HYPERBOLIC BINARY NEURAL NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Binary Neural Network (BNN) converts the full-precision weights and activations to the extreme 1-bit counterparts, which is especially suitable to be deployed on lightweight mobile devices. Neural network binarization is usually formulated as a constrained optimization problem, which restricts its optimized potential. In this paper, we introduce the dynamic exponential map that converts a constrained problem in the Riemannian manifold into an unconstrained one in the Euclidean space. Specifically, we propose a Hyperbolic Binary Neural Network (HBNN) by representing the parameter vector in the Euclidean space as the one in the hyperbolic space, which would enable us to optimize the parameter in an unconstrained space. By analyzing the parameterized representation, we present that the dynamic exponential map is a diffeomorphism in the Poincaré ball. Theoretically, this property will not create extra saddle points or local minima in the Poincaré ball, which also explains the good performance of the HBNN. Experiments on CIFAR10, CIFAR100, and ImageNet classification datasets with VGGsmall, ResNet18, and ResNet34 demonstrate the superiorities of our HBNN over existing state-of-the-art methods.

1 INTRODUCTION

Deep Neural Networks (DNNs) have greatly succeeded in various computer vision fields such as image classification (Krizhevsky et al., 2012; He et al., 2016), object detection (Redmon et al., 2016; He et al., 2017), semantic segmentation (Long et al., 2015; Noh et al., 2015), etc. However, such success is greatly attributed to the massive parameters and computational complexity of DNNs, which limits the deployment of DNNs to lightweight mobile devices. To address this problem, many model-based compression methods are being proposed, mainly including pruning (Ding et al., 2019b; Lin et al., 2020a) and quantization (Banner et al., 2018; Helwegen et al., 2019).

Quantization seems to be a better and more general choice for resource-constrained and low-power devices than pruning (Chen et al., 2021). Specifically, quantization converts the full-precision weights and activations into low-precision counterparts. In the extreme case, neural network binarization restricts its weights and activations to two possible discrete values (-1 or $+1$), which brings two benefits: (a) $32\times$ reduction in memory than the corresponding full-precision version; (b) the multiply-accumulation operation can be replaced with the efficient xnor and bitcount operations.

Neural network binarization is always formulated as a constrained optimization problem depending on the dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{m}$ and the set of all possible binarized solutions $\mathcal{X} \subset \mathbb{R}^n$:

$$\min_{\mathbf{w}\in\mathcal{X}} \mathcal{L}(\mathbf{w};\mathcal{D}) := \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\mathbf{w};(\mathbf{x}_i,\mathbf{y}_i)\right), \tag{1}$$

where $\mathbf{w}$ is the n dimensional parameter vector and $\mathcal{L}$ is the loss function (e.g., cross-entropy loss). Based on a mirror descent framework (Bubeck et al., 2015), MD (Ajanthan et al., 2021) converts the constrained problem into an unconstrained one via a mapping $P : \mathbb{R}^n \to \mathcal{X}$ such that

$$\min_{\tilde{\mathbf{w}}\in\mathbb{R}^n} \mathcal{L}(P(\tilde{\mathbf{w}});\mathcal{D}). \tag{2}$$

And then $P(\tilde{\mathbf{w}}) \in \mathcal{X}$ is not binarized to a discrete set $\mathcal{B}^n = \{-1,+1\}^n$ until the end of training. In particular, MD (Ajanthan et al., 2021) requires the constrained space to be convex and compact when P is defined as a mirror map (convex function).

In this paper, we extend the constrained optimization problem of binarization from the constraint set $\mathcal{X}$ to a Riemannian manifold $\mathcal{M}$. In the context of a connected manifold, we consider the transformation of a constrained optimization problem into an unconstrained one:

$$\text{Original problem: } \min_{\mathbf{w}\in\mathcal{M}} \mathcal{L}(\mathbf{w};\mathcal{D}) \qquad \text{Unconstrained problem: } \min_{\tilde{\mathbf{w}}\in\mathbb{R}^n} \mathcal{L}(\phi(\tilde{\mathbf{w}});\mathcal{D}). \tag{3}$$

In this way, we replace the mirror map P with the Riemannian exponential map $\phi : \mathbb{R}^n \to \mathcal{M}$ (Petersen, 2006) that is a differentiable map from the tangent space to a connected manifold. Thus, the parameterized representation $\phi(\tilde{\mathbf{w}})$ also acts the metric on $\mathcal{M}$, and performs Riemannian gradient descent on the original optimization problem.

Motivation. We note that the length of the binarized parameter vector in a layer is fixed, which is only related to the dimension n of the parameter vector. Consequently, the binarized parameter vector is constantly updated on a ball with the radius $|\mathcal{B}^n|$. Following such properties, we consider the connected manifold as the Poincaré ball (Ganea et al., 2018a). Then the constrained optimization problem of neural network binarization on the hyperbolic space seems reasonable and helpful.

The main contributions of this paper are summarized as:

1. By introducing the dynamic exponential map, we provide a Riemannian geometry framework to formulate neural network binarization as an unconstrained optimization problem. Specifically, we propose a Hyperbolic Binary Neural Network (HBNN) that represents the parameter vector in the Euclidean space as the one in the hyperbolic space, which would enable us to optimize the parameter in an unconstrained space.
2. Theoretically, the dynamic exponential map is a diffeomorphism in the Poincaré ball, and will not add extra saddle points or local minima, which seems to indicate that the good performance of the HBNN is benefited from the hyperbolic representation.
3. Practically, experiments on CIFAR10, CIFAR100, and ImageNet classification datasets with VGGsmall, ResNet18, and ResNet34 demonstrate the superiorities of our HBNN over existing state-of-the-art methods.

2 Preliminaries

Here we present background knowledge about the Riemannian geometry and BNNs.

2.1 Riemannian Geometry

We briefly introduce the basic concepts of Riemannian geometry used in this work. For more in-depth propositions, see (Petersen, 2006; Guggenheimer, 2012).

Tangent Space. For an n-dimensional connected manifold $\mathcal{M}$, the tangent space at a point $p \in \mathcal{M}$ is defined as $T_p\mathcal{M}$. This is a real vector space that can be described as a high-dimensional generalization of a tangent plane. And such a tangent space exists for all points $p \in \mathcal{M}$. In this way, the description of tangent spaces is consistent with the Euclidean space, i.e., $T_p\mathcal{M} \cong \mathbb{R}^n$.

Riemannian Manifold. Riemannian manifolds endow with a smooth metric $g_p : T_p\mathcal{M}\times T_p\mathcal{M} \to \mathbb{R}$ varying smoothly with p, which allows us to construct a distance $d_g : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$. While we describe a Riemannian manifold, the Riemannian metric is always equipped by default, i.e., $(\mathcal{M}, g)$.

Geodesics. Given a complete Riemannian manifold, a smooth path of minimal length between two points on $\mathcal{M}$ is called a geodesic, which is defined as $\gamma_{p,v}(t) : t \in [0,1] \to \mathcal{M}$ such that $\gamma_{p,v}(0) = p$, $\gamma'_{p,v}(0) = v$ for $v \in T_p\mathcal{M}$. It is the generalization of a straight line in Euclidean space.

