Hyperbolic Binary Neural Network

Jun Chen, Jingyang Xiang, Tianxin Huang, Xiangrui Zhao, and Yong Liu, *Member, IEEE*

***Abstract*— Binary neural network (BNN) converts full-precision weights and activations into their extreme 1-bit counterparts, making it particularly suitable for deployment on lightweight mobile devices. While BNNs are typically formulated as a constrained optimization problem and optimized in the binarized space, general neural networks are formulated as an unconstrained optimization problem and optimized in the continuous space. This article introduces the hyperbolic BNN (HBNN) by leveraging the framework of hyperbolic geometry to optimize the constrained problem. Specifically, we transform the constrained problem in hyperbolic space into an unconstrained one in Euclidean space using the Riemannian exponential map. On the other hand, we also propose the exponential parametrization cluster (EPC) method, which, compared with the Riemannian exponential map, shrinks the segment domain based on a diffeomorphism. This approach increases the probability of weight flips, thereby maximizing the information gain in BNNs. Experimental results on CIFAR10, CIFAR100, and ImageNet classification datasets with VGGsmall, ResNet18, and ResNet34 models illustrate the superior performance of our HBNN over state-of-the-art methods.**

***Index Terms*— Binary neural network (BNN), deep learning, hyperbolic geometry, model compression.**

I. Introduction

DEEP neural networks (DNNs) have achieved remarkable success in various computer vision fields, including image classification [1], [2], object detection [3], [4], semantic segmentation [5], [6], and more. However, the massive parameters and computational complexity of DNNs, which contribute to their success, limit their deployment on lightweight mobile devices. To address this problem, various compression methods are being proposed, with the main approaches, including pruning [7], [8], [9], quantization [10], [11], [12], [13], [14], and distillation [15].

In the context of resource-constrained and low-power devices, quantization emerges as a more effective and universal scheme compared with pruning [16]. Specifically, quantization converts full-precision weights and activations into their low-precision counterparts. In the extreme case, neural network binarization restricts weights and activations to two possible discrete values $\{-1, +1\}$, offering two advantages as follows: 1) a 32× reduction in memory compared with the corresponding full-precision version and 2) the multiply-accumulation operation can be replaced with the efficient XNOR and bitcount operations.

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

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

where $\mathbf{w}$ is an n dimensional weight vector and $\mathcal{L}$ represents the loss function, such as cross-entropy loss. Using a mirror descent framework [17], Ajanthan et al. [18] transformed the constrained problem into an unconstrained one through a mapping $P : \mathbb{R}^n \to \mathcal{X}$, such that

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

Subsequently, $P(\tilde{\mathbf{w}}) \in \mathcal{X}$ is gradually binarized to a discrete set $\mathcal{B}^n = \{-1, +1\}^n$ during the training process, where P is defined as a mirror map.

In BNNs, the norm of the binarized weight vector in each layer is fixed and is solely determined by the dimension of the weight. In other words, the binarized weight resides on a ball with a constant radius, forming a hyperbolic space. In this article, we introduce a hyperbolic binary neural network (HBNN) to formulate neural network binarization as a optimization problem in the framework of hyperbolic space. Specifically, we transform the constrained problem in hyperbolic space into an unconstrained one in Euclidean space using the Riemannian exponential map. This approach is more conducive to optimizing BNNs than directly converting the optimization problem from the constrained and binarized space to the unconstrained and continuous space.

On the other hand, recent research [19] has demonstrated that a high ratio of weight flips, where weight flips mean that positive values turn to negative values and vice versa, can maximize the information gain to optimize BNNs' performance. In this context, we propose the exponential parametrization cluster (EPC) ($\phi_{\mathcal{F}}(\cdot) : \mathbb{R}^n \to \mathbb{D}_r^n$) shown in Fig. 1. This approach is a differentiable map from the tangent space ($\mathbb{R}^n$) to the hyperbolic space ($\mathbb{D}_r^n$). In this case, the constrained optimization problem in hyperbolic space is transformed into an unconstrained one in Euclidean space

$$\text{Original problem: } \min_{\mathbf{w}\in\mathbb{D}_r^n} \mathcal{L}(\mathbf{w}; \mathcal{D})$$
$$\text{Unconstrained problem: } \min_{\tilde{\mathbf{w}}\in\mathbb{R}^n, \mathcal{F}\in\mathbb{D}_r^n} \mathcal{L}(\phi_{\mathcal{F}}(\tilde{\mathbf{w}}); \mathcal{D}) \quad (1)$$

Received 1 March 2023; revised 17 January 2024 and 9 May 2024; accepted 14 October 2024. Date of publication 31 October 2024; date of current version 4 June 2025. This work was supported by the National Natural Science Foundation of China under Grant 62103363. *(Corresponding authors: Jun Chen; Yong Liu.)*

Jun Chen is with the National Special Education Resource Center for Children with Autism, Zhejiang Normal University, Hangzhou 311231, China, also with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China, and also with the School of Computer Science and Technology, Zhejiang Normal University, Jinhua 321004, China (e-mail: junc@zju.edu.cn).

Jingyang Xiang, Tianxin Huang, Xiangrui Zhao, and Yong Liu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: yongliu@iipc.zju.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2024.3485115

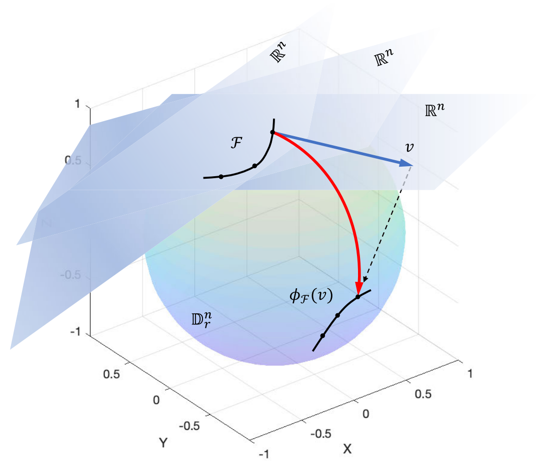


Fig. 1. EPC $\phi_{\mathcal{F}}$ transforms a vector v into the mapped cluster $\phi_{\mathcal{F}}(v)$ using an original cluster $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_t\}$, where $\mathcal{F}$ and $\phi_{\mathcal{F}}(v)$ exist in hyperbolic space, while v resides in Euclidean space. In contrast, the Riemannian exponential map exp transforms a vector v into the mapped point $\exp(v)$.

where the cluster $\mathcal{F}$ consists of a series of candidate points $\{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_t\}$. In comparison with the Riemannian exponential map [20] $\exp(\cdot)$, our proposed EPC extends the mapping result from a single point to a cluster of points. Inherently, the Riemannian exponential map $\exp(\cdot)$ is equivalent to $\phi_{\mathcal{F}_i}(\cdot)$, where $\mathcal{F}_i$ is a candidate point from the cluster $\mathcal{F}$.

The main contributions of this article are summarized in the following three aspects.

