Lipschitz Continuity Retained Binary Neural Network

Yuzhang Shang[1], Dan Xu[2], Bin Duan[1],
Ziliang Zong[3], Liqiang Nie[4], and Yan Yan[1]*

[1] Illinois Institute of Technology, USA
[2] Hong Kong University of Science and Technology, Hong Kong
[3] Texas State University, USA
[4] Harbin Institute of Technology, Shenzhen, China
{yshang4, bduan2}@hawk.iit.edu, danxu@cse.ust.hk, ziliang@txstate.edu,
nieliqiang@gmail.com, and yyan34@iit.edu

Abstract. Relying on the premise that the performance of a binary neural network can be largely restored with eliminated quantization error between full-precision weight vectors and their corresponding binary vectors, existing works of network binarization frequently adopt the idea of model robustness to reach the aforementioned objective. However, robustness remains to be an ill-defined concept without solid theoretical support. In this work, we introduce the Lipschitz continuity, a well-defined functional property, as the rigorous criteria to define the model robustness for BNN. We then propose to retain the Lipschitz continuity as a regularization term to improve the model robustness. Particularly, while the popular Lipschitz-involved regularization methods often collapse in BNN due to its extreme sparsity, we design the Retention Matrices to approximate spectral norms of the targeted weight matrices, which can be deployed as the approximation for the Lipschitz constant of BNNs without the exact Lipschitz constant computation (NP-hard). Our experiments prove that our BNN-specific regularization method can effectively enhance the robustness of BNN (testified on ImageNet-C), achieving SoTA on CIFAR10 and ImageNet. Our code is available at https://github.com/42Shawn/LCR_BNN.

Keywords: Neural Network Compression, Network Binarization, Lipschitz Continuity

1 Introduction

Recently, Deep Neural Networks achieve significant accomplishment in computer vision tasks such as image classification [26] and object detection [42,27]. However, their inference-cumbersome problem hinders their broader implementations. To develop deep models in resource-constrained edge devices, researchers propose several neural network compression paradigms, *e.g.,* knowledge distillation [21,20], network pruning [28,16] and network quantization [24,40]. Among

* Corresponding author.

the network quantization methods, the network binarization [24] stands out, as it extremely quantizes weights and activations (*i.e.* intermediate feature maps) to ± 1. Under this framework, the full-precision (FP) network is compressed $32\times$ more, and the time-consuming inner-product operations are replaced with the efficient Xnor-bitcount operations.

However, BNNs can hardly achieve comparable performance to the original models due to the loss of FP weights and activations. A major reason for the performance drop is that the inferior robustness comes from the error amplification effect, where the binarization operation degrades the distance induced by amplified noise [30]. The destructive manner of $sgn(\cdot)$ severely corrupts the robustness of the BNN, and thus undermines their representation capacity [6,18,34].

As some theoretical works validated, robustness is a significant property for functions (neural networks in our context), which further influences their generalization ability [35,3]. In the above-mentioned binarization works, researchers investigate the effectiveness of their methods via the ill-defined concepts of function robustness without solid theoretical support, such as observing the visualized distributions of weights and activations [18,31,34,30]. However, they rarely introduced the well-defined mathematical property, Lipschitz continuity, for measuring the robustness of functions into BNN. Lipschitz continuity has been proven to be a powerful and strict tool for systematically analyzing deep learning models. For instance, Miyato *et. al.* propose the well-known Spectral Normalization [49,36] utilizing the Lipschitz constant to regularize network training, which is initially designed for GAN and then extended to other network architectures, achieving great success [37]; Lin *et. al.* [30] design a Lipschitz-based regularization method for network (low-bit) quantization, and testify that Lipschitz continuity is significantly related to the robustness of the low-bit network. But simply bridging those existing Lipschitz-based regularization methods with the binary neural networks (1-bit) is sub-optimal, as the exclusive property of BNN, *e.g.*, the extreme sparsity of binary weight matrix [24] impedes calculating the singular values, which is the core module in those Lipschitz-involved methods.

To tackle this problem, we analyze the association between the structures and the Lipschitz constant of BNN. Motivated by this analysis, we design a new approach to effectively retain the Lipschitz constant of BNNs and make it close to the Lipschitz constant of its latent FP counterpart. Particularly, we develop a Lipschitz Continuity Retention Matrix (**RM**) for each block and calculate the spectral norm of **RM** via the iterative power method to avoid the high complexity of calculating exact Lipschitz constants. It is worth to note that the designed loss function for retaining the Lipschitz continuity of BNNs is differentiable *w.r.t.* the binary weights.

Overall, the contributions of this paper are three-fold:

- We propose a novel network binarization framework, named as **L**ipschitz **C**ontinuity **R**atined Binary Neural Network (***LCR***-BNN), to enhance the robustness of binary network optimization process. To the best of our knowl-

edge, we are the first on exploring the Lipschitz continuity to enhance the representation capacity of BNNs;
– We devise a Lipschitz Continuity Retention Matrix to approximate the Lipschitz constant with activations (instead of directly using weights as SN [36] and DQ [30] devised) of networks in the BNN forward pass;
– By adding our designed regularization term on the existing state-of-the-art methods, we observe the enhanced robustness are validated on ImageNet-C and promising accuracy improvement on CIAFR and ImageNet datasets.

2 Related Work

2.1 Network Binarization

In the pioneer art of BNNs, Hubara *et. al.* [24] quantize weights and activations to ± 1 via sign function. Due to the non-differentiability of the sign function, the straight-through estimator (STE) [4] is introduced for approximating the derivative of the sign function. Inspired by this archetype, numerous researchers dig into the field of BNNs and propose their modules to improve the performance of BNNs. For instance, Rastegari *et. al.* [41] reveal that the quantization error between the FP weights and corresponding binarized weights is one of the obstacles degrading the representation capabilities of BNNs. Then they propose to introduce a scaling factor calculated by the L1-norm for both weights and activation functions to minimize the quantization error. XNOR++ [6] absorbs the idea of scaling factor and proposes learning both spatial and channel-wise scaling factors to improve performance. Furthermore, Bi-Real [33] proposes double residual connections with full-precision downsampling layers to lessen the information loss. ProxyBNN [18] designs a proxy matrix as a basis of the latent parameter space to guide the alignment of the weights with different bits by recovering the smoothness of BNNs. Those methods try to lessen the quantization error and investigate the effectiveness from the perspective of model smoothness (normally via visualizing the distribution of weights). A more detailed presentation and history of BNNs can be found in the Survey [39].

