experimental studies reveal that clipped HNNs are more robust to adversarial attacks such as PGD and FGSM. Clipped HNNs also show stronger out-of-distribution detection capability than ENNs. When the feature dimension is low, clipped HNNs even outperform ENNs.

Acknowledgements. This work was supported, in part, by Berkeley Deep Drive and the National Science Foundation under Grant No. 2131111. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] 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. 1

[2] James W Anderson. *Hyperbolic geometry*. Springer Science & Business Media, 2006. 2, 3

[3] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013. 4, 6

[4] Manfredo Perdigao do Carmo. *Riemannian geometry*. Birkhäuser, 1992. 2

[5] Hyunghoon Cho, Benjamin DeMeo, Jian Peng, and Bonnie Berger. Large-margin classification in hyperbolic space. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1832–1840. PMLR, 2019. 1

[6] 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*, pages 3606–3613, 2014. 8

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[8] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *arXiv preprint arXiv:1805.09112*, 2018. 1, 2, 3, 4, 6, 7

[9] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 7

[10] 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. 2

[11] 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*, pages 557–566, 2011. 1

[12] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *arXiv preprint arXiv:1801.03744*, 2018. 4

[13] Elad Hazan, Kfir Y Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *arXiv preprint arXiv:1507.02030*, 2015. 5

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4, 6, 7

[15] 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. 4

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4

[17] 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. 2

[18] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6428, 2020. 1, 2, 4, 6, 7

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[20] 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. 1

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[22] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*. 6

[23] 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. 4, 6

[24] John M Lee. *Introduction to Riemannian manifolds*. Springer, 2018. 2

[25] Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *arXiv preprint arXiv:1910.12892*, 2019. 2, 5, 6

[26] Weitang Liu, Xiaoyun Wang, John D Owens, and Yixuan Li. Energy-based out-of-distribution detection. *arXiv preprint arXiv:2010.03759*, 2020. 8

[27] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[28] 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. 7

[29] 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. 2

[30] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995. 1

[31] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015. 4

[32] 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*, pages 4693–4702. PMLR, 2019. 2

[33] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 8

[34] Maximilian Nickel and Douwe Kiela. Poincar\'e embeddings for learning hierarchical representations. *arXiv preprint arXiv:1705.08039*, 2017. 1, 3

[35] Maximillian Nickel and Douwe Kiela. Learning continuous hierarchies in the lorentz model of hyperbolic geometry. In *International Conference on Machine Learning*, pages 3779–3788. PMLR, 2018. 5, 6

[36] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1924–1932. PMLR, 2018. 4

[37] 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. 4

[38] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. 4

[39] 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. 7

[40] Frederic Sala, Chris De Sa, Albert Gu, and Christopher Ré. Representation tradeoffs for hyperbolic embeddings. In *International conference on machine learning*, pages 4460–4469. PMLR, 2018. 1

[41] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pages 355–366. Springer, 2011. 1

[42] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. Hyperbolic neural networks++. *arXiv preprint arXiv:2006.08210*, 2020. 2

[43] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017. 7

[44] Abraham A Ungar. *Analytic hyperbolic geometry: Mathematical foundations and applications*. World Scientific, 2005. 2, 3

[45] Abraham Albert Ungar. A gyrovector space approach to hyperbolic geometry. *Synthesis Lectures on Mathematics and Statistics*, 1(1):1–194, 2008. 3

[46] 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. 1

[47] 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. 7

[48] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015. 4

[49] 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*, 2015. 8

[50] 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. 8

[51] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 6

[52] Yiding Zhang, Xiao Wang, Xunqiang Jiang, Chuan Shi, and Yanfang Ye. Hyperbolic graph attention network. *arXiv preprint arXiv:1912.03046*, 2019. 2

[53] 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. 8

[54] 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. 2