1) We propose the HBNN by leveraging the framework of hyperbolic geometry to optimize the constrained problem. Specifically, we transform the constrained problem in hyperbolic space into an unconstrained one in Euclidean space using the Riemannian exponential map.
2) We introduce the EPC, which, compared with the Riemannian exponential map, shrinks the segment domain on the basis of a diffeomorphism. This approach increases the probability of weight flips, maximizing the information gain in BNNs.
3) Experimental results on CIFAR10, CIFAR100, and ImageNet classification datasets with VGGsmall, ResNet18, and ResNet34 models illustrate the superior performance of our HBNN over state-of-the-art methods.

II. Related Work

A. Optimization on Manifolds

Many optimization methods on manifolds have Riemannian analogs [21], [22]. Parametrization is an important technique for converting problems with manifold constraints into unconstrained problems in Euclidean space. Helfrich et al. [23] introduced orthogonal and unitary Cayley parametrizations, which construct orthogonal weight matrices through a scaled Cayley transform in recurrent neural networks. Lezcano-Casado and Martinez-Rubio [24] introduced the orthogonal exponential parametrization derived from Lie group theory using the Riemannian exponential map. Casado [25] further introduced dynamic parametrization as a gradient-based optimization that combines the advantages of the Riemannian exponential and Lie exponential.

B. Binarization Methods

The introduction of the nondifferentiable sign function in neural network binarization leads to a performance drop. For instance, XNOR [26] introduced accurate approximations by binarizing not only the weights but also the intermediate representations in DNNs. This approach aims to reduce the quantization error between the full-precision weights and their binarized counterparts. XNOR++ [27] further fused the activation and weight scaling factors into a single factor, improving overall performance. BiReal [28] addressed the problem of infinite or zero gradients caused by the sign function by propagating full-precision activations through a parameter-free shortcut in each binarized convolution. Proxy-BNN [29] introduced a proxy matrix to serve as the basis for the latent parameter space, aiming to reduce the quantization error of weights and restore the smoothness of BNNs. Recently, IR-Net [30] proposed a balanced and standardized binarization method in the forward pass, minimizing the information loss by maximizing the information entropy of binarized weights and minimizing the quantization error. RBNN [19] analyzed the angle alignment between full-precision weights and their binarized counterparts, highlighting that around 50% weight flips can maximize the information gain. ReCU [31] employed the weight normalization [32], [33] to revive "dead weights," increasing the probability of updating these weights in BNNs.

III. Preliminaries

Here, we provide background knowledge on Riemannian geometry and BNNs.

A. Riemannian Geometry

We briefly introduce the basic concepts of Riemannian geometry, and for more in-depth propositions, see [34] and [20].

1) 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. Also, such a tangent space exists for all points $p \in \mathcal{M}$. Thus, the description of tangent spaces aligns with Euclidean space, denoted as $T_p\mathcal{M} \cong \mathbb{R}^n$.

2) Riemannian Manifold: Riemannian manifolds are endowed with a smooth metric $g_p : T_p\mathcal{M} \times T_p\mathcal{M} \to \mathbb{R}$ that varies smoothly with p, enabling the construction of a distance function $d_g : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$. When describing a Riemannian manifold, the Riemannian metric is inherently equipped by default, denoted as $(\mathcal{M}, g)$.

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

4) Exponential Map: The Riemannian exponential map, denoted as $\exp : T_p\mathcal{M} \to \mathcal{M}$, serves to map rays starting at the origin in the tangent space $T_p\mathcal{M}$ to geodesics on $\mathcal{M}$. For a given geodesic, the parameter t ranges from 0 to 1, resulting in $\exp(tv) := \gamma_{p,v}(t)$. Specifically, the distance on the manifold between a point p and the exponential map $\exp(v)$ is given by $d_g(p, \exp(v)) = \|v\|_g$.

B. Binary Neural Network

Now, let us delve into the mechanism of BNNs and explore how the binarization and gradients are computed.

1) Forward Pass: During the inference phase of a BNN, the binarization function is expressed in a deterministic form [26], [35]

$$x^b = \operatorname{sign}(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{2}$$

where x can represent either weights $\mathbf{w}$ or activations $\mathbf{a}$.

2) Backward Pass: During backpropagation, the gradient suffers from the problem of either infinite or zero when propagating through the binarization function. To address this problem, Hinton et al. [36] and Bengio et al. [37] proposed the straight-through estimator. Consequently, this estimator of the gradient with respect to binarized weights can be approximated as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{\partial \mathcal{L}}{\partial \mathbf{w}^b} \cdot \frac{\partial \mathbf{w}^b}{\partial \mathbf{w}}, \quad \frac{\partial \mathbf{w}^b}{\partial \mathbf{w}} := \begin{cases} 1, & \text{if } |\mathbf{w}| \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

On the other hand, based on the polynomial function [28], an estimator of the gradient with respect to binarized activations can be formulated as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^b} \cdot \frac{\partial \mathbf{a}^b}{\partial \mathbf{a}}, \quad \frac{\partial \mathbf{a}^b}{\partial \mathbf{a}} := \begin{cases} 2 + 2\mathbf{a}, & \text{if } -1 \leq \mathbf{a} < 0 \\ 2 - 2\mathbf{a}, & \text{if } 0 \leq \mathbf{a} \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

3) Activation Function: In a BNN, the activation function, such as ReLU, is avoided, because the binarized activation values through ReLU would all become 1. Typically, *Hardtanh* is applied instead.

IV. Hyperbolic BNN

A. Poincaré Ball

The hyperbolic space has several isometric models [38], which are not only conformal to Euclidean space but also offer powerful and meaningful geometrical representations [39]. We choose the Poincaré ball model, as suggested by the previous works [40], [41]. By denoting an n-dimensional Poincaré ball with radius $1/\sqrt{r}$ as $\mathbb{D}_r^n := \{x \in \mathbb{R}^n \mid r\|x\|^2 < 1\}$, the equipped hyperbolic metric is given by

$$g_x^H = \lambda_x^2 g^E, \quad \text{where } \lambda_x := \frac{2}{1 - r\|x\|^2}. \tag{5}$$

Here, g^E represents the Euclidean metric, i.e., the identity matrix. For $r > 0$, $\mathbb{D}_r^n$ denotes the open ball (Poincaré ball). When the radius r equals 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 $1/\sqrt{r}$ as $\mathbb{S}_r^n := \{x \in \mathbb{R}^n \mid r\|x\|^2 = 1\}$, expressed by the boundary of the Poincaré ball, namely, $\partial\mathbb{D}_r^n$.

B. Exponential Parametrization Cluster

Building upon (1), we aim to transform the constrained optimization problem of binarization in hyperbolic space into an unconstrained optimization problem in Euclidean space. For a weight vector $\tilde{\mathbf{w}}$ in Euclidean space, we can compute its EPC, i.e., $\phi_{\mathcal{F}}(\tilde{\mathbf{w}})$, which is composed of a series of weight vectors $\{\phi_{\mathcal{F}_1}(\tilde{\mathbf{w}}), \phi_{\mathcal{F}_2}(\tilde{\mathbf{w}}), \ldots, \phi_{\mathcal{F}_t}(\tilde{\mathbf{w}})\}$ in hyperbolic space.