However, none of them take the functional property, Lipschitz continuity, into consideration, which is a well-developed mathematical tool to study the robustness of functions. Bridging Lipschitz continuity with BNNs, we propose to retain the Lipschitz continuity of BNNs, which can serve as a regularization term and further improve the performance of BNNs by strengthening their robustness.

2.2 Lipschitz Continuity in Neural Networks

The Lipschitz constant is an upper bound of the ratio between input perturbation and output variation within a given distance. It is a well-defined metric to quantify the robustness of neural networks to small perturbations [45]. Also, the Lipschitz constant $\|f\|_{Lip}$ can be regarded as a functional norm to measure the Lipschitz continuity of given functions. Due to its property, the Lipschitz constant is the primary concept to measure the robustness of functions [3,35,37]. In the deep learning era, previous theoretical arts [47,37] disclose the regularity of deep networks via Lipschitz continuity. Lipschitz continuity is widely introduced into many deep learning topics for achieving the SoTA performance [36,49,46,50]. For example, in image synthesis, Miyato *et. al.* [36,49] devise spectral normalization to constrain the Lipschitz constant of the discriminator for optimizing a generative adversarial network, acting as a regularization term to smooth the discriminator function; in knowledge distillation, Shang *et. al.* [46] propose to utilize the Lipschitz constant as a form of knowledge to supervise the training process of student network; in neural network architecture design, Zhang *et. al.* [50] propose a novel L_∞-dist network using naturally 1-Lipschitz functions as neurons.

The works above highlight the significance of Lipschitz constant in expressiveness and robustness of deep models. Particularly, retaining Lipschitz continuity at an appropriate level is proven to be an effective technique for enhancing the model robustness. Therefore, the functional information of neural networks, Lipschitz constant, should be introduced into network binarization to fill the robustness gap between BNN and its real-valued counterpart.

Relation to Spectral Normalization (SN) [36]. We empirically implement the SN in BNN but fail. By analyzing the failure of the implementation, we conclude that the SN is not suitable for BNNs. The reasons are: (i) One of the key modules in SN is spectral norm computation based on singular value calculatiuon, which is directly implemented on the weight matrix (*e.g.*, the matrices of convolutional and linear layers). But the binarization enforcing the FP weight into 1 or -1 makes the weight matrix extremely sparse. Thus, applying the existing algorithm to binary matrices collapses. (ii) In contrast to normal networks, the forward and backward passes of BNN are more complex, *e.g.*, FP weights (after backpropagation) and binary weights (after binarization) exist in the same training iteration. This complexity problem impedes broader implementations of SN on BNNs as the number of structures in a BNN exceeds the number in a normal network. To tackle those problems, we propose a novel Lipschitz regularization technique targeted to train BNNs. We elaborate more technical comparisons between our method and SN in the following Section 3.3.

3 Lipschitz Continuity Retention for BNNs

3.1 Preliminaries

We first define a general neural network with L fully-connected layers (without bias term for simplification). This network $f(\mathbf{x})$ can be denoted as:

$$f(\mathbf{W}^1, \cdots, \mathbf{W}^L; \mathbf{x}) = (\mathbf{W}^L \cdot \sigma \cdot \mathbf{W}^{L-1} \cdot \cdots \cdot \sigma \cdot \mathbf{W}^1)(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is the input sample and $\mathbf{W}^k \in \mathbb{R}^{d_{k-1} \times d_k} (k = 1, ..., L-1)$ stands for the weight matrix connecting the $(k-1)$-th and the k-th layer, with d_{k-1} and d_k representing the sizes of the input and output of the k-th network layer, respectively. The $\sigma(\cdot)$ function performs element-wise activation for the activations.

Binary Neural Networks. Here, we revisit the general gradient-based method in [7], which maintains full-precision latent variables $\mathbf{W}_F$ for gradient updates, and the k-th weight matrix $\mathbf{W}_F^k$ is binarized into ± 1 binary weight matrix $\mathbf{W}_B^k$ by a binarize function (normally $sgn(\cdot)$) as $\mathbf{W}_B^k = sgn(\mathbf{W}_F^k)$. Then the activation map of the k-th layer is produced by $\mathbf{A}^k = \mathbf{W}_B^k \mathbf{A}^{k-1}$, and a whole forward pass of binarization is performed by iterating this process for L times.
Lipschitz Constant (Definition 1). A function $g : \mathbb{R}^n \longmapsto \mathbb{R}^m$ is called Lipschitz continuous if there exists a constant L such that:

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \|g(\mathbf{x}) - g(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2, \tag{2}$$

where $\mathbf{x}, \mathbf{y}$ represent two random inputs of the function g. The smallest L holding the inequality is the Lipschitz constant of function g, denoted as $\|g\|_{Lip}$. By Definition 1, $\|\cdot\|_{Lip}$ can upper bound of the ratio between input perturbation and output variation within a given distance (generally L2 norm), and thus it is naturally considered as a metric to evaluate the robustness of neural networks [45,43,46].

In the following section, we propose our Lipschitz Continuity Retention Procedure (Sec. 3.2), where the a BNN is enforced to close to its FP counterpart in term of Lipschitz constant. In addition, we introduce the proposed loss function and gradient approximation for optimizing the binary network (Sec. 3.3). Finally, we discuss the relation between *LCR* and Lipschitz continuity, and compare our method to the well-known Spectral Normalization [36] (Sec. 3.3).

3.2 Lipschitz Continuity Retention Procedure

We aim to retain the Lipschitz constants in an appropriate level. In practice, we need to pull $\|f_B\|_{Lip}$ and $\|f_F\|_{Lip}$ closely to stabilize the Lipschitz constant of the BNNs. However, it is NP-hard to compute the exact Lipschitz constant of neural networks [47], especially involving the binarization process. To solve this problem, we propose to bypass the exact Lipschitz constant computation by introducing a sequence of Retention Matrices produced by the adjacent activations, and then compute their spectral norms via power iteration method to form a LCR loss for retaining the Lipschitz continuity of the BNN as demonstrated in Figure 1.
Lipschitz constant of neural networks. We fragment an affine function for the k-th layer with weight matrix $\mathbf{W}^k$, $f^k(\cdot)$ mapping $\mathbf{a}^{k-1} \longmapsto \mathbf{a}^k$, in which $\mathbf{a}^{k-1} \in \mathbb{R}^{d_{k-1}}$ and $\mathbf{a}^k \in \mathbb{R}^{d_k}$ are the activations produced from the $(k-1)$-th and the k-th layer, respectively. Based on Lemma 1 in the Supplemental Materials, $\|f^k\|_{Lip} = \sup_{\mathbf{a}} \|\nabla \mathbf{W}^k(\mathbf{a})\|_{SN}$, where $\|\cdot\|_{SN}$ is the matrix spectral norm formally defined as:

$$\|\mathbf{W}^k\|_{SN} \triangleq \max_{\mathbf{x}:\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{W}^k\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \max_{\|\mathbf{x}\|_2 \leq 1} \|\mathbf{W}^k\mathbf{x}\|_2, \tag{3}$$

where the spectral norm of the matrix $\mathbf{W}$ is equivalent to its largest singular value. Thus, for the f^k, based on Lemma 2 in the Supplemental Materials, its