Exponential Map. The Riemannian exponential map is defined as $\exp_p : T_p\mathcal{M} \to \mathcal{M}$ that maps rays starting at the origin in the tangent space $T_p\mathcal{M}$ to geodesics on $\mathcal{M}$. Given a geodesic, the range of its parameter t is $[0,1]$, thus $\exp_p(tv) := \gamma_{p,v}(t)$. Specifically, the distance on the manifold between a point p and the exponential map $\exp_p(v)$ is $d_g(p, \exp_p(v)) = \|v\|_g$.

Based on the above exponential map $\exp_p : T_p\mathcal{M}(\cong \mathbb{R}^n) \to \mathcal{M}$, we consider a parameter vector $\tilde{\mathbf{w}} \in \mathbb{R}^n$, and achieve the parameterized representation of the parameter vector $\mathbf{w} = \exp_p(\tilde{\mathbf{w}}) \in \mathcal{M}$.

2.2 Binary Neural Network

Now we detail the mechanism of BNNs, and show how we compute the binarization and gradients.

Forward Pass. For the inference phase of a BNN, the binarization function can be simply expressed as the deterministic form (Courbariaux et al., 2016; Rastegari et al., 2016):

$$x^b = \operatorname{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0, \\ -1 & \text{otherwise,} \end{cases} \tag{4}$$

where x can be the weights $\mathbf{w}$ or activations $\mathbf{a}$.

Backward Pass. In back-propagation, the gradient will suffer from either infinite or zero while propagating through the binarization function. (Hinton et al., 2012; Bengio et al., 2013) proposed the Straight-Through Estimator to solve this problem. Then an estimator of the gradient with respect to binarized weights can be simply approximated by

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial \mathbf{w}^b} \cdot \frac{\partial \mathbf{w}^b}{\partial \mathbf{w}}, \quad \text{where } \frac{\partial \mathbf{w}^b}{\partial \mathbf{w}} := \begin{cases} 1 & \text{if} \quad |\mathbf{w}| \leq 1 \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

On the other hand, base on the polynomial function (Liu et al., 2020), an estimator of the gradient with respect to binarized activations can be written as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^b} \cdot \frac{\partial \mathbf{a}^b}{\partial \mathbf{a}}, \quad \text{where } \frac{\partial \mathbf{a}^b}{\partial \mathbf{a}} := \begin{cases} 2 + 2\mathbf{a}, & \text{if } -1 \leq \mathbf{a} < 0 \\ 2 - 2\mathbf{a}, & \text{if } 0 \leq \mathbf{a} \leq 1 \\ 0, & \text{otherwise} \end{cases}. \tag{6}$$

Activation Function. In a BNN, the activation functions like *ReLU* are not used, because the binarized activation values through *ReLU* will all become 1. One usually uses *Hardtanh* instead.

Except for the above parts, the mechanism of BNN is the same as that of general DNN. Therefore, other parts will not be expanded here.

3 Hyperbolic Binary Neural Network

3.1 The Poincaré Ball

The hyperbolic space has several isometric models (Anderson, 2006). According to our discussion in Section 1, the binarized parameter vector is always on the ball with a fixed radius. Therefore, we choose the Poincaré ball model followed by (Nickel & Kiela, 2017; Ganea et al., 2018b).

We denote an n-dimensional Poincaré ball with radius $\frac{1}{\sqrt{r}}$ as $\mathbb{D}_r^n := \{x \in \mathbb{R}^n \mid r\|x\|^2 < 1\}$. And the equipped hyperbolic metric is:

$$g_x^H = \lambda_x^2 g^E, \quad \text{where } \lambda_x := \frac{2}{1 - r\|x\|^2}. \tag{7}$$

In particular, g^E is the Euclidean metric, i.e., the identity matrix. For $r > 0$, $\mathbb{D}_r^n$ denotes the open ball (Poincaré ball). While the radius r equals to zero, the Poincaré ball $\mathbb{D}_r^n$ recovers the Euclidean space, .i.e., $\mathbb{D}_0^n = \mathbb{R}^n$. Similarly, we can denote an n-dimensional sphere with radius $\frac{1}{\sqrt{r}}$ as $\mathbb{S}_r^n := \{x \in \mathbb{R}^n \mid r\|x\|^2 = 1\}$ that is expressed by the boundary of the Poincaré ball $\partial\mathbb{D}_r^n$.

3.2 Parameterized Representation for the Parameters

In order to incorporate the hyperbolic space into the unconstrained optimization problem of neural network binarization, we consider the original problem based on Eq.(3):

$$\text{Original problem: } \min_{\mathbf{w} \in \mathbb{D}_r^n} \mathcal{L}(\mathbf{w}; \mathcal{D}) \qquad \text{Unconstrained problem: } \min_{\tilde{\mathbf{w}} \in \mathbb{R}^n} \mathcal{L}(\phi(\tilde{\mathbf{w}}); \mathcal{D}). \tag{8}$$

In particular, we define the Riemannian manifold as the Poincaré ball $\mathcal{M} := \mathbb{D}_r^n$, then the Riemannian exponential map can be denoted as $\phi : T_p\mathbb{D}_r^n (\cong \mathbb{R}^n) \to \mathbb{D}_r^n$. And we transform $\tilde{\mathbf{w}}$ by the exponential map and express $\phi(\tilde{\mathbf{w}})$ as the parameterized representation.

Lemma 1 *Given a vector $v \in T_p\mathbb{D}_r^n(\cong \mathbb{R}^n)\backslash\{\boldsymbol{0}\}$ and a point $p \in \mathbb{D}_r^n$, the exponential map ϕ : $T_p\mathbb{D}_r^n(\cong \mathbb{R}^n) \to \mathbb{D}_r^n$ can be written in the Poincaré ball with the radius $\frac{1}{\sqrt{r}}$ as:*

$$\phi_p(v) := p \oplus \left(\tanh\left(\sqrt{r}\frac{\lambda_p\|v\|}{2}\right)\frac{v}{\sqrt{r}\|v\|}\right), \quad \forall r \in \mathbb{R}^+, p \in \mathbb{D}_r^n, v \in T_p\mathbb{D}_r^n(\cong \mathbb{R}^n)\backslash\{\boldsymbol{0}\}. \tag{9}$$

Proof. The proofs can be found in (Ganea et al., 2018b). □

Geometrically, the exponential map starts from a point p and takes v as the initial tangent vector on the geodesic, which satisfies that the geodesic distance from the mapped point $\phi(v)$ to the point p is $\|v\|_g$. Then the exponential map is the result of adding p to $\|v\|_g$ as shown in Figure 1. Note that the notation $\oplus$ used here is the addition formalism for hyperbolic geometry instead of traditional Euclidean geometry. We can express the non-associative algebra for hyperbolic geometry in the framework of gyrovector spaces (Ungar, 2001; 2008).