Given a weight vector $\tilde{\mathbf{w}} \in T_p\mathbb{D}_r^n(\cong\mathbb{R}^n)\backslash\{\mathbf{0}\}$, where $p \in \mathbb{D}_r^n$, the EPC with a cluster $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_t\} \in \mathbb{D}_r^n$ can be expressed in the Poincaré ball with the radius $1/\sqrt{r}$ as follows:

$$\phi_{\mathcal{F}}(\cdot) := \begin{bmatrix} \mathcal{F}_1 \oplus \left(\tanh\left(\sqrt{r}\frac{\lambda_p\|\cdot\|}{2}\right)\frac{\cdot}{\sqrt{r}\|\cdot\|}\right) \\ \mathcal{F}_2 \oplus \left(\tanh\left(\sqrt{r}\frac{\lambda_p\|\cdot\|}{2}\right)\frac{\cdot}{\sqrt{r}\|\cdot\|}\right) \\ \vdots \\ \mathcal{F}_t \oplus \left(\tanh\left(\sqrt{r}\frac{\lambda_p\|\cdot\|}{2}\right)\frac{\cdot}{\sqrt{r}\|\cdot\|}\right) \end{bmatrix}. \tag{6}$$

Geometrically, the EPC starts with a cluster $\mathcal{F}$ and takes v as the initial tangent vector on the geodesic. This vector satisfies that the geodesic distance from the mapped cluster $\phi_{\mathcal{F}}(v)$ to the original cluster is $\|v\|_g$. It is important to note that the notation $\oplus$ used here follows the addition formalism for hyperbolic geometry, differing from the traditional Euclidean geometry. The nonassociative algebra for hyperbolic geometry can be expressed in the framework of gyrovector spaces [42], [43].

Addition [39]: In the Poincaré ball, the addition of p and q in $\mathbb{D}_r^n$ is defined as follows:

$$p \oplus q := \frac{\left(1 + 2r\langle p, q\rangle + r\|q\|^2\right)p + \left(1 - r\|p\|^2\right)q}{1 + 2r\langle p, q\rangle + r^2\|p\|^2\|q\|^2}. \tag{7}$$

Given that the Riemannian exponential map $\exp(\cdot)$ is constrained by a point, the corresponding representations $\exp(\tilde{\mathbf{w}})$ do not contribute to an increased probability of weight flips. In contrast, our mapped cluster $\phi_{\mathcal{F}}(\tilde{\mathbf{w}})$ provides more candidate representations by training a cluster $\mathcal{F}$, thereby increasing the probability of weight flips. We will theoretically elaborate on the role of the EPC in weight flips in Section V. An overview of our HBNN with the EPC is presented in Fig. 2.

Subsequently, we can formulate the unconstrained problem for the weight vector, unifying (1) and (6), as follows:

$$\min_{\tilde{\mathbf{w}}\in\mathbb{R}^n} \min_{\mathcal{F}\in\mathbb{D}_r^n} \mathcal{L}\left(\{\phi_{\mathcal{F}_1}(\tilde{\mathbf{w}}), \phi_{\mathcal{F}_2}(\tilde{\mathbf{w}}), \ldots, \phi_{\mathcal{F}_t}(\tilde{\mathbf{w}})\}; \mathcal{D}\right). \tag{8}$$

C. Backward Mode and Gradient Computation

In order to fully implement our HBNN in the deep learning framework, it is crucial to efficiently compute gradients for the problem stated in (8). During backpropagation, we first keep the weight vector $\tilde{\mathbf{w}}$ unchanged. Using a learning rate $\eta > 0$,

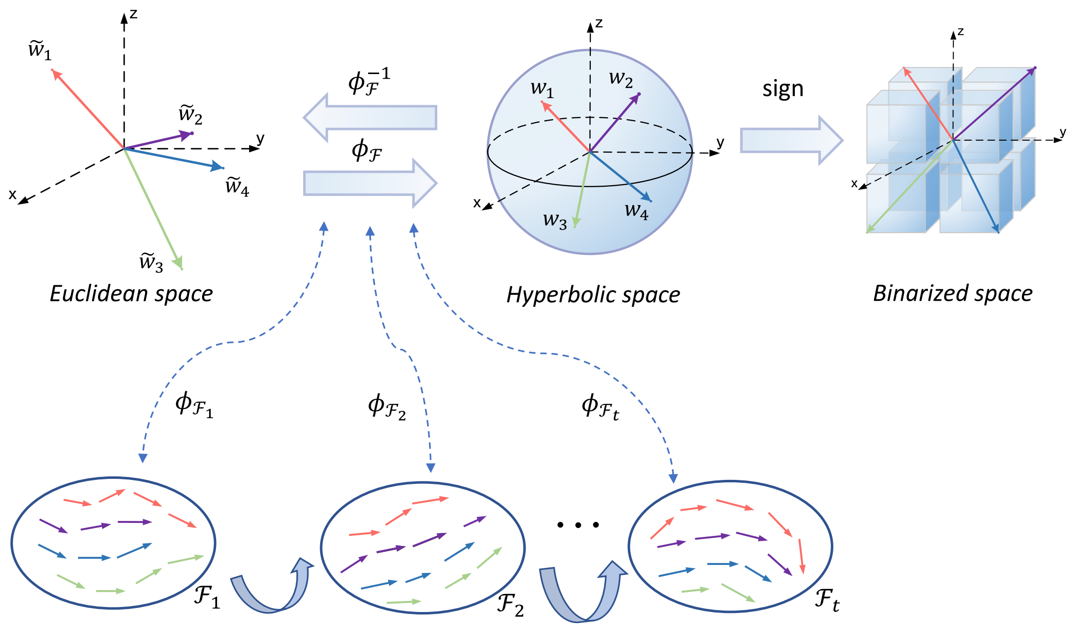


Fig. 2. Overview of our HBNN with the EPC. By training an original cluster $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_t\}$, we map a weight vector $\tilde{\mathbf{w}}$ into the mapped cluster $\phi_{\mathcal{F}}(\tilde{\mathbf{w}}) = \{\phi_{\mathcal{F}_1}(\tilde{\mathbf{w}}), \phi_{\mathcal{F}_2}(\tilde{\mathbf{w}}), \ldots, \phi_{\mathcal{F}_t}(\tilde{\mathbf{w}})\}$. Subsequently, we obtain an optimal exponential parametrization [let us assume $\phi_{\mathcal{F}_i}(\cdot)$] based on the mapped cluster. Consequently, we continue to optimize the weight vector $\tilde{\mathbf{w}}$ via $\phi_{\mathcal{F}_i}(\cdot)$. Note that HBNN obtains the binarized weight vector via sign$(\phi_{\mathcal{F}_i}(\tilde{\mathbf{w}}))$.

we then update the cluster $\mathcal{F}$ in hyperbolic space

$$\mathcal{F} \leftarrow \begin{Bmatrix} \mathcal{F}_1 \oplus -\eta \otimes \dfrac{\partial \mathcal{L}}{\partial \mathcal{F}_1} \\ \mathcal{F}_2 \oplus -\eta \otimes \dfrac{\partial \mathcal{L}}{\partial \mathcal{F}_2} \\ \vdots \\ \mathcal{F}_t \oplus -\eta \otimes \dfrac{\partial \mathcal{L}}{\partial \mathcal{F}_t} \end{Bmatrix} \tag{9}$$

where the notation $\otimes$ represents the multiplication formalism for hyperbolic geometry.