Lipschitz constant can be derived as:

$$\|\mathbf{W}^k\|_{Lip} = \sup_{\mathbf{a}} \|\nabla \mathbf{W}^k(\mathbf{a})\|_{SN} = \|\mathbf{W}^k\|_{SN}. \tag{4}$$

Moreover, as for the most functional structures in neural network such as ReLU, Tanh, Sigmoid, Sign, batch normalization and other pooling layers, they all have simple and explicit Lipschitz constants [14,36,46]. Note that for the sign function in BNN, though it is not theoretically differentiable, it still has an explicit Lipschitz constant as its derivative is numerically approximated by Hard-Tanh function [4]. This fixed Lipschitz constant property renders our derivation to be applicable to most network architectures, such as binary ResNet [17,24] and variant binary ResNet [34,5].

By the inequality of norm, *i.e.* $\|\mathbf{W}^k \cdot \mathbf{W}^{k+1}\|_{Lip} \leq \|\mathbf{W}^k\|_{Lip} \cdot \|\mathbf{W}^{k+1}\|_{Lip}$, we obtain the following upper bound of the Lipschitz constant of network f, *i.e.,*

$$\|f\|_{Lip} \leq \|\mathbf{W}^L\|_{Lip} \cdot \|\sigma\|_{Lip} \cdot \dots \cdot \|\mathbf{W}^1\|_{Lip} = \prod_{k=1}^{L} \|\mathbf{W}^k\|_{SN}. \tag{5}$$

In this way, we can retain the Lipschitz constant through maintaining a sequence of spectral norms of intermediate layers in the network.

Construction of Lipschitz Continuity Retention Matrix. We now aim to design a novel optimization loss to retain Lipschitz continuity by narrowing the distance between the spectral norms of corresponding weights of full-precision and binary networks. Moreover, we need to compute the spectral norm of binarized weight matrices. Nevertheless, it is inaccessible to calculate the spectral norm of the binary weight matrix $\mathbf{W}_B^k$ in BNNs by popular SVD-based methods [1]. Therefore, we design the Lipschitz Continuity Retention Matrix (**RM**) to bypass the complex calculation of the spectral norm of $\mathbf{W}_B^k$. Approaching the final goal through the bridge of the Retention Matrix allows feasible computation to retain the Lipschitz constant and facilitates its further use as a loss function.

For training data with a batch size of N, we have a batch of corresponding activations after a forward process for the (k-1)-th layer as

$$\mathbf{A}^{k-1} = (\mathbf{a}_1^{k-1}, \cdots, \mathbf{a}_n^{k-1}) \in \mathbb{R}^{d_{k-1} \times N}, \tag{6}$$

where $\mathbf{W}^k\mathbf{A}^{k-1} = \mathbf{A}^k$ for each $k \in \{1, \dots, L-1\}$.

Studies about similarity of activations illustrate that for well-trained networks, their batch of activations in the same layer (*i.e.* $\{\mathbf{a}_i^{k-1}\}, i \in \{1, \dots, n\}$) have strong mutual linear independence. We formalize the independence of the activations as follows:

$$\begin{aligned} (\mathbf{a}_i^{k-1})^\mathsf{T}\mathbf{a}_j^{k-1} &\approx 0, \qquad \forall i \neq j \in \{1, \cdots, N\}, \\ (\mathbf{a}_i^{k-1})^\mathsf{T}\mathbf{a}_i^{k-1} &\neq 0, \qquad \forall i \in \{1, \cdots, N\}. \end{aligned} \tag{7}$$

We also empirically and theoretically discuss the validation of this assumption in the Sec. 4.4.

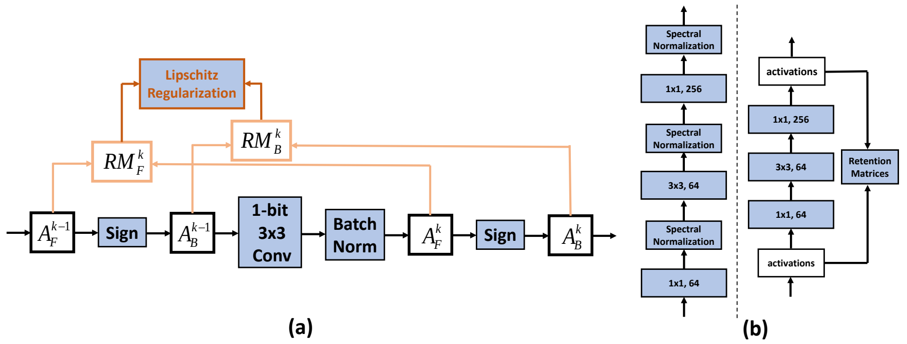

Fig. 1. (**a**) An overview of our Lipschitz regularization for a binary convolutional layer: regularizing the BNN via aligning the Lipschitz constants of binary network and its latent full-precision counterpart is the goal of our work. To reach this goal, the input and output activations of the k-th layer compose the Retention Matrix ($\mathbf{RM}^k$) for approximating the Lipschitz constant of this layer. $\mathbf{RM}_F^k$ and $\mathbf{RM}_B^k$ are then used to calculate the Lipschitz constant of this layer (the validation of this approximation is elaborated in 3.2). Finally, the Lipschitz continuity of the BNN is retained under a regularization module. (**b**) Difference between Spectral Normalization (Left) and *LCR* (Right). More details are discussed in 3.3.