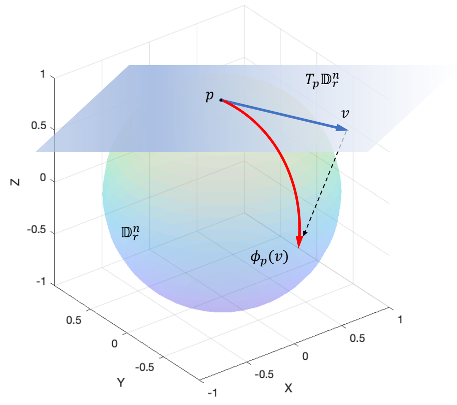


Figure 1: The exponential map from the tangent space $T_p\mathbb{D}_r^n$ to the Poincaré ball $\mathbb{D}_r^n$.

Since the exponential map ϕ_p is limited by p, the parameter vector $\mathbf{w} = \phi_p(\tilde{\mathbf{w}})$ can not be fully represented. Consequently, we introduce the dynamic exponential map to obtain a more fine-gained representation by iterating a series of $p_1, p_2, \cdots, p_t$ while learning $\tilde{\mathbf{w}}$. The entire process of the parameterized representation of HBNN with the dynamic exponential map is shown in Figure 2.

Definition 1 *In the Poincaré ball $\mathbb{D}_r^n$, the addition of two points p and q is defined as:*

$$p \oplus q := \frac{\left(1 + 2r\langle p, q\rangle + r\|q\|^2\right)p + \left(1 - r\|p\|^2\right)q}{1 + 2r\langle p, q\rangle + r^2\|p\|^2\|q\|^2}, \quad \forall r \in \mathbb{R}^+, p, q \in \mathbb{D}_r^n. \tag{10}$$

Based on the above preparations, we can give the unconstrained problem in the HBNN unifying Eq.(4) and Eq.(8) as:

$$\begin{aligned} &\min_{\tilde{\mathbf{w}}\in\mathbb{R}^n, p\in\mathbb{D}_r^n} \mathcal{L}(\text{sign}\left(\phi_p(\tilde{\mathbf{w}})\right); \mathcal{D}) \\ =&\min_{\tilde{\mathbf{w}}\in\mathbb{R}^n, p\in\mathbb{D}_r^n} \mathcal{L}\left(\text{sign}\left(p \oplus \left(\tanh\left(\sqrt{r}\frac{\lambda_p\|\tilde{\mathbf{w}}\|}{2}\right)\frac{\tilde{\mathbf{w}}}{\sqrt{r}\|\tilde{\mathbf{w}}\|}\right)\right); \mathcal{D}\right). \end{aligned} \tag{11}$$

In this way, this unconstrained problem is the multi-objective optimization. For the point p, it is constrained on the Poincaré ball, and further constrains the dynamic exponential map. However, for the parameter vector $\tilde{\mathbf{w}}$, it is unconstrained. Given the radius r, our HBNN requires two variables p and $\tilde{\mathbf{w}}$ to be set and updated during the back-propagation process.

3.3 Backward Mode and Gradient Computation

In order to fully achieve our HBNN in the deep learning framework, we must efficiently compute gradients for the problem of Eq.(11). Intuitively, we can first utilize the inverse of the exponential map to represent $\mathbf{w}$ back to $\tilde{\mathbf{w}}$. Then we update $\tilde{\mathbf{w}}$ in the Euclidean space and p in the Poincaré ball, respectively. We can denote the logarithmic map as $\phi^{-1}: \mathbb{D}_r^n \to T_p\mathbb{D}_r^n(\cong \mathbb{R}^n)$.

Lemma 2 *Given two points p and q ($p \neq q$), the logarithmic map $\phi^{-1}: \mathbb{D}_r^n \to T_p\mathbb{D}_r^n(\cong \mathbb{R}^n)$ can be written in the Poincaré ball with the radius $\frac{1}{\sqrt{r}}$ as:*

$$\phi_p^{-1}(q) := \frac{2}{\sqrt{r}\lambda_p}\tanh^{-1}\left(\sqrt{r}\,\|-p \oplus q\|\right)\frac{-p \oplus q}{\|-p \oplus q\|}, \quad \forall r \in \mathbb{R}^+, p, q \in \mathbb{D}_r^n(p \neq q). \tag{12}$$

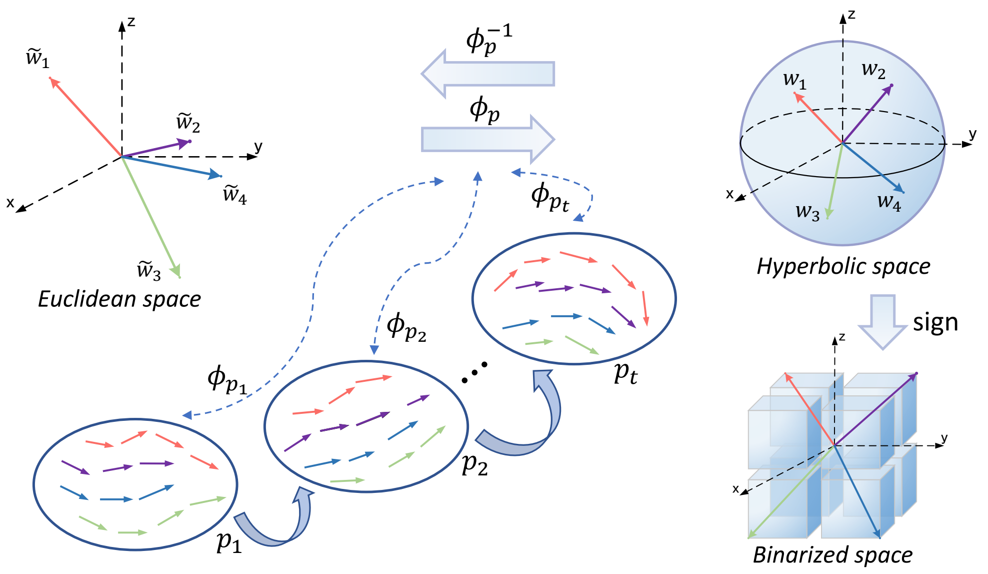


Figure 2: The parameterized representation of HBNN with the dynamic exponential map that is iterated via a series of $p_1, p_2, \cdots, p_t$. The weight of Euclidean space is first transformed into hyperbolic space through the dynamic exponential map, and then transformed into binarized space.

Proof. The proofs can be found in (Ganea et al., 2018b). □

In particular, we can check the algebraic identity $\phi_p^{-1}(\phi_p(v)) = v$ or $\phi_p(\phi_p^{-1}(q)) = q$ satisfying the closed-formula between the exponential and logarithmic map.

Recall that the Straight-Through Estimator $\partial\mathcal{L}/\partial\mathbf{w} = \partial\mathcal{L}/\partial\,\text{sign}(\mathbf{w})$ holds when $|\mathbf{w}| \le 1$ is satisfied based on Eq.(5). Specifically, the parameter vector $\mathbf{w} := \phi_p(\tilde{\mathbf{w}}) \in \mathbb{D}_r^n$ naturally satisfies the constraint of $\|\mathbf{w}\| < \frac{1}{\sqrt{r}}$. If we change the bounds of Straight-Through Estimator a little bit $(1 \to \frac{1}{\sqrt{r}})$, then we can use $\partial\mathcal{L}/\partial\mathbf{w} = \partial\mathcal{L}/\partial\,\text{sign}(\mathbf{w})$ directly, which is always guaranteed to hold.