Multiplication [39]: In the Poincaré ball, the scalar multiplication of $p \in \mathbb{D}_r^n \backslash \{\mathbf{0}\}$ by $c \in \mathbb{R}$ is defined as follows:

$$c \otimes p := (1/\sqrt{r}) \tanh\left(c \tanh^{-1}\left(\sqrt{r}\|p\|\right)\right) \frac{p}{\|p\|}. \tag{10}$$

Recall that the straight-through estimator $\partial\mathcal{L}/\partial\mathbf{w} = \partial\mathcal{L}/\partial \operatorname{sign}(\mathbf{w})$ holds when $|\mathbf{w}| \leq 1$ is satisfied, as indicated by (3). In hyperbolic space, the weight vector $\mathbf{w} := \phi_{\mathcal{F}}(\tilde{\mathbf{w}}) \in \mathbb{D}_r^n$ naturally satisfies the constraint $\|\mathbf{w}\| < 1/\sqrt{r}$. By slightly modifying the bounds of the straight-through estimator $(1 \to 1/\sqrt{r})$, we can directly use $\partial\mathcal{L}/\partial\mathbf{w} = \partial\mathcal{L}/\partial \operatorname{sign}(\mathbf{w})$, which is always guaranteed to hold.

Assuming that $\mathcal{F}_i$ is an optimal point ($\phi_{\mathcal{F}_i}(\cdot)$ represents an optimal exponential parametrization) obtained by updating (9), we have

$$\min_{\tilde{\mathbf{w}} \in \mathbb{R}^n} \mathcal{L}\left(\phi_{\mathcal{F}_i}(\tilde{\mathbf{w}}); \mathcal{D}\right). \tag{11}$$

Following the straight-through estimator, we can compute the gradients $\partial\mathcal{L}/\partial\mathbf{w}$ to update the weight vector in the unconstrained Euclidean space:

$$\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}} \phi'_{\mathcal{F}_i}(\tilde{\mathbf{w}}). \tag{12}$$

Notably, the optimization of HBNN is an iterative process. Initially, we update the EPC $\phi_{\mathcal{F}}(\cdot)$ to obtain the optimal exponential parametrization $\phi_{\mathcal{F}_i}(\cdot)$ while keeping the weight vector $\tilde{\mathbf{w}}$ fixed. Subsequently, we update the weight vector $\tilde{\mathbf{w}}$ using $\phi_{\mathcal{F}_i}(\cdot)$. The training process is summarized in Algorithm 1.

Intuitively, we can map the weight vector from hyperbolic space back to Euclidean space using the inverse of the optimal exponential parametrization $\phi_{\mathcal{F}_i}(\cdot)$. This mapping is a differentiable, as confirmed by the algebraic identity $\phi_{\mathcal{F}_i}^{-1}(\phi_{\mathcal{F}_i}(v)) = v$, satisfying a closed formula.

V. Method Analysis

A. Theoretical Analysis

In Riemannian geometry, the Riemannian exponential map serves as a metric change. In this article, the EPC $\phi_{\mathcal{F}}(\cdot)$ applies gradient descent to update the cluster $\mathcal{F}$, which is an operation equivalent to the Riemannian exponential map $\exp(\cdot)$ with an optimal metric change in hyperbolic space. This optimal metric change is evaluated by comparing multiple changes of metric, as opposed to a single change, in hyperbolic space.

Definition 1 (Diffeomorphism [38]): Given a 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.

According to Definition 1, the EPC is a diffeomorphism in the Poincaré ball $\mathbb{D}_r^n$. This property ensures that the optimization of parametrized weight vectors does not introduce or eliminate local minima at the loss landscape. However, the diffeomorphism ceases at the boundary of the Poincaré ball $\partial\mathbb{D}_r^n$, i.e., the sphere $\mathbb{S}_r^n$, potentially altering the local minima. Therefore, the Poincaré ball $\mathbb{D}_r^n$ is a preferable choice over the sphere $\mathbb{S}_r^n$.