With the above assumption, we formalize the devised Retention Matrix $\mathbf{RM}^k$ for estimating the spectral norm of matrix $\mathbf{W}^k$ as:

$$\begin{aligned}\mathbf{RM}^k &\triangleq \left[(\mathbf{A}^{k-1})^\mathsf{T}\mathbf{A}^k\right]^\mathsf{T}\left[(\mathbf{A}^{k-1})^\mathsf{T}\mathbf{A}^k\right] \\ &= (\mathbf{A}^{k-1})^\mathsf{T}(\mathbf{W}^k)^\mathsf{T}(\mathbf{A}^{k-1})(\mathbf{A}^{k-1})^\mathsf{T}\mathbf{W}^k\mathbf{A}^{k-1}.\end{aligned} \tag{8}$$

Incorporating independence assumption in Eq. 7 (*i.e.*, $(\mathbf{A}^{k-1})(\mathbf{A}^{k-1}) = \mathbf{I}$)) with Eq. 8, we can transfer the $\mathbf{RM}^k$ as follows:

$$\mathbf{RM}^k = (\mathbf{A}^{k-1})^\mathsf{T}(\mathbf{W}^{k^\mathsf{T}}\mathbf{W}^k)\mathbf{A}^{k-1}. \tag{9}$$

Based on Theorem 1 in supplemental material and Eq. 9, $\sigma_1(\mathbf{RM}^k) = \sigma_1(\mathbf{W}^{k^\mathsf{T}}\mathbf{W}^k)$ where $\sigma_1(\cdot)$ is the function for computing the largest eigenvalue, *i.e.*, Retention Matrix $\mathbf{RM}^k$ has the same largest eigenvalue with $\mathbf{W}^{k^\mathsf{T}}\mathbf{W}^k$. Thus, with the definition of spectral norm $\|\mathbf{W}^k\|_{SN} = \sigma_1(\mathbf{W}^{k^\mathsf{T}}\mathbf{W}^k)$, the spectral norm of the matrix $\mathbf{W}^k$ can be yielded through calculating the largest eigenvalue of $\mathbf{RM}^k$, *i.e.* $\sigma_1(\mathbf{RM}^k)$, which is solvable [46].

For networks with more complex layers, such as the residual block and block in MobileNet [17,22], we can also design such a Retention Matrix to bypass the Lipschitz constant computation layer-wisely. By considering the block as an affine mapping from front to back activations, the proposed Retention Matrix can also be produced block-wisely, making our spectral norm calculation more efficient. Specifically, we define the Retention Matrix $\mathbf{RM}$ for the residual blocks as follows:

$$\mathbf{RM}_m \triangleq \left[(\mathbf{A}^f)^\mathsf{T}\mathbf{A}^l\right]^\mathsf{T}\left[(\mathbf{A}^f)^\mathsf{T}\mathbf{A}^l\right], \tag{10}$$

where $\mathbf{A}^f$ and $\mathbf{A}^l$ denote the front-layer activation maps and the back-layer activation maps of the residual block, respectively.

Calculation of Spectral Norms. Here, to calculate the spectral norms of two matrices, an intuitive way is to use SVD to compute the spectral norm, which results in overloaded computation. Rather than SVD, we utilize **Power Iteration** method [12,36] to approximate the spectral norm of the targeted matrix with a small trade-off of accuracy. By Power Iteration Algorithm (see Supplemental Material), we can obtain the spectral norms of the binary and corresponding FP Retention Matrices, respectively (*i.e.* $\|\mathbf{RM}_F^k\|_{SN}$ and $\|\mathbf{RM}_B^k\|_{SN}$ for each $k \in \{1, \dots, L-1\}$). And then, we can calculate the distance between these two spectral norms to construct the loss function.

3.3 Binary Neural Network Optimization

Optimization losses. We define the Lipschitz continuity retention loss function $\mathcal{L}_{Lip}$ as

$$\mathcal{L}_{Lip} = \sum_{k=1}^{L-1} \left[\left(\frac{\|\mathbf{RM}_B^k\|_{SN}}{\|\mathbf{RM}_F^k\|_{SN}} - 1 \right) \beta^{k-L} \right]^2, \tag{11}$$

where β is a coefficient greater than 1. Hence, with k increasing, the $\left[\left(\frac{\|\mathbf{RM}_B^k\|_{SN}}{\|\mathbf{RM}_F^k\|_{SN}} - 1 \right) \beta^{k-L} \right]^2$ increases. In this way, the spectral norm of latter layer can be more retained.

Combined with the cross entropy loss $\mathcal{L}_{CE}$, we propose a novel loss function for the overall optimization objective as

$$\mathcal{L} = \frac{\lambda}{2} \cdot \mathcal{L}_{Lip} + \mathcal{L}_{CE}, \tag{12}$$

where λ is used to control the degree of retaining the Lipschitz constant. We analyze the effect of the coefficient λ in the supplementary material. After we define the overall loss function, our method is finally formulated. The forward and backward propagation processes of *LCR* are elaborated in Algorithm 1.

Gradient Approximation. Several works [44,30,36] investigate the robustness of neural networks by introducing the concept of Lipschitzness. In this section, we differentiate the loss function of our proposed method, and reveal the mechanism of how Lipschitzness effect the robustness of BNNs.

The derivative of the loss function $\mathcal{L}$ w.r.t $\mathbf{W}_B^k$ is:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_B} &= \frac{\partial(\mathcal{L}_{CE})}{\partial \mathbf{W}_B} + \frac{\partial(\mathcal{L}_{Lip})}{\partial \mathbf{W}_B^k} \\ &\approx \mathbf{M} - \lambda \sum_{k=1}^{L-1} \beta^{k-L} \left(\frac{\|\mathbf{RM}_F^k\|_{SN}}{\|\mathbf{RM}_B^k\|_{SN}} \right) \mathbf{u}_1^k (\mathbf{v}_1^k)^\top, \end{aligned} \tag{13}$$

where $\mathbf{M} \triangleq \frac{\partial(\mathcal{L}_{CE})}{\partial \mathbf{W}_B}$, $\mathbf{u}_1^k$ and $\mathbf{v}_1^k$ are respectively the first left and right singular vectors of $\mathbf{W}_B^k$. In the content of SVD, $\mathbf{W}_B^k$ can be re-constructed by a series of