Following the Straight-Through Estimator, we can compute the gradients in our HBNN:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \frac{\partial\mathcal{L}}{\partial\phi_p(\tilde{\mathbf{w}})} = \frac{\partial\mathcal{L}}{\partial\,\text{sign}(\phi_p(\tilde{\mathbf{w}}))} = \frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{w}}}\cdot\frac{1}{\phi_p'(\tilde{\mathbf{w}})} = \frac{\partial\mathcal{L}}{\partial\phi_p^{-1}(\mathbf{w})}\cdot\frac{1}{\phi_p'(\phi_p^{-1}(\mathbf{w}))}. \tag{13}$$

For a learning rate $\eta > 0$, the update rule of the parameter vector in our HBNN can be written as

$$\tilde{\mathbf{w}}_t \leftarrow \tilde{\mathbf{w}}_{t-1} - \eta\frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{w}}_{t-1}} \quad \mathbf{w}_t \leftarrow \mathbf{w}_{t-1} \oplus -\eta \otimes \left(\frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{w}}_{t-1}}\cdot\frac{1}{\phi_p'(\phi_p^{-1}(\mathbf{w}_{t-1}))}\right). \tag{14}$$

And $\mathbf{w}$ can also be written directly in the form of $\tilde{\mathbf{w}}$:

$$\mathbf{w}_t \leftarrow \phi_p\left(\tilde{\mathbf{w}}_{t-1} - \eta\frac{\partial\mathcal{L}}{\partial\tilde{\mathbf{w}}_{t-1}}\right). \tag{15}$$

Similarly, the update rule of the point p can be written as

$$p_t \leftarrow p_{t-1} \oplus -\eta \otimes \frac{\partial\mathcal{L}}{\partial p_{t-1}}, \tag{16}$$

where the notation $\otimes$ is denoted as the multiplication formalism for hyperbolic geometry.

Definition 2 *In the Poincaré ball $\mathbb{D}_r^n$, the scalar multiplication of the point $p \in \mathbb{D}_r^n \backslash \{\mathbf{0}\}$ by $c \in \mathbb{R}$ is defined as:*

$$c \otimes p := (1/\sqrt{r})\tanh\left(c\tanh^{-1}(\sqrt{r}\|p\|)\right)\frac{p}{\|p\|}, \quad \forall r \in \mathbb{R}^+, c \in \mathbb{R}, p \in \mathbb{D}_r^n\backslash\{\mathbf{0}\}. \tag{17}$$

4 Method Analysis

4.1 Theoretical Analysis

According to the Gauss' lemma (Petersen, 2016), we know that the exponential map changes the metric on the Poincaré ball around the point p into a new one with the square of the distance near p. By summarizing Section 3, we realize that the constrained optimization problem $\min_{\mathbf{w}\in\mathbb{D}_r^n} \mathcal{L}(\operatorname{sign}(\mathbf{w})\,;\mathcal{D})$ via Euclidean descent accounts for the unconstrained one $\min_{\tilde{\mathbf{w}}\in\mathbb{R}^n} \mathcal{L}(\operatorname{sign}(\phi(\tilde{\mathbf{w}}))\,;\mathcal{D})$ through Riemannian gradient descent, which is equipped with a metric on $\mathbb{D}_r^n$ induced by ϕ. In particular, the parameterized representation of our HBNN using ϕ will not add any saddle points or local minima when $\phi : T_p\mathbb{D}_r^n(\cong \mathbb{R}^n) \to \mathbb{D}_r^n$ is a diffeomorphism.

Theorem 1 *Given a connected and complete Riemannian manifold $(\mathcal{M}, g)$ and a point $p \in \mathcal{M}$, the exponential map ϕ with respect to the largest convex open neighborhood of zero $\mathcal{X}_p \subseteq T_p\mathcal{M}$ is a diffeomorphism.*

Proof. The proofs can be found in(Anderson, 2006). □

Combined with Theorem 1, it seems that ϕ is indeed a diffeomorphism in the Poincaré ball $\mathbb{D}_r^n$. However, the exponential map ϕ stops being a diffeomorphism on the boundary $\partial\mathbb{D}_r^n$.

Theorem 2 *For the segment domain seg_p defined by*

$$\operatorname{seg}_p = \{v \in T_p\mathbb{D}_r^n \mid \phi_p(tv) : [0,1] \to \mathbb{D}_r^n \textit{ is a segment }\},$$

the exponential map ϕ is not a diffeomorphism on the open neighborhood $V \in T_p\mathbb{D}_r^n$ containing a point p in the cut locus $\mathcal{C} := \phi_p(\operatorname{seg}_p \backslash \mathcal{X}_p)$ where $\mathcal{X}_p \subseteq T_p\mathbb{D}_r^n$ is the largest radially convex open neighborhood of zero.

Proof. According to the Hopf-Rinow thorem (Spiegel, 2016), we have $\mathbb{D}_r^n = \phi_p(\operatorname{seg}_p)$. The rest of the proof can refer to (Petersen, 2016), which gives the case of general manifolds. Obviously, the case of the Poincaré ball is also fully applicable. □

Actually, for the Poincaré ball, Theorem 2 gives a more vivid conclusion that the set $\mathcal{C}$ at a point p is the boundary of the Poincaré ball. Therefore, the parameterized representation via the exponential map can add saddle points or local minima at these points on the boundary $\partial\mathbb{D}_r^n$.

4.2 Method Comparison and Explanation

HBNN *vs.* **BNN.** The improvement of our HBNN over the traditional BNN can be primarily attributed to the parameterized representation via the dynamic exponential map. On the other hand, this brings additional computational overhead to the training process. Based on Eq.(14) and Eq.(16), we consider updating both $\tilde{\mathbf{w}}$ and p, which doubles the trainable parameters at once. However, our HBNN is not different from the BNN in the inference phase, because both $\tilde{\mathbf{w}}$ and p are parameterized to $\operatorname{sign}(\mathbf{w}) = \operatorname{sign}(\phi_p(\tilde{\mathbf{w}}))$, which keeps the parameter amount constant.

HBNN *vs.* **Mirror Descent.** MD (Ajanthan et al., 2021) provided the mirror descent framework mapping the variables from the unconstrained space to the quantized one. However, MD regards neural network binarization as a single-objective optimization problem, while HBNN treats neural network binarization as a multi-objective one. Consequently, the mirror map of MD can only be set artificially, while the dynamic exponential map of our HBNN can be optimized via the derivative of loss function with respect to p. In essence, we provide the new framework mapping the variables from the unconstrained space to the Riemannian manifold. And HBNN also directly benefits from the properties of hyperbolic spaces. As for the reason why the hyperbolic space is chosen, it is the binarized parameter vector fits the hyperbolic space very well (refer to the discussion of Section 1 for details). On the other hand, MD is not fully binarized until the end of training. Specifically, MD is more like relaxed quantization during the back-propagation. As a contrast, HBNN invariably remains binary during either backward or forward pass.