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 cluster $\mathcal{F}$, and a learning rate $\eta$.
Ensure: Update $\tilde{\mathbf{w}}_k$ and $\mathcal{F}$.
1: {Forward propagation}
2: for $k = 1$ to $l-1$ do
3:     Compute the weight via an optimal exponential parametrization: $\mathbf{w}_k \leftarrow \phi_{\mathcal{F}_i}(\tilde{\mathbf{w}}_k)$;
4:     Binarize the weight: $\mathbf{w}_k^b \leftarrow \text{sign}(\mathbf{w}_k)$;
5:     Binarize the activation: $\mathbf{a}_{k-1}^b \leftarrow \text{sign}(\mathbf{a}_{k-1})$;
6:     Perform: $\mathbf{a}_k \leftarrow \text{XnorDotProduct}(\mathbf{w}_k^b, \mathbf{a}_{k-1}^b)$;
7:     Perform: $\mathbf{a}_k \leftarrow \text{BatchNorm}(\mathbf{a}_k)$;
8: end for
9: {Backward propagation}
10: Optimize the unconstrained problem with Eq. (8);
11: Compute the gradient of the overall loss function, i.e., $\frac{\partial \mathcal{L}}{\partial \mathbf{a}}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$ and $\frac{\partial \mathcal{L}}{\partial \mathcal{F}}$, where the sign function can be handled in Eq. (3) for the weight and Eq. (4) for activation;
12: {The parameter update}
13: Optimize the exponential parametrization cluster $\phi_{\mathcal{F}}(\cdot)$ in Eq. (6) by updating the cluster in Eq. (9), then obtain the optimal exponential parametrization $\phi_{\mathcal{F}_i}(\cdot)$;
14: Update the weight using Eq. (12) based on the optimal exponential parametrization $\phi_{\mathcal{F}_i}(\cdot)$;
```

Definition 2 (Segment [44]): The segment domain seg_p is

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

which satisfies $\mathbb{D}_r^n = \exp(\text{seg}_p)$.

For the Riemannian exponential map, Definition 2 indicates that the segment domain seg_p is a closed star-shaped subset of $\mathbb{R}^n$. As for the EPC, we have $\mathbb{D}_r^n = \phi_{\mathcal{F}_1}(\text{seg}_p^*) \cup \phi_{\mathcal{F}_2}(\text{seg}_p^*) \cdots \cup \phi_{\mathcal{F}_t}(\text{seg}_p^*)$. This implies that, in order to cover $\mathbb{D}_r^n$, the required segment seg_p^* for the EPC is less than or equal to the required segment seg_p for the Riemannian exponential map, i.e., $\text{seg}_p^* \subseteq \text{seg}_p$. In practice, the EPC increases the probability of weight flips by shrinking the segment domain, suggesting that weight vectors in seg_p^* can explore more efficiently than in seg_p.

B. Method Comparison and Explanation

1) HBNN Versus BNN: The improvement of HBNN over the general BNN can be primarily attributed to the unconstrained optimization via the EPC. In backpropagation, HBNN introduces additional computational overhead to the training process. Considering (9) and (12), we update both $\mathcal{F}$ and $\tilde{\mathbf{w}}$, thereby increasing the number of trainable parameters. In the inference phase, HBNN behaves similar to general BNNs, because both $\tilde{\mathbf{w}}$ and $\mathcal{F}$ contribute to binarized weight vectors $\text{sign}(\mathbf{w}) = \text{sign}(\phi_{\mathcal{F}_i}(\tilde{\mathbf{w}}))$ based on the optimal exponential parametrization $\phi_{\mathcal{F}_i}$. While general BNNs obtain binarized weight vectors through $\text{sign}(\tilde{\mathbf{w}})$, the representations of $\text{sign}(\tilde{\mathbf{w}})$ and $\text{sign}(\mathbf{w})$ involve the same parameter size and OPs in the inference phase. Therefore, HBNN does not introduce additional computational overhead to the inference process.

TABLE I
TOP-1 CLASSIFICATION ACCURACY RESULTS ON CIFAR100 WITH RESNET18 WITH RESPECT TO DIFFERENT RADII r

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

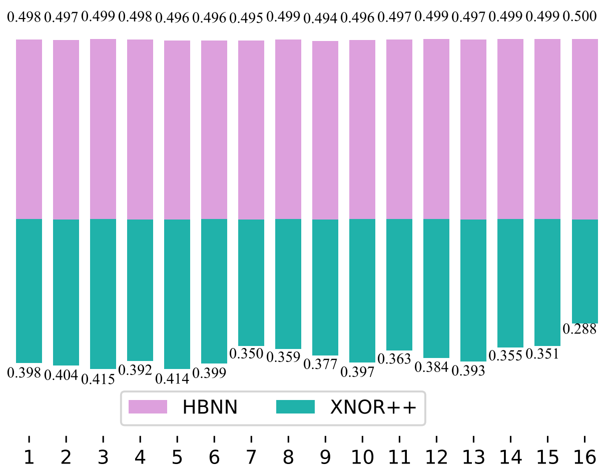


Fig. 3. Weight flip rates of our HBNN and XNOR++ in different layers of ResNet18.

2) HBNN Versus MD: The method of MD [18] presents the mirror descent framework, mapping variables from the unconstrained space to the quantized one, which proves beneficial to BNN optimization. In contrast, HBNN provides the Riemannian geometry framework, mapping variables from the unconstrained space to hyperbolic space. From the perspective of mapping, the mirror map of MD is set artificially, whereas the EPC in HBNN is optimized via the derivative of the loss function with respect to $\mathcal{F}$. From the perspective of optimization, the unconstrained problem of MD solely aims at optimizing the weight vector. However, HBNN also takes into account the optimization of the mapping itself, i.e., the EPC. This dual optimization is advantageous for increasing the probability of weight flips to maximize the information gain.

VI. EXPERIMENTS

In this section, we conduct experiments to compare our HBNN, trained from scratch, with the existing state-of-the-art methods in classification tasks. We evaluate the performance of the proposed method on CIFAR [45] and ImageNet [1] datasets. All experiments are implemented on NVIDIA 3090Ti using the PyTorch framework.

CIFAR Datasets: The CIFAR benchmarks consist of natural color images with 32×32 pixels. There are two datasets: CIFAR10 (C10) with images organized into ten classes and CIFAR100 (C100) with images organized into 100 classes.

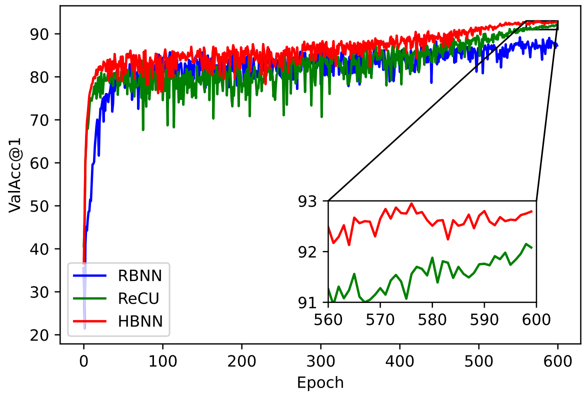


Fig. 4. Validation accuracy curves of our HBNN, RBNN, and ReCU on the CIFAR10 dataset with VGGsmall.

TABLE II

Top-1 Classification Accuracy Results on CIFAR10 and CIFAR100 Datasets With ResNet18 and VGGsmall. W/A Denotes the Bit Width of Weights/Activations

	Method	Bit (W/A)	Acc.(%) (C10)	Acc.(%) (C100)
ResNet18	Full-precision	32/32	94.8	77.0
	IR-Net [30]	1/1	91.5	64.5
	RBNN [19]	1/1	92.2	65.3
	IR-Net+CMIM [47]	1/1	92.2	71.2
	ReSTE [48]	1/1	92.6	-
	ReCU [31]	1/1	92.8	-
	SBNN (ours)	1/1	92.8	71.2
	HBNN (ours)	1/1	**93.3**	**71.7**
	BC [49]	1/32	91.6	72.1
	MD-softmax-s [18]	1/32	93.3	72.2
	SBNN (ours)	1/32	unstable	unstable
	HBNN (ours)	1/32	**94.8**	**74.8**
VGGsmall	Full-precision	32/32	94.1	75.5
	XNOR [26]	1/1	89.8	-
	DoReFa [50]	1/1	90.2	-
	RAD [51]	1/1	90.5	-
	Proxy-BNN [29]	1/1	91.8	67.2
	RBNN [19]	1/1	91.3	67.4
	DSQ [52]	1/1	91.7	-
	SLB [53]	1/1	92.0	-
	ReCU [31]	1/1	92.2	-
	RBNN+CMIM [47]	1/1	92.2	71.0
	ReSTE [48]	1/1	92.5	-
	SBNN (ours)	1/1	92.8	72.2
	HBNN (ours)	1/1	**93.4**	**72.6**