Algorithm 1 Forward and Backward Propagation of *LCR*-BNN

```
Require: A minibatch of data samples $(\mathbf{X}, \mathbf{Y})$, current binary weight $\mathbf{W}_B^k$, latent
    full-precision weights $\mathbf{W}_F^k$, and learning rate $\eta$.
Ensure: Update weights $\mathbf{W}_F^{k\,\prime}$.
 1: Forward Propagation:
 2: for $k = 1$ to $L-1$ do
 3:     Binarize latent weights: $\mathbf{W}_B^k \leftarrow \mathrm{sgn}(\mathbf{W}_F^k)$;
 4:     Perform binary operation with the activations of last layer: $\mathbf{A}_F^k \leftarrow \mathbf{W}_B^k \cdot \mathbf{A}_B^{k-1}$;
 5:     Binarize activations: $\mathbf{A}_B^k \leftarrow \mathrm{sgn}(\mathbf{A}_F^k)$;
 6:     Produce the Retention Matrices $\mathbf{RM}_F^k$ and $\mathbf{RM}_B^k$ by Eq. 9;
 7: end for
 8: Approximate the spectral norm of a series of RMs by Algorithm ?? in the Supple-
    mental Material, and obtain $\|\mathbf{RM}_F^k\|_{SN}$ and $\|\mathbf{RM}_B^k\|_{SN}$ for each $k \in \{1, \ldots, L-1\}$;
 9: Compute the Lipschitz continuity retention loss $\mathcal{L}_{Lip}$ by Eq. 11;
10: Combine the cross entropy loss $\mathcal{L}_{CE}$ and the quantization error loss $\mathcal{L}_{QE}$ for the
    overall loss $\mathcal{L}$ by Eq. 12;
11: Backward Propagation: compute the gradient of the overall loss function,
    i.e. $\frac{\partial \mathcal{L}}{\partial \mathbf{W_B}}$, using the straight through estimator (STE) [4] to tackle the sign function;
12: Parameter Update: update the full-precision weights: $\mathbf{W}_F^{k\,\prime} \leftarrow \mathbf{W}_F^k - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_B^k}$.
```

singular vector, *i.e.*

$$\mathbf{W}_B^k = \sum_{j=1}^{d_k} \sigma_j(\mathbf{W}_B^k)\mathbf{u}_j^k \mathbf{v}_j^k, \tag{14}$$

where d_k is the rank of $\mathbf{W}_B^k$, $\sigma_j(\mathbf{W}_B^k)$ is the j-th biggest singular value, $\mathbf{u}_j^k$ and $\mathbf{v}_j^k$ are left and singular vectors, respectively [46]. In Eq. 13, the first term $\mathbf{M}$ is the same as the derivative of the loss function of general binarization method with reducing quantization error. As for the second term, based on Eq. 14, it can be seen as the regularization term penalizing the general binarization loss with an adaptive regularization coefficient $\gamma \triangleq \lambda\beta^{k-L}(\frac{\|\mathbf{RM}_F^k\|_{SN}}{\|\mathbf{RM}_B^k\|_{SN}})$ (More detailed derivation can be found in the supplemental materials). Note that even we analyze the regularization property under the concept of SVD, we do not actually use SVD in our algorithm. And Eq. 13 and 14 only demonstrate that *LCR* regularization is related to the biggest singular value and its corresponding singular vectors. The *LCR* Algorithm 1 only uses the Power Iteration (see Algorithm in the Supplemental Materials) within less iteration steps (5 in practice) to approximate the biggest singular value.

Discussion on Retention Matrix. Here, we would like to give a straightforward explanation of why optimizing *LCR* Loss in Eq. 11 is equivalent to retaining Lipschitz continuity of BNN. Since the Lipschitz constant of a network $\|f\|_{Lip}$ can be upper-bounded by a set of spectral norms of weight matrices, *i.e.* $\{\|\mathbf{W}_F^k\|_{SN}\}$ (see Eq. 3-5), we aim at retaining the spectral norms of binary weight matrices, instead of targeting on the network itself. And because Eq. 7 to 9 derive $\|\mathbf{RM}_F^k\|_{SN} = \|\mathbf{W}_F^k\|_{SN}$ and $\|\mathbf{RM}_B^k\|_{SN} = \|\mathbf{W}_B^k\|_{SN}$, we only need to calculate the spectral norm of our designed Retention Matrix $\|\mathbf{RM}_B^k\|_{SN}$. Finally, minimizing Eq. 11 equals to enforcing $\|\mathbf{RM}_B^k\|_{SN} \longrightarrow \|\mathbf{RM}_F^k\|_{SN}$, which retains the spectral norm (Lipschitz continuity) of BNN. Therefore, the BNNs trained by our method have better performance, because the Lipschitz continuity is retained, which can smooth the BNNs.

Differences with Spectral Normalization (SN) and Defensive Quantization (DQ). There are two major differences: (i) In contrast to SN and DQ directly calculating the spectral norm with weight matrix, our method compute the spectral norm of specifically designed Retention Matrix to approximate the targeted spectral norms by leveraging the activations in BNNs. In this way, we can approximate the targeted yet inaccessible Lipschitz constant of binary networks as shown in Fig. 1 (a), in which the weight matrix is extremely sparse. Particularly, instead of layer-wisely calculating the spectral norm of weight matrix proposed in SN, our method does *not rely on weight matrix* since the calculation can be done using only the in/out activations (Eq. 8). (ii) To tackle the training architecture complexity, our designed Retention Matrix gives flexibility to regularize BNNs via utilizing Lipschitz constant in a module manner (*e.g.*, residual blocks in ResNet [17]), instead of calculating the spectral norm and normalizing the weight matrix to 1 for each layer as shown in Fig. 1 (b). Benefit from module-wise simplification, total computation cost of our method is much lower compared with SN and DQ.

4 Experiments

In this section, we conduct experiments on the image classification. Following popular setting in most studies[40,31], we use the CIFAR-10 [26] and the ImageNet ILSVRC-2012 [26] to validate the effectiveness of our proposed binarization method. In addition to comparing our method with the state-of-the-art methods, we design a series of ablative studies to verify the effectiveness of our proposed regularization technique. All experiments are implemented using PyTorch [38]. We use one NVIDIA GeForce 3090 GPU when training on the CIFAR-10 dataset, and four GPUs on the ImageNet dataset.

Experimental Setup. On CIFAR-10, the BNNs are trained for 400 epochs, batch size is 128 and initial learning rate is 0.1. We use SGD optimizer with the momentum of 0.9, and set weight decay is 1e-4. On ImageNet, the binary models are trained the for 120 epochs with a batch size of 256. We use cosine learning rate scheduler, and the learning rate is initially set to 0.1. All the training and testing settings follow the codebases of IR-Net [40] and RBNN [31].