Table 1: Top-1 classification accuracy results on CIFAR100 with ResNet18 w.r.t. different radii.

Parametr space ($\mathbb{D}_r^n$)		**Parametr space** ($\partial\mathbb{D}_r^n$)	
Radius (r)	mean $\pm$ std (%)	Radius (r)	mean $\pm$ std (%)
0.01	69.34 $\pm$ 0.15	0.01	69.24 $\pm$ 0.10
0.05	**69.50** $\pm$ **0.10**	0.05	69.31 $\pm$ 0.37
0.10	69.45 $\pm$ 0.09	0.10	68.96 $\pm$ 0.27
0.50	69.33 $\pm$ 0.19	0.50	69.16 $\pm$ 0.09
1.00	69.19 $\pm$ 0.21	**1.00**	**69.47** $\pm$ **0.11**
5.00	68.84 $\pm$ 0.33	5.00	69.01 $\pm$ 0.17

5 Related Work

Optimization on Manifolds. Most optimization methods on manifolds have analogs in the Riemannian form (Absil et al., 2009), which can be roughly divided into two categories: following geodesics (Zhang & Sra, 2016) and following first-order approximations to geodesics (Lezcano Casado, 2019). Our proposed method utilizes the Riemannian exponential map that maps the parameter vector onto geodesics and clearly belongs to the first category.

Parametrizations of BNNs. As the sign function used in BNNs is non-differentiable, some methods, including DoReFa (Zhou et al., 2016) and MD (Ajanthan et al., 2021), use the similar differentiable function (*tanh*) for representation in training. Other methods (such as ReCU (Xu et al., 2021b)) use the weight normalization (Salimans & Kingma, 2016; Huang et al., 2017) to achieve the reparameterized representation of BNNs.

Hyperbolic Embeddings. In many machine learning fields (Sala et al., 2018; Ganea et al., 2018b), hyperbolic embeddings outperformed Euclidean embeddings, which is attributed to the fact that the hyperbolic space provides more powerful and meaningful geometrical representations than the Euclidean space (Ganea et al., 2018a). For example in a tree structure, the Euclidean space with infinite dimensions can not be embedded with arbitrary low distortion while the hyperbolic space with only 2 dimensions can achieve this goal (Sala et al., 2018).

6 Experiments

In this section, we design experiments to compare our HBNN trained from scratch with existing state-of-the-art methods in classification tasks. Furthermore, we evaluate the performance of the proposed method via CIFAR (Krizhevsky et al., 2009) and ImageNet (Krizhevsky et al., 2012) datasets. All experiments are implemented on NVIDIA 3090Ti based on the framework of PyTorch. See Appendix A, B and C for more details about compatibility, algorithm and visualization of HBNN.

Experimental Setup. For CIFAR datasets, our HBNNs are totally trained for 600 epochs with a batch size of 256. We adopt SGD optimizer with momentum of 0.9 and a weight decay of 5e-4. For ImageNet dataset, our HBNN is totally trained for 250 epochs with a batch size of 512. We adopt SGD optimizer with momentum of 0.9 and a weight decay of 1e-4. In particular, we use an initial learning rate of 0.1 and the cosine learning rate scheduler in CIFAR10/CIFAR100 and ImageNet.

6.1 Ablation Study

CIFAR datasets. There are two CIFAR benchmarks consisting of natural color images with 32x32 pixels, respectively, 50k training and 10k test images. CIFAR10 consists of images organized into 10 classes and CIFAR100 into 100 classes. We adopt a standard data augmentation scheme (symmetric padding, random clipping and random flipping) that is widely used (Wang et al., 2021). We normalize the images with the means of the channel and standard deviations in preprocessing.

Table 2: Top-1 classification accuracy results on CIFAR10 and CIFAR100 datasets with ResNet18 and VGGsmall. W/A denotes the bit-width of weights/activations.

Model	Method	Bit-width (W/A)	Acc.(%) (CIFAR10)	Acc.(%) (CIFAR100)
ResNet18	Full-precision	32/32	94.8	77.0
	IR-Net (Qin et al., 2020)	1/1	91.5	64.5
	RBNN (Lin et al., 2020b)	1/1	92.2	65.3
	IR-Net+CMIM (Shang et al., 2022)	1/1	92.2	71.2
	ReCU (Xu et al., 2021b)	1/1	92.8	-
	HBNN ($\partial\mathbb{D}_1$)	1/1	92.8	71.2
	HBNN ($\mathbb{D}_{0.05}$)	1/1	**93.0**	**71.6**
	BC (Courbariaux et al., 2015)	1/32	91.6	72.1
	MD-softmax-s (Ajanthan et al., 2021)	1/32	93.3	72.2
	HBNN ($\mathbb{D}_{0.05}$)	1/32	**94.8**	**74.8**
VGGsmall	Full-precision	32/32	94.1	75.5
	XNOR (Rastegari et al., 2016)	1/1	89.8	-
	DoReFa (Zhou et al., 2016)	1/1	90.2	-
	RAD (Ding et al., 2019a)	1/1	90.5	-
	Proxy-BNN (He et al., 2020)	1/1	91.8	67.2
	RBNN (Lin et al., 2020b)	1/1	91.3	67.4
	DSQ (Gong et al., 2019)	1/1	91.7	-
	SLB (Yang et al., 2020)	1/1	92.0	-
	ReCU (Xu et al., 2021b)	1/1	92.2	-
	RBNN+CMIM (Shang et al., 2022)	1/1	92.2	71.0
	HBNN ($\partial\mathbb{D}_1$)	1/1	92.8	72.2
	HBNN ($\mathbb{D}_{0.05}$)	1/1	**93.0**	**72.5**

We first conduct a series of ablation studies of HBNN in CIFAR100 with the ResNet18 model. Based on the two parameter spaces (Poincaré ball $\mathbb{D}_r^n$ and the boundary of Poincaré ball $\partial\mathbb{D}_r^n$), we adjust the different radii to determine the optimal radius using the classification accuracies at epoch 120. The mean top-1 accuracies (mean $\pm$ std) are reported in Table 1. Consequently, we have $r = 0.05$ for the parameter space $\mathbb{D}_r^n$ and $r = 1$ for the parameter space $\partial\mathbb{D}_r^n$, which will be used in the following experiments. Although we choose the radius like this, the effect of the radius is very small by considering the choice of random seeds. It seems that our HBNN has great robustness.

6.2 Experimental Results

For ResNet18, we compare with IR-Net (Qin et al., 2020), RBNN (Lin et al., 2020b), IR-Net+CMIM (Shang et al., 2022), ReCU (Xu et al., 2021b), BC (Courbariaux et al., 2015) and MD-softmax-s (Ajanthan et al., 2021). For VGGsmall, our HBNN is compared with XNOR (Rastegari et al., 2016), DoReFa (Zhou et al., 2016), RAD (Ding et al., 2019a), Proxy-BNN (He et al., 2020), DSQ (Gong et al., 2019) and SLB (Yang et al., 2020), etc.