Each dataset comprises 50k training images and 10k test images. We adopt a standard data augmentation scheme, including random clipping and flipping, which is widely used [46]. The images are normalized in preprocessing using the means and standard deviations of channels.

ImageNet Dataset: The ImageNet benchmark consists of 1.2 million high-resolution natural images, with a validation set containing 50k images. These images are organized into 1000 object categories for training and resized to 224 × 224 pixels before being fed into the network. Standard data augmentation strategies, such as random clips and horizontal flips [46], are applied. Single-crop evaluation results are reported using Top-1 and Top-5 accuracies.

Experimental Setup: For CIFAR datasets, our HBNNs are trained for a total of 600 epochs with a batch size of 256. We adopt the SGD optimizer with a momentum of 0.9 and a weight decay of 5e-4. For the ImageNet dataset, our HBNN is trained for a total of 250 epochs with a batch size of 512. The same SGD optimizer settings are used, with a momentum of 0.9 and a weight decay of 1e-4. Notably, we initialize the learning rate at 0.1 and utilize the cosine learning rate scheduler in CIFAR10/CIFAR100 and ImageNet.

TABLE III

Top-1 and Top-5 Classification Accuracy Results on the ImageNet Dataset With ResNet18 and ResNet34. W/A Denotes the Bit Width of Weights/Activations

	Method	Bit (W/A)	Acc.(%) (Top-1)	Acc.(%) (Top-5)
ResNet18	Full-precision	32/32	69.6	89.2
	ABC-Net [54]	1/1	42.7	67.6
	XNOR [26]	1/1	51.2	73.2
	BiReal [28]	1/1	56.4	79.5
	IR-Net [30]	1/1	58.1	80.0
	RBNN [19]	1/1	59.9	81.9
	FDA-BNN [55]	1/1	60.2	82.3
	ReCU [31]	1/1	61.0	82.6
	RBNN+CMIM [47]	1/1	61.2	82.2
	SBNN (ours)	1/1	61.5	83.3
	ReBNN [56]	1/1	61.6	83.4
	HBNN (ours)	1/1	**61.8**	**83.6**
ResNet34	Full-precision	32/32	73.3	91.3
	XNOR++ [27]	1/1	57.1	79.9
	LNS [57]	1/1	59.4	81.7
	BiReal [28]	1/1	62.2	83.9
	IR-Net [30]	1/1	62.9	84.1
	RBNN [19]	1/1	63.1	84.4
	RBNN+CMIM [47]	1/1	65.0	85.7
	ReCU [31]	1/1	65.1	85.8
	SBNN (ours)	1/1	65.6	86.0
	ReBNN [56]	1/1	65.8	86.2
	HBNN (ours)	1/1	**65.9**	**86.4**

A. Ablation Study

We conduct a series of ablation studies on CIFAR100 using the ResNet18 model. Leveraging two parameter spaces, namely, HBNN for the Poincaré ball $\mathbb{D}_r^n$ and SBNN for the sphere $\mathbb{S}_r^n$, i.e., the boundary of the Poincaré ball $\partial\mathbb{D}_r^n$, we adjust different radii r to determine the optimal radius by evaluating classification accuracies at epoch 120. The mean Top-1 accuracies (mean ± std) are presented in Table I. Consequently, we determine $r = 0.05$ for the parameter space $\mathbb{D}_r^n$ and $r = 1$ for the parameter space $\mathbb{S}_r^n$, which will be used in subsequent experiments. While the radius choice has a slight impact, its effect is minimal when considering the variation in results due to different random seeds. Thus, the radius can be considered a robust parameter in our method.

Fig. 3 illustrates the weight flip rates of our HBNN and XNOR++ in different layers of ResNet18 on CIFAR10. As observed, HBNN results in approximately 50% weight flips in each layer, demonstrating that the EPC effectively increases the probability of weight flips.

B. Comparison With State-of-the-Art Methods

The validation curves for ResNet18 are presented in Fig. 4. In comparison with RBNN [19] and ReCU [31] on CIFAR10,

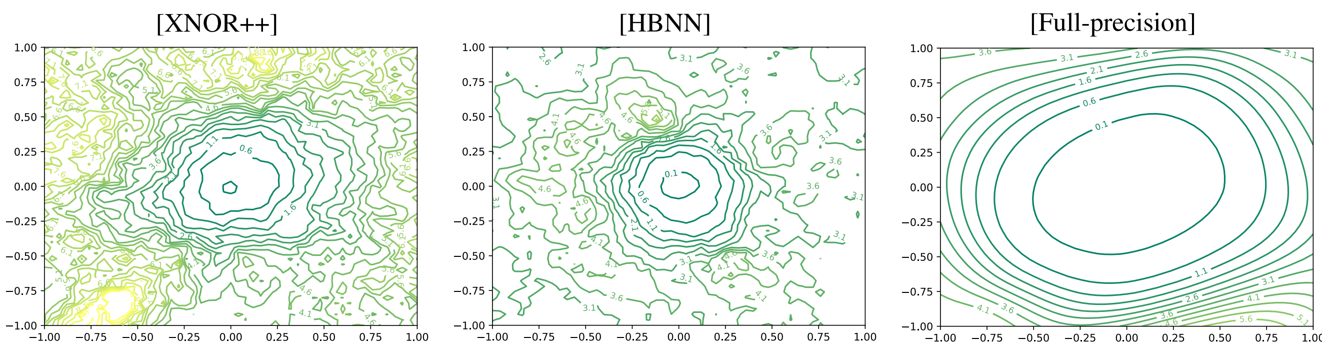


Fig. 5. 2-D visualization of the loss surfaces of ResNet18 on CIFAR10 dataset enables comparisons of the sharpness/flatness of different methods. The sharpness of loss surfaces is indicated by the accompanying numbers, with the yellow area representing particularly large peaks. In comparison with XNOR++, HBNN exhibits flatter loss surfaces.

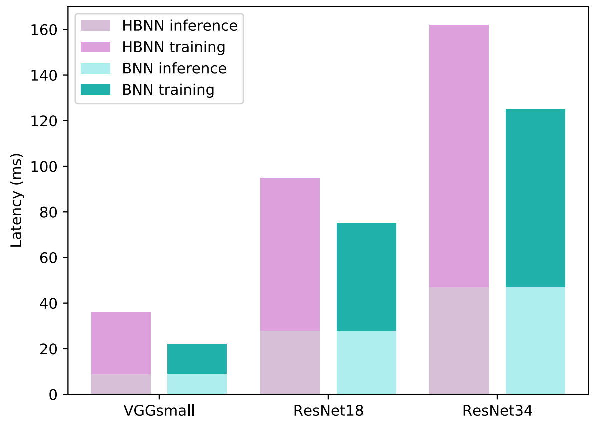


Fig. 6. Latency comparison between HBNN and BNN during inference and training phases.

TABLE IV
PARAMETER SIZE AND OPS IN RESNET MODELS

	Method	Size (MB)	Size reduction	OPs (10^8)
ResNet18	Full-precision	46.76	-	18.21
	ReBNN [56]	4.15	11.26×	1.63
	RBNN [19]	4.15	11.26×	1.63
	ReCU [31]	4.15	11.26×	1.63
	HBNN (ours)	4.15	11.26×	1.63
ResNet34	Full-precision	87.19	-	36.74
	ReBNN [56]	5.41	16.11×	1.93
	RBNN [19]	5.41	16.11×	1.93
	ReCU [31]	5.41	16.11×	1.93
	HBNN (ours)	5.41	16.11×	1.93

TABLE V
TOP-1 CLASSIFICATION ACCURACY RESULTS ON THE CIFAR10 DATASET WITH RESNET18 AND VGGSMALL. W/A DENOTES THE BIT WIDTH OF WEIGHTS/ACTIVATIONS

	Method	Bit-width (W/A)	Acc.(%) (CIFAR10)
ResNet18	Full-precision	32/32	94.8
	IR-Net [30]	1/1	91.5
	IR-Net+HBNN	1/1	92.1
	ReCU [31]	1/1	92.8
	ReCU+HBNN	1/1	93.0
VGGsmall	Full-precision	32/32	94.1
	IR-Net [30]	1/1	90.4
	IR-Net+HBNN	1/1	92.6
	ReCU [31]	1/1	92.2
	ReCU+HBNN	1/1	93.0

TABLE VI
TOP-1 AND TOP-5 CLASSIFICATION ACCURACY RESULTS ON THE IMAGENET DATASET WITH RESNET18 AND RESNET34. W/A DENOTES THE BIT WIDTH OF WEIGHTS/ACTIVATIONS

	Method	Bit-width (W/A)	Acc.(%) (Top-1)	Acc.(%) (Top-5)
ResNet18	Full-precision	32/32	69.6	89.2
	IR-Net [30]	1/1	58.1	80.0
	IR-Net+HBNN	1/1	60.9	82.9
	ReCU [31]	1/1	61.0	82.6
	ReCU+HBNN	1/1	61.5	83.3
ResNet34	Full-precision	32/32	73.3	91.3
	IR-Net [30]	1/1	62.9	84.1
	IR-Net+HBNN	1/1	64.2	85.2
	ReCU [31]	1/1	65.1	85.8
	ReCU+HBNN	1/1	65.8	86.4

the validation accuracies of our HBNN show robust and stable convergence throughout the training epochs.

We conduct a thorough evaluation of our method against state-of-the-art methods, repeating each experiment five times and reporting statistics from the last 10/5 epochs' test accuracies for a fair comparison. As indicated in Table II, HBNN consistently outperforms the existing SOTA methods. Notably, our HBNN (with 1-bit weights and 1-bit activations) achieves performance improvements exceeding 1.2% and 1.6% with the VGGsmall architecture on CIFAR10 and CIFAR100, respectively. For the 1/32 case, the instability in the training of SBNN is noteworthy, possibly stemming from the EPC stopping a diffeomorphism in the sphere $\mathbb{S}_r^n$, based on our theoretical analysis in Section V.

Table III highlights that HBNN consistently outperforms the existing state-of-the-art methods in both Top-1 and Top-5 accuracies. Specifically, our proposed method achieves a 0.8% improvement in Top-1 accuracy with the ResNet18 and ResNet34 architectures compared with the ReCU method on ImageNet.

C. Visualization