4.1 CIFAR

CIFAR-10 [25] is the most widely-used image classification dataset, which consists of 50K training images and 10K testing images of size 32×32 divided into 10 classes. For training, 10,000 training images are randomly sampled for validation

Table 1. Top-1 and Top-5 accuracy on ImageNet. † represents the architecture which varies from the standard ResNet architecture but in the same FLOPs level.

Topology	Method	BW (W/A)	Top-1 (%)	Top-5 (%)
ResNet-18	Baseline	32/32	69.6	89.2
	ABC-Net [32]	1/1	42.7	67.6
	XNOR-Net [41]	1/1	51.2	73.2
	BNN+ [8]	1/1	53.0	72.6
	DoReFa [51]	1/2	53.4	-
	BiReal [33]	1/1	56.4	79.5
	XNOR++ [6]	1/1	57.1	79.9
	IR-Net [40]	1/1	58.1	80.0
	ProxyBNN [18]	1/1	58.7	81.2
	Ours	1/1	**59.6**	**81.6**
	Baseline	32/32	69.6	89.2
	SQ-BWN [11]	1/32	58.4	81.6
	BWN [41]	1/32	60.8	83.0
	HWGQ [29]	1/32	61.3	83.2
	SQ-TWN [11]	2/32	63.8	85.7
	BWHN [23]	1/32	64.3	85.9
	IR-Net [40]	1/32	66.5	85.9
	Ours	1/32	**66.9**	**86.4**
ResNet-34	Baseline	32/32	73.3	91.3
	ABC-Net [32]	1/1	52.4	76.5
	Bi-Real [33]	1/1	62.2	83.9
	IR-Net [40]	1/1	62.9	84.1
	ProxyBNN [18]	1/1	62.7	84.5
	Ours	1/1	**63.5**	**84.6**
Variant ResNet	ReActNet† [34]	1/1	69.4	85.5
	Ours†	1/1	**69.8**	**85.7**

Table 2. Top-1 accuracy (%) on CIFAR-10 (C-10) test set. The higher the better. W/A denotes the bit number of weights/activations.

Topology	Method	Bit-width (W/A)	Acc. (%)
ResNet-18	Baseline	32/32	93.0
	RAD [10]	1/1	90.5
	IR-Net [40]	1/1	91.5
	Ours	1/1	**91.8**
ResNet-20	Baseline	32/32	91.7
	DoReFa [51]	1/1	79.3
	DSQ [13]	1/1	84.1
	IR-Net [40]	1/1	85.5
	IR-bireal [40]	1/1	86.5
	LNS [15]	1/1	85.7
	SLB [48]	1/1	85.5
	Ours	1/1	**86.0**
	Ours-bireal	1/1	**87.2**
	Baseline	32/32	91.7
	DoReFa [51]	1/32	90.0
	DSQ [13]	1/32	90.1
	IR-Net [40]	1/32	90.2
	LNS [15]	1/32	90.8
	SLB [48]	1/32	90.6
	Ours	1/32	**91.2**

and the rest images are for training. Data augmentation strategy includes random crop and random flipping as in [17] during training. For testing, we evaluate the single view of the original image for fair comparison.

For ResNet-18, we compare with RAD [10] and IR-Net [40]. For ResNet-34, we compare with LNS [15] and SLB [48], *etc.* As the Table 1 presented, our method constantly outperforms other methods. *LCR*-BNN achieves 0.3%, 0.7% and 0.6% performance improvement over ResNet-18, ResNet-20 and ResNet-20 (without binarizing activations), respectively. In addition, our method also validate the effectiveness of bi-real structure [33]. When turning on the bi-real module, IR-Net achieves 1.0% accuracy improvements yet our method improves 1.2%.

4.2 ImageNet

ImageNet [9] is a larger dataset with 1.2 million training images and 50k validation images divided into 1,000 classes. ImageNet has greater diversity, and its image size is 469×387 (average). The commonly used data augmentation strategy including random crop and flipping in PyTorch examples [38] is adopted for training. We report the single-crop evaluation result using 224×224 center crop from images.

For ResNet-18, we compare our method with XNOR-Net [41], ABC-Net [32], DoReFa [51], BiReal [33], XNOR++ [6], IR-Net [40], ProxyBNN [18]. For ResNet-34, we compare our method with ABC-Net [32], BiReal [33], IR-Net [40], ProxyBNN [18]. As demonstrated in Table 2, our proposed method also outperforms other methods in both top-1 and top-5 accuracy on the ImageNet. Particularly, *LCR*-BNN achieves 0.9% Top-1 accuracy improvement with ResNet-18 architecture, compared with STOA method ProxyBNN [18], as well as 0.6% Top-1 accuracy improvement with ResNet-34 architecture, compared with state-of-the-art method ProxyBNN [40]. Apart from those methods implemented on standard ResNet architectures, by adding our Lipschitz regularization module on ResNet-variant architecture, ReActNet [34], we also observe the accuracy improvement. Note that the training setting of adding our *LCR* module on ReActNet is also different based on the codebase of ReActNet.

4.3 Ablation Study

In this section, the ablation study is conducted on CIFAR-10 with ResNet-20 architecture and on ImageNet with ResNet-18. The results are presented in Table 4. By piling up our regularization term on IR-Net [40] and ReActNet [34], our method achieves 1.2% and 0.4% improvement on ImageNet, respectively. Note that ReActNet is a strong baseline with a variant ResNet architecture. We also study the effect of hyper-parameter λ in loss function on CIFAR. As shown in Fig 3, we can observe that the performance improves with λ increasing. Both experiments validate the effectiveness of our method. Apart from that, to investigate the regularization property of our method, we visualize several training and testing curves with various settings. Due to the space limitation, we put those demonstrations in the supplemental materials.

4.4 Further Analysis

Computational Cost Analysis. In Table 5, we separate the number of binary operations and floating point operations, including all types of operations such as skip structure, max pooling, *etc.* It shows that our method leaves the number of BOPs and number of FLOPs constant in the model inference stage, even though our method is more computational expensive in the training stage. Thus, our Lipschitz regularization term does not undermine the main benefit of the network binarization, which is to speed up the inference of neural networks.

Weight Distribution Visualization. To validate the effectiveness of our proposed method from the perspective of weight distribution, we choose our *LCR*-BNN and IR-Net to visualize the distribution of weights from different layers. For fair comparison, we randomly pick up 10,000 parameters in each layer to formulate the Figure 2. Compared with IR-Net, the BNN trained by our method

Table 3. Effect of hyper-parameter λ in loss function. Higher is better.