As shown in Table 2, HBNN always outperforms the existing state-of-the-art methods. In particular, our HBNN (1-bit weights and 1-bit activations) achieves over 1.5% performance improvement with VGGsmall architecture on CIFAR100 dataset, which even exceeds the methods of 1-bit weights and 32-bit activations with ResNet18 architecture.

For the 1/32 case, the training of HBNN on the boundary $\partial\mathbb{D}_r$ is unstable, which may be attributed to the introduction of extra saddle points or local minima that causes the HBNN to get stuck in them.

Table 3: Top-1 and Top-5 classification accuracy results on ImageNet dataset with ResNet18 and ResNet34. W/A denotes the bit-width of weights/activations.

Model	Method	Bit-width (W/A)	Acc.(%) (Top-1)	Acc.(%) (Top-5)
ResNet18	Full-precision	32/32	69.6	89.2
	ABC-Net (Lin et al., 2017)	1/1	42.7	67.6
	XNOR (Rastegari et al., 2016)	1/1	51.2	73.2
	BiReal (Liu et al., 2020)	1/1	56.4	79.5
	IR-Net (Qin et al., 2020)	1/1	58.1	80.0
	RBNN (Lin et al., 2020b)	1/1	59.9	81.9
	FDA-BNN (Xu et al., 2021a)	1/1	60.2	82.3
	ReCU (Xu et al., 2021b)	1/1	61.0	82.6
	RBNN+CMIM (Shang et al., 2022)	1/1	61.2	82.2
	HBNN ($\partial\mathbb{D}_1$)	1/1	**61.5**	**83.3**
	HBNN ($\mathbb{D}_{0.05}$)	1/1	**61.5**	**83.3**
ResNet34	Full-precision	32/32	73.3	91.3
	XNOR++ (Bulat & Tzimiropoulos, 2019)	1/1	57.1	79.9
	LNS (Han et al., 2020)	1/1	59.4	81.7
	BiReal (Liu et al., 2020)	1/1	62.2	83.9
	IR-Net (Qin et al., 2020)	1/1	62.9	84.1
	RBNN (Lin et al., 2020b)	1/1	63.1	84.4
	RBNN+CMIM (Shang et al., 2022)	1/1	65.0	85.7
	ReCU (Xu et al., 2021b)	1/1	65.1	85.8
	HBNN ($\partial\mathbb{D}_1$)	1/1	65.6	86.0
	HBNN ($\mathbb{D}_{0.05}$)	1/1	**65.7**	**86.2**

ImageNet dataset. The ImageNet benchmark consists of 1.2 million high-resolution natural images, where the validation set contains 50k images. These images are organized into 1000 categories of objects for training, which are resized to 224x224 pixels before fed into the network. We follow the standard data augmentation strategies, including random clips and horizontal flips (Wang et al., 2021). Then we report our single-crop evaluation results using Top-1 and Top-5 accuracies.

For ResNet18, we compare with ABC-Net (Lin et al., 2017), XNOR (Rastegari et al., 2016), BiReal (Liu et al., 2020) and FDA-BNN (Xu et al., 2021a), etc. For ResNet34, our HBNN is compared with XNOR++ (Bulat & Tzimiropoulos, 2019) and LNS (Han et al., 2020), etc.

As shown in Table 3, HBNN continues to exceed the existing state-of-the-art methods in both top-1 and top-5 accuracies. Specifically, our proposed method achieves 0.6% Top-1 accuracy improvement with ResNet34 architecture compared with ReCU method.

7 Conclusion

In this paper, we have introduced the Riemannian manifold framework for neural network binarization via the dynamic exponential map that is learned with the training iteration. Specifically, we proposed a Hyperbolic Binary Neural Network (HBNN) by converting a constrained optimization problem in the Poincaré ball into an unconstrained one in the Euclidean space. By analyzing the dynamic exponential map, we present that HBNN will not create extra saddle points or local minima in the Poincaré ball. In the future, we will intend to focus more on the dynamic representation and optimization of neural networks in the geometrical aspects.

References

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. Optimization algorithms on matrix manifolds. In *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

Thalaiyasingam Ajanthan, Kartik Gupta, Philip Torr, Richad Hartley, and Puneet Dokania. Mirror descent view for neural network quantization. In *International Conference on Artificial Intelligence and Statistics*, pp. 2809–2817. PMLR, 2021.

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

Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. Scalable methods for 8-bit training of neural networks. *Advances in neural information processing systems*, 31, 2018.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019.

Jun Chen, Liang Liu, Yong Liu, and Xianfang Zeng. A learning framework for n-bit quantized neural networks toward fpgas. *IEEE Transactions on Neural Networks and Learning Systems*, 32 (3):1067–1081, 2021. doi: 10.1109/TNNLS.2020.2980041.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Ruizhou Ding, Ting-Wu Chin, Zeye Liu, and Diana Marculescu. Regularizing activation distribution for training binarized deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11408–11417, 2019a.

Xiaohan Ding, Xiangxin Zhou, Yuchen Guo, Jungong Han, Ji Liu, et al. Global sparse momentum sgd for pruning very deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019b.

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

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

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4852–4861, 2019.

Heinrich W Guggenheimer. *Differential geometry*. Courier Corporation, 2012.

Kai Han, Yunhe Wang, Yixing Xu, Chunjing Xu, Enhua Wu, and Chang Xu. Training binary neural networks through learning with noisy supervision. In *International Conference on Machine Learning*, pp. 4017–4026. PMLR, 2020.

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.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Xiangyu He, Zitao Mo, Ke Cheng, Weixiang Xu, Qinghao Hu, Peisong Wang, Qingshan Liu, and Jian Cheng. Proxybnn: Learning binarized neural networks via proxy matrices. In *European Conference on Computer Vision*, pp. 223–241. Springer, 2020.

Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. *Advances in neural information processing systems*, 32, 2019.

Geoffrey Hinton, Nitsh Srivastava, and Kevin Swersky. Neural networks for machine learning. *Coursera, video lectures*, 264(1):2146–2153, 2012.

Lei Huang, Xianglong Liu, Yang Liu, Bo Lang, and Dacheng Tao. Centered weight normalization in accelerating training of deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2803–2811, 2017.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Mario Lezcano Casado. Trivializations for gradient-based optimization on manifolds. *Advances in Neural Information Processing Systems*, 32, 2019.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1529–1538, 2020a.

Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. *Advances in neural information processing systems*, 33:7474–7485, 2020b.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. *Advances in neural information processing systems*, 30, 2017.

Zechun Liu, Wenhan Luo, Baoyuan Wu, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Binarizing deep network towards real-network performance. *International Journal of Computer Vision*, 128(1):202–219, 2020.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

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

Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pp. 1520–1528, 2015.

Peter Petersen. *Riemannian geometry*, volume 171. Springer, 2006.