In addition, we provide a 2-D visualization of the loss surfaces for both HBNN and XNOR++ in according with previous work [58]. Analyzing Fig. 5, it becomes evident that the loss surface of the full-precision model is smooth and flat, a characteristic beneficial to training and representing in neural networks. With the EPC, HBNN maintains the property of diffeomorphism by not introducing or eliminating local minima in the loss surfaces, in contrast to XNOR++. Consequently, HBNN exhibits relatively flatter loss surfaces than XNOR++, suggesting that the EPC contributes to more effective optimization of BNNs.

D. Computational Complexity

Based on the analysis in Section V, HBNN introduces additional computational overhead to the training process, as it requires training weights and the EPC. Nevertheless, during inference, HBNN behaves similar to general BNNs, because both $\tilde{\mathbf{w}}$ and $\mathcal{F}$ contribute to binarized weight vectors $\text{sign}(\mathbf{w}) = \text{sign}(\phi_{\mathcal{F}_i}(\tilde{\mathbf{w}}))$ based on the optimal exponential parametrization $\phi_{\mathcal{F}_i}$. While general BNNs obtain binarized weight vectors through $\text{sign}(\tilde{\mathbf{w}})$, the representations of $\text{sign}(\tilde{\mathbf{w}})$ and $\text{sign}(\mathbf{w})$ involve the same parameter size and OPs in the inference phase. Therefore, HBNN does not introduce additional computational overhead to the inference process.

In Fig. 6, when comparing the latency of HBNN and BNN in the inference and training phases, we observe that the latency of HBNN is slightly higher than that of BNN during the training process across different models, while their inference latency remains consistent, thereby confirming our previous analysis. Furthermore, we use the parameter size and OPs following [56] for comparison with other methods. As shown in Table IV, HBNN exhibits the same parameter size and OPs as other methods in the inference phase, which is significant for real-time applications.

E. Compatibility

We further evaluate the compatibility of HBNN to illustrate the universality of our method. We integrate HBNN into IR-Net and ReCU as a plug-and-play module, as presented in Tables V and VI. The incorporation of HBNN into these methods results in a noticeable performance improvement.

VII. Conclusion

This article introduces the optimization framework of HBNN, which transforms a constrained problem in hyperbolic space into an unconstrained one in Euclidean space using the EPC. Through the analysis of the EPC, we have determined that it accelerates the exploration of weight vectors, thereby increasing the probability of weight flips compared to the Riemannian exponential map. Experimental results demonstrate that HBNN achieves approximately 50% weight flips, effectively optimizing BNNs to achieve state-of-the-art performance. In the future, our focus will shift toward further exploring the optimization of neural networks from a geometrical perspective.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2016, pp. 779–788.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2961–2969.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[6] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1520–1528.

[7] X. Ding, X. Zhou, Y. Guo, J. Han, and J. Liu, "Global sparse momentum SGD for pruning very deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[8] M. Lin et al., "HRank: Filter pruning using high-rank feature map," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1529–1538.

[9] S. Bai, J. Chen, X. Shen, Y. Qian, and Y. Liu, "Unified data-free compression: Pruning and quantization without fine-tuning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 5876–5885.

[10] R. Banner, I. Hubara, E. Hoffer, and D. Soudry, "Scalable methods for 8-bit training of neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

[11] K. Helwegen, J. Widdicombe, L. Geiger, Z. Liu, K.-T. Cheng, and R. Nusselder, "Latent weights do not exist: Rethinking binarized neural network optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[12] J. Chen, Y. Liu, H. Zhang, S. Hou, and J. Yang, "Propagating asymptotic-estimated gradients for low bitwidth quantized neural networks," *IEEE J. Sel. Top. Signal Process.*, vol. 14, no. 4, pp. 848–859, May 2020.

[13] J. Chen, H. Chen, M. Wang, G. Dai, I. W. Tsang, and Y. Liu, "Learning discretized neural networks under Ricci flow," 2023, *arXiv:2302.03390*.

[14] J. Chen, S. Bai, T. Huang, M. Wang, G. Tian, and Y. Liu, "Data-free quantization via mixed-precision compensation without fine-tuning," *Pattern Recognit.*, vol. 143, Nov. 2023, Art. no. 109780.

[15] Y. Liu, J. Chen, and Y. Liu, "DCCD: Reducing neural network redundancy via distillation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 10006–10017, Jul. 2024.

[16] J. Chen, L. Liu, Y. Liu, and X. Zeng, "A learning framework for n-bit quantized neural networks toward FPGAs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 1067–1081, Mar. 2021.

[17] S. Bubeck, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, nos. 3–4, pp. 231–357, 2015.

[18] T. Ajanthan, K. Gupta, P. Torr, R. Hartley, and P. Dokania, "Mirror descent view for neural network quantization," in *Proc. Int. Conf. Artif. Intell. Statist.*, Mar. 2021, pp. 2809–2817.

[19] M. Lin et al., "Rotated binary neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 7474–7485.

[20] H. W. Guggenheimer, *Differential Geometry*. Courier Corporation, 2012.

[21] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[22] J. Chen, "Decentralized Riemannian conjugate gradient method on the stiefel manifold," in *Proc. 12th Int. Conf. Learn. Represent.*, 2024. [Online]. Available: https://openreview.net/forum?id=PQbFUMKLFp

[23] K. Helfrich, D. Willmott, and Q. Ye, "Orthogonal recurrent neural networks with scaled Cayley transform," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2018, pp. 1969–1978.

[24] M. Lezcano-Casado and D. Martinez-Rubio, "Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2019, pp. 3794–3803.

[25] M. L. Casado, "Trivializations for gradient-based optimization on manifolds," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[26] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 525–542.