Topology \ $\log_2 \lambda$	$\lambda = 0$	-1	0	1	2	3
ResNet-18	85.9	86.2	87.9	90.1	91.2	**91.8**
ResNet-20	83.9	83.7	84.5	85.9	**87.2**	86.5

Table 4. Ablation Study of LCR-BNN.

Dataset	Method	Acc(%)
CIFAR	Full Precision	91.7
	IR-Net [40] (w/o BiReal)	85.5
	IR-Net + LCR (w/o BiReal)	86.0
	IR-Net [40] (w/ BiReal)	86.5
	IR-Net + LCR (w/o BiReal)	87.2
ImageNet	Full Precision	69.6
	IR-Net [40] (w/o BiReal)	56.9
	IR-Net + LCR (w/o BiReal)	58.4
	IR-Net [40] (w/ BiReal)	58.1
	IR-Net + LCR	59.6
	ReActNet	69.4
	ReActNet + LCR	69.8

Table 5. FLOPS and BOPS for ResNet-18

Method	BOPS	FLOPS
BNN [24]	1.695×10^9	1.314×10^8
XNOR-Net [41]	1.695×10^9	1.333×10^8
ProxyBNN [18]	1.695×10^9	1.564×10^8
IR-Net [40]	1.676×10^9	1.544×10^8
Ours	1.676×10^9	1.544×10^8
Full Precision	0	1.826×10^9

Table 6. mCE on ImageNet-C. Lower is better.

Method	mCE (%)
IR-Net [40]	89.2
IR-Net + LCR (ours)	84.9 $\downarrow$
RBNN [31]	87.5
RBNN + LCR (ours)	84.8 $\downarrow$
ReActNet [34]	87.0
IR-Net + LCR (ours)	84.9 $\downarrow$

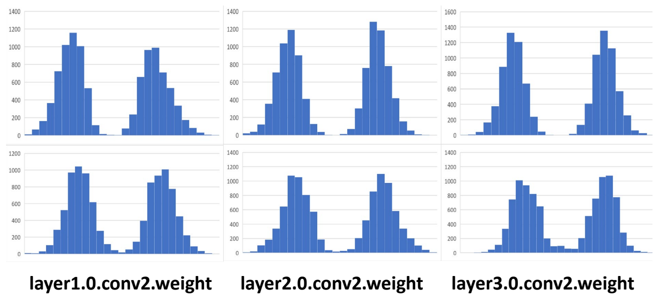

Fig. 2. Histograms of weights (before binarization) of the IR-Net [40] and *LCR*-BNN with ResNet-18 architecture. The first row shows the results of the IR-Net, and the second row shows the results of ours. The BNN trained by our method has smoother weight distribution.

possesses smoother weight distribution, which correspondingly helps our method achieve 1.6% accuracy improvement on ImageNet as listed in Table 2. More precisely, the standard deviation of the distribution of the IR-Net is 1.42, 28% higher than ours 1.11, in the layer3.0.conv2 layer.

Robustness Study on ImageNet-C. ImageNet-C [19] becomes the standard dataset for investigation of model robustness, which consists of 19 different types of corruptions with five levels of severity from the noise, blur, weather and digital categories applied to the validation images of ImageNet (see Samples in Supplemental Materials). We consider all the 19 corruptions at the highest severity level (severity = 5) and report the mean top-1 accuracy. We use Mean Corruption Error (mCE) to measure the robustness of models on this dataset. We freeze the backbone for learning the representations of data *w.r.t.* classification task, and

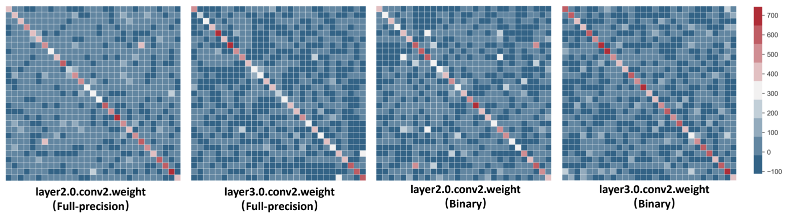

Fig. 3. Correlation maps for reflecting independence assumption in Eq. 7.

only fine-tune the task-specific heads over the backbone (*i.e.* linear protocol). The results in Table 6 prove that add *LCR* on the existing methods can improve the robustness of binary models.

Independence Assumption Reflection. The assumption used in Eq. 7 is the core of our method derivation, as it theoretically supports the approximation of the spectral norms of weight matrix with the designed retention matrix. Thus, we investigate this assumption by visualizing the correlation matrix of feature maps in the same batch. Specifically, we visualise the correlation matrices of full-precision and binary activations, where red stands for two activations are similar and blue *vice versa.* As shown in Fig 3, we can clearly observe that an activation is only correlated with itself, which largely testify this assumption. Besides, we also design another mechanism to use this assumption properly. We set a coefficient β greater than 1 to give more weight on latter layer's features such that they contribute more to $\mathcal{L}_{Lip}$ (Eq. 11). As in neural network, the feature maps of latter layers have stronger mutual linear independence [2].

5 Conclusion

In this paper, we introduce Lipschitz continuity to measure the robustness of BNN. Motivated by this, we propose *LCR*-BNN to retain the Lipschitz constant serving as a regularization term to improve the robustness of binary models. Specifically, to bypass the NP-hard Lipschitz constant computation in BNN, we devise the Retention Matrices to approximate the Lipschitz constant, and then constrain the Lipschitz constants of those Retention Matrices. Experimental results demonstrate the efficacy of our method.

Ethical Issues. All datasets used in our paper are open-source datasets and do not contain any personally identifiable or sensitive personally identifiable information. **Limitations.** Although our method achieve SoTA, adding it on existing method costs more time (around 20% more) to train BNN, which is the obvious limitation of our method.

Acknowledgements. This research was partially supported by NSF CNS-1908658 (ZZ,YY), NeTS-2109982 (YY), Early Career Scheme of the Research Grants Council (RGC) of the Hong Kong SAR under grant No. 26202321 (DX), HKUST Startup Fund No. R9253 (DX) and the gift donation from Cisco (YY). This article solely reflects the opinions and conclusions of its authors and not the funding agents.