Peter Petersen. Riemannian metrics. In *Riemannian Geometry*, pp. 1–39. Springer, 2016.

Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2250–2259, 2020.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

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.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Yuzhang Shang, Dan Xu, Ziliang Zong, and Yan Yan. Network binarization via contrastive learning. *arXiv preprint arXiv:2207.02970*, 2022.

Daniel Spiegel. The hopf-rinow theorem. *Notes available online*, 2016.

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

Wenxiao Wang, Minghao Chen, Shuai Zhao, Long Chen, Jinming Hu, Haifeng Liu, Deng Cai, Xiaofei He, and Wei Liu. Accelerate cnns from three dimensions: a comprehensive pruning framework. In *International Conference on Machine Learning*, pp. 10717–10726. PMLR, 2021.

Yixing Xu, Kai Han, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Learning frequency domain approximation for binary neural networks. *Advances in Neural Information Processing Systems*, 34:25553–25565, 2021a.

Zihan Xu, Mingbao Lin, Jianzhuang Liu, Jie Chen, Ling Shao, Yue Gao, Yonghong Tian, and Rongrong Ji. Recu: Reviving the dead weights in binary neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5198–5208, 2021b.

Zhaohui Yang, Yunhe Wang, Kai Han, Chunjing Xu, Chao Xu, Dacheng Tao, and Chang Xu. Searching for low-bit weights in quantized neural networks. *Advances in neural information processing systems*, 33:4091–4102, 2020.

Hongyi Zhang and Suvrit Sra. First-order methods for geodesically convex optimization. In *Conference on Learning Theory*, pp. 1617–1638. PMLR, 2016.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

A Compatibility

In this section, we further evaluate the compatibility of HBNN. And we plug HBNN into IR-Net and ReCU as a plug-and-play module, as shown in Table 4 and 5. By plugging our HBNN, the corresponding methods both have improved the accuracies.

Table 4: Top-1 classification accuracy results on CIFAR10 dataset with ResNet18 and VGGsmall. W/A denotes the bit-width of weights/activations.

Model	Method	Bit-width (W/A)	Acc.(%) (CIFAR10)
ResNet18	Full-precision	32/32	94.8
	IR-Net (Qin et al., 2020)	1/1	91.5
	IR-Net+HBNN	1/1	91.9
	ReCU (Xu et al., 2021b)	1/1	92.8
	ReCU+HBNN	1/1	92.8
VGGsmall	Full-precision	32/32	94.1
	IR-Net (Qin et al., 2020)	1/1	90.4
	IR-Net+HBNN	1/1	92.4
	ReCU (Xu et al., 2021b)	1/1	92.2
	ReCU+HBNN	1/1	92.9

Table 5: Top-1 and Top-5 classification accuracy results on ImageNet dataset with ResNet18 and ResNet34. W/A denotes the bit-width of weights/activations.

Model	Method	Bit-width (W/A)	Acc.(%) (Top-1)	Acc.(%) (Top-5)
ResNet18	Full-precision	32/32	69.6	89.2
	IR-Net (Qin et al., 2020)	1/1	58.1	80.0
	IR-Net+HBNN	1/1	60.9	82.9
	ReCU (Xu et al., 2021b)	1/1	61.0	82.6
	ReCU+HBNN	1/1	61.5	83.1
ResNet34	Full-precision	32/32	73.3	91.3
	IR-Net (Qin et al., 2020)	1/1	62.9	84.1
	IR-Net+HBNN	1/1	64.2	85.2
	ReCU (Xu et al., 2021b)	1/1	65.1	85.8
	ReCU+HBNN	1/1	65.8	86.2

B Algorithm

Algorithm 1 Forward and Backward Propagation of HBNN

```
Require: A minibatch of data samples $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^m$, current binary weight $\mathbf{w}_k^b$, latent full-precision unconstrained weight $\tilde{\mathbf{w}}_k$, latent full-precision constrained weight $\mathbf{w}_k$, the dynamic exponential map $\phi_p$, and a learning rate $\eta$.
Ensure: Update $\tilde{\mathbf{w}}_k$, $\mathbf{w}_k$ and $p$.
 1: {Forward propagation}
 2: for $k = 1$ to $l - 1$ do
 3:     Compute the weight in the hyperbolic space: $\mathbf{w}_k \leftarrow \phi_p(\tilde{\mathbf{w}}_k)$;
 4:     Binarize the weight: $\mathbf{w}_k^b \leftarrow \mathrm{sign}(\mathbf{w}_k)$;
 5:     Binarize the activation: $\mathbf{a}_{k-1}^b \leftarrow \mathrm{sign}(\mathbf{a}_{k-1})$;
 6:     Perform binary operation: $\mathbf{a}_k \leftarrow \mathrm{XnorDotProduct}(\mathbf{w}_k^b, \mathbf{a}_{k-1}^b)$;
 7:     Perform Batch Normalization: $\mathbf{a}_k \leftarrow \mathrm{BatchNorm}(\mathbf{a}_k)$;
 8: end for
 9: Optimize the unconstrained problem with Eq.(11);
10: {Backward propagation}
11: Compute the gradient of the overall loss function, i.e., $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}$, $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{w}}}$ and $\frac{\partial \mathcal{L}}{\partial p}$, where the sign function can be handled in Eq.(5) for the weight and Eq.(6) for activation;
12: {The parameters update}
13: Update the full-precision unconstrained weight: $\tilde{\mathbf{w}}_t \leftarrow \tilde{\mathbf{w}}_{t-1} - \eta \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{w}}_{t-1}}$;
14: Update the dynamic exponential map $\phi_p$ in Eq.(9) with $p$: $p_t \leftarrow p_{t-1} \oplus -\eta \otimes \frac{\partial \mathcal{L}}{\partial p_{t-1}}$;
15: Update the full-precision constrained weight: $\mathbf{w}_t \leftarrow \mathbf{w}_{t-1} \oplus -\eta \otimes \left( \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{w}}_{t-1}} \cdot \frac{1}{\phi'_{p_t}(\phi_{p_t}^{-1}(\mathbf{w}_{t-1}))} \right)$;
```