[27] A. Bulat and G. Tzimiropoulos, "XNOR-Net++: Improved binary neural networks," 2019, *arXiv:1909.13863*.

[28] Z. Liu, W. Luo, B. Wu, X. Yang, W. Liu, and K.-T. Cheng, "Bi-Real net: Binarizing deep network towards real-network performance," *Int. J. Comput. Vis.*, vol. 128, no. 1, pp. 202–219, Jan. 2020.

[29] X. He et al., "ProxyBNN: Learning binarized neural networks via proxy matrices," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Springer, Aug. 2020, pp. 223–241.

[30] H. Qin et al., "Forward and backward information retention for accurate binary neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2250–2259.
[31] Z. Xu et al., "ReCU: Reviving the dead weights in binary neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Oct. 2021, pp. 5198–5208.
[32] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016.
[33] L. Huang, X. Liu, Y. Liu, B. Lang, and D. Tao, "Centered weight normalization in accelerating training of deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2822–2830.
[34] P. Petersen, *Riemannian Geometry* (Graduate Texts in Mathematics). Springer-Verlarg, 2006.
[35] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016, *arXiv:1602.02830*.
[36] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning," *Coursera, Video Lectures*, vol. 264, no. 1, pp. 2146–2153, 2012.
[37] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.
[38] J. W. Anderson, *Hyperbolic Geometry*. Springer, 2006.
[39] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.
[40] M. Nickel and D. Kiela, "Poincaré embeddings for learning hierarchical representations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
[41] O. Ganea, G. Bécigneul, and T. Hofmann, "Hyperbolic entailment cones for learning hierarchical embeddings," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2018, pp. 1646–1655.
[42] A. A. Ungar, "Hyperbolic trigonometry and its application in the Poincaré ball model of hyperbolic geometry," *Comput. Math. Appl.*, vol. 41, nos. 1–2, pp. 135–147, Jan. 2001.
[43] A. A. Ungar, "A gyrovector space approach to hyperbolic geometry," *Synth. Lectures Math. Statist.*, vol. 1, no. 1, pp. 1–194, Jan. 2008.
[44] P. Petersen, *Riemannian Geometry*. Springer, 2016, pp. 1–39.
[45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Toronto, ON, Canada, 2009.
[46] W. Wang et al., "Accelerate CNNs from three dimensions: A comprehensive pruning framework," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2021, pp. 10717–10726.
[47] Y. Shang, D. Xu, Z. Zong, L. Nie, and Y. Yan, "Network binarization via contrastive learning," 2022, *arXiv:2207.02970*.
[48] X.-M. Wu, D. Zheng, Z. Liu, and W.-S. Zheng, "Estimator meets equilibrium perspective: A rectified straight through estimator for binary neural networks training," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2023, pp. 17009–17018.
[49] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015.
[50] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients," 2016, *arXiv:1606.06160*.
[51] R. Ding, T.-W. Chin, Z. Liu, and D. Marculescu, "Regularizing activation distribution for training binarized deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11408–11417.
[52] R. Gong et al., "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4852–4861.
[53] Z. Yang et al., "Searching for low-bit weights in quantized neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 4091–4102.
[54] X. Lin, C. Zhao, and W. Pan, "Towards accurate binary convolutional neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.
[55] Y. Xu, K. Han, C. Xu, Y. Tang, C. Xu, and Y. Wang, "Learning frequency domain approximation for binary neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 25553–25565.
[56] S. Xu et al., "Resilient binary neural network," in *Proc. AAAI Conf. Artif. Intell.*, 2023, vol. 37, no. 9, pp. 10620–10628.
[57] K. Han, Y. Wang, Y. Xu, C. Xu, E. Wu, and C. Xu, "Training binary neural networks through learning with noisy supervision," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2020, pp. 4017–4026.
[58] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018.

Jun Chen received the B.S. degree from the Department of Mechanical and Electrical Engineering, China Jiliang University, Hangzhou, China, in 2016, and the M.S. degree in control engineering and Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, in 2020 and 2024, respectively.

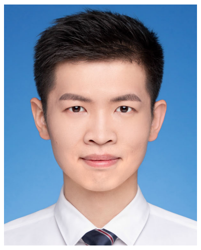
He is currently a Distinguished Professor at Zhejiang Normal University, Jinhua, China. His research interests include deep learning, model compression, decentralized optimization, and manifold optimization.

Jingyang Xiang received the B.S. degree in electrical engineering and automation from Zhejiang University of Technology, Hangzhou, China, in 2022. He is currently pursuing the M.S. degree with the College of Control Science and Engineering, Zhejiang University, Hangzhou.

His current research interests include efficient AI, especially LLM quantization.

Tianxin Huang received the bachelor's degree in mechanical engineering from Xi'an Jiaotong University (XJTU), Xi'an, China, in 2017, and the doctor's degree from the April Lab, Zhejiang University, Hangzhou, China, in 2023.

He is currently a Research Fellow (Postdoc) at the National University of Singapore (NUS), School of Computing, Singapore, focusing on 3-D computer vision. His current research interests include but not limited to 3-D reconstruction, neural rendering, 3-D face reconstruction, and 3-D point cloud analysis.



Xiangrui Zhao received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2018, and the Ph.D. degree from the Institute of Cyber Systems and Control, Zhejiang University, Hangzhou, China, in 2023.

His current research interests include BEV perception and end-to-end autonomous driving.

Yong Liu (Member, IEEE) received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2001 and 2007, respectively.

He is currently a Professor with the Institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. He has published more than 30 research articles in machine learning, computer vision, information fusion, and robotics. His latest research interests include machine learning, robotics vision, information processing, and granular computing.