References

1. Aharon, M., Elad, M., Bruckstein, A.: K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing (2006) 6
2. Alain, G., Bengio, Y.: Understanding intermediate layers using linear classifier probes. arXiv preprint arXiv:1610.01644 (2016) 14
3. Bartlett, P.L., Foster, D.J., Telgarsky, M.J.: Spectrally-normalized margin bounds for neural networks. In: NeurIPS (2017) 2, 3
4. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv:1308.3432 (2013) 3, 6, 9
5. Bulat, A., Martinez, B., Tzimiropoulos, G.: Bats: Binary architecture search. In: ECCV (2020) 6
6. Bulat, A., Tzimiropoulos, G.: Xnor-net++: Improved binary neural networks. In: BMVC (2019) 2, 3, 11, 12
7. Courbariaux, M., Bengio, Y., David, J.P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: NeurIPS (2016) 5
8. Darabi, S., Belbahri, M., Courbariaux, M., Nia, V.P.: Bnn+: Improved binary network training. CoRR (2018) 11
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009) 11
10. Ding, R., Chin, T.W., Liu, Z., Marculescu, D.: Regularizing activation distribution for training binarized deep networks. In: CVPR (2019) 11
11. Dong, Y., Ni, R., Li, J., Chen, Y., Zhu, J., Su, H.: Learning accurate low-bit deep neural networks with stochastic quantization. In: BMVC (2017) 11
12. Golub, G.H., Van der Vorst, H.A.: Eigenvalue computation in the 20th century. JCAM (2000) 8
13. Gong, R., Liu, X., Jiang, S., Li, T., Hu, P., Lin, J., Yu, F., Yan, J.: Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In: ICCV (2019) 11
14. Goodfellow, I., Bengio, Y., Courville, A., Bengio, Y.: Deep learning (2016) 6
15. Han, K., Wang, Y., Xu, Y., Xu, C., Wu, E., Xu, C.: Training binary neural networks through learning with noisy supervision. In: ICML (2020) 11
16. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In: ICLR (2016) 1
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 6, 7, 10, 11
18. He, X., Mo, Z., Cheng, K., Xu, W., Hu, Q., Wang, P., Liu, Q., Cheng, J.: Proxybnn: Learning binarized neural networks via proxy matrices. In: CVPR (2020) 2, 3, 11, 12, 13
19. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. In: ICLR (2019) 13
20. Heo, B., Kim, J., Yun, S., Park, H., Kwak, N., Choi, J.Y.: A comprehensive overhaul of feature distillation. In: ICCV (2019) 1
21. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NeurIPS (2014) 1
22. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: NeurIPS (2017) 7

23. Hu, Q., Wang, P., Cheng, J.: From hashing to cnns: Training binary weight networks via hashing. In: AAAI (2018) 11
24. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. In: NeurIPS (2016) 1, 2, 3, 6, 13
25. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 10
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS (2012) 1, 10
27. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature (2015) 1
28. LeCun, Y., Denker, J., Solla, S.: Optimal brain damage. In: NeurIPS (1989) 1
29. Li, Z., Ni, B., Zhang, W., Yang, X., Gao, W.: Performance guaranteed network acceleration via high-order residual quantization. In: ICCV (2017) 11
30. Lin, J., Gan, C., Han, S.: Defensive quantization: When efficiency meets robustness. arXiv preprint arXiv:1904.08444 (2019) 2, 3, 8
31. Lin, M., Ji, R., Xu, Z., Zhang, B., Wang, Y., Wu, Y., Huang, F., Lin, C.W.: Rotated binary neural network. In: NeurIPS (2020) 2, 10, 13
32. Lin, X., Zhao, C., Pan, W.: Towards accurate binary convolutional neural network. In: NeurIPS (2017) 11, 12
33. Liu, Z., Luo, W., Wu, B., Yang, X., Liu, W., Cheng, K.T.: Bi-real net: Binarizing deep network towards real-network performance. IJCV (2020) 3, 11, 12
34. Liu, Z., Shen, Z., Savvides, M., Cheng, K.T.: Reactnet: Towards precise binary neural network with generalized activation functions. In: ECCV (2020) 2, 6, 11, 12, 13
35. Luxburg, U.v., Bousquet, O.: Distance-based classification with lipschitz functions. JMLR (2004) 2, 3
36. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y.: Spectral normalization for generative adversarial networks. In: ICLR (2018) 2, 3, 4, 5, 6, 8
37. Neyshabur, B., Bhojanapalli, S., McAllester, D., Srebro, N.: Exploring generalization in deep learning. In: NeurIPS (2017) 2, 3, 4
38. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS (2019) 10, 11
39. Qin, H., Gong, R., Liu, X., Bai, X., Song, J., Sebe, N.: Binary neural networks: A survey. PR (2020) 3
40. Qin, H., Gong, R., Liu, X., Shen, M., Wei, Z., Yu, F., Song, J.: Forward and backward information retention for accurate binary neural networks. In: CVPR (2020) 1, 10, 11, 12, 13
41. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: ECCV (2016) 3, 11, 12, 13
42. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: NeurIPS (2015) 1
43. Rosca, M., Weber, T., Gretton, A., Mohamed, S.: A case for new neural network smoothness constraints. In: NeurIPS Workshop (2020) 5
44. Santurkar, S., Tsipras, D., Ilyas, A., Madry, A.: How does batch normalization help optimization? In: NeurIPS (2018) 8
45. Scaman, K., Virmaux, A.: Lipschitz regularity of deep neural networks: analysis and efficient estimation. In: NeurIPS (2018) 3, 5
46. Shang, Y., Duan, B., Zong, Z., Nie, L., Yan, Y.: Lipschitz continuity guided knowledge distillation. In: ICCV (2021) 4, 5, 6, 7, 9

47. Virmaux, A., Scaman, K.: Lipschitz regularity of deep neural networks: analysis and efficient estimation. In: NeurIPS (2018) 4, 5
48. Yang, Z., Wang, Y., Han, K., Xu, C., Xu, C., Tao, D., Xu, C.: Searching for low-bit weights in quantized neural networks. In: NeurIPS (2020) 11
49. Yoshida, Y., Miyato, T.: Spectral norm regularization for improving the generalizability of deep learning. arXiv:1705.10941 (2017) 2, 4
50. Zhang, B., Cai, T., Lu, Z., He, D., Wang, L.: Towards certifying robustness using neural networks with l-dist neurons. In: ICML (2021) 4
51. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv:1606.06160 (2016) 11, 12