C Visualization

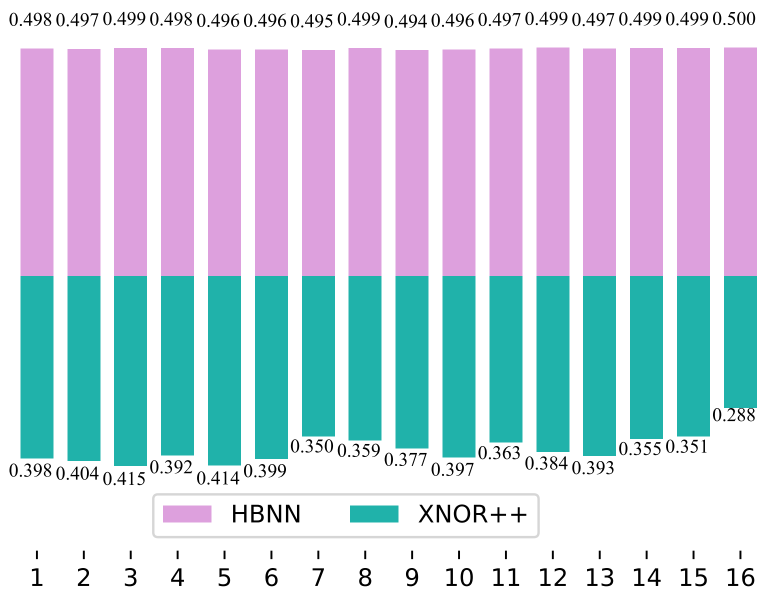
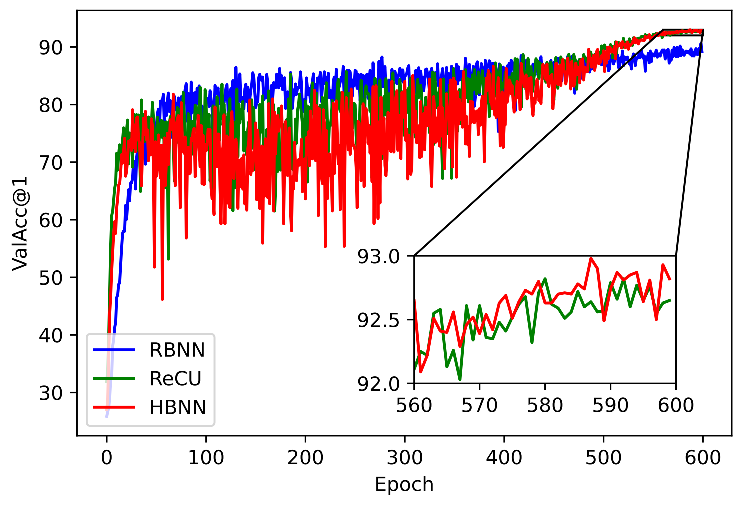


Figure 3: Weight flip rates of our HBNN and XNOR++ in different layers of ResNet18.

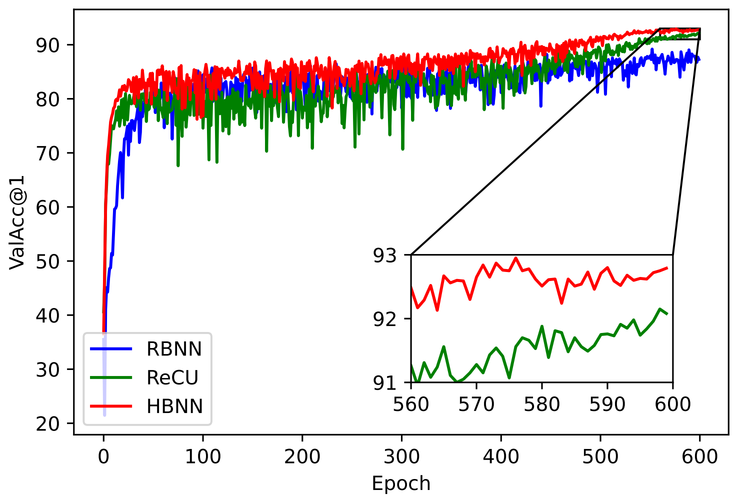
Figure 3 shows the weight flip rates of our HBNN and XNOR++ in different layers of ResNet18 in CIFAR10. By the hyperbolic representation, HBNN leads to around 50% weight flips each layer. It seems that the introduction of hyperbolic geometry expands the expressive ability of BNN compared with XNOR++.

The validation curves with ResNet18 are shown in Figure 4. Compared to RBNN and ReCU on CIFAR10, the validation accuracies of our HBNN show a good and stable convergence followed by the training epoch.

Furthermore, we show 2D visualization of the loss surfaces based on the previous work (Li et al., 2018). By analyzing Figure 5, we find that the loss surface of the full-precision model is smooth and flat, which is beneficial for the neural network to arrive at the global optimum. Our HBNN has a relatively flat loss surface, which can also explain its better performance than XNOR++.

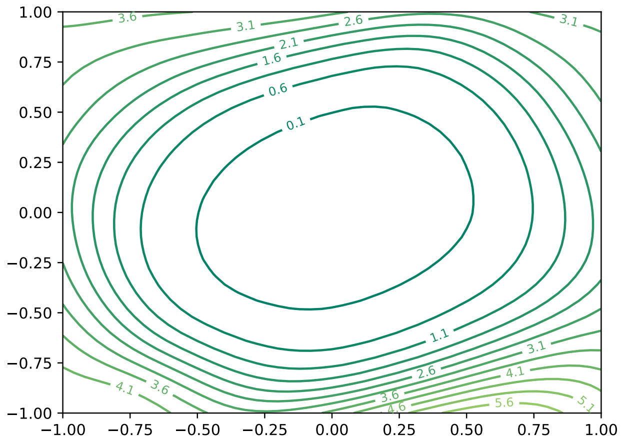


(a) ResNet18

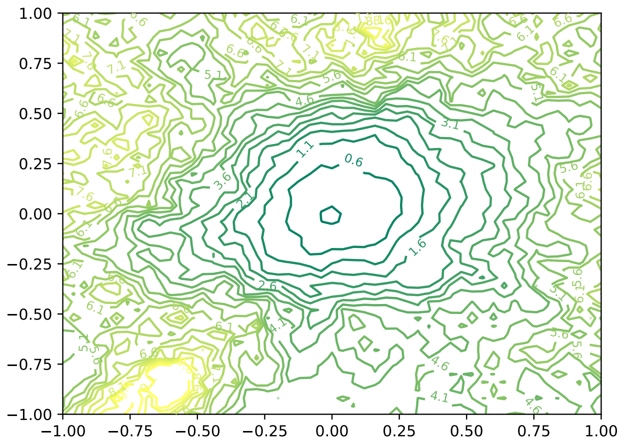


(b) VGGsmall

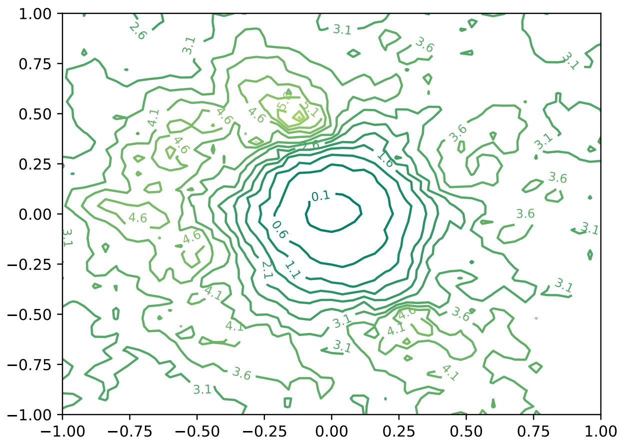
Figure 4: Validation accuracy curves on CIFAR10 dataset.



(a) Full-precision



(b) XNOR++



(c) HBNN

Figure 5: 2D visualization of the loss surfaces of ResNet18 for CIFAR10 dataset, which is used to enable comparisons of sharpness/flatness of different methods.