ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-6654-0540-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICASSP43922.2022.9747328

DYNAMIC BINARY NEURAL NETWORK BY LEARNING CHANNEL-WISE THRESHOLDS

Jiehua Zhang[1], Zhuo Su[1], Yanghe Feng[2], Xin Lu[2], Matti Pietikäinen[1], Li Liu[2,1]*

[1]CMVS, University of Oulu [2] National University of Defense Technology

ABSTRACT

Binary neural networks (BNNs) constrain weights and activations to +1 or -1 with limited storage and computational cost, which is hardware-friendly for portable devices. Recently, BNNs have achieved remarkable progress and been adopted into various fields. However, the performance of BNNs is sensitive to activation distribution. The existing BNNs utilized the Sign function with predefined or learned static thresholds to binarize activations. This process limits representation capacity of BNNs since different samples may adapt to unequal thresholds. To address this problem, we propose a dynamic BNN (DyBNN) incorporating dynamic learnable channel-wise thresholds of Sign function and shift parameters of PReLU. The method aggregates the global information into the hyper function and effectively increases the feature expression ability. The experimental results prove that our method is an effective and straightforward way to reduce information loss and enhance performance of BNNs. The DyBNN based on two backbones of ReActNet (MobileNetV1 and ResNet18) achieve 71.2% and 67.4% top1-accuracy on ImageNet dataset, outperforming baselines by a large margin (*i.e.*, 1.8% and 1.5% respectively).

***Index Terms*—** deep learning, binary neural networks, network compression, computer vision, image analysis

1. INTRODUCTION

With the great progress made in deep learning in recent years, convolutional neural networks (CNNs) have achieved state-of-art performance in a broad range of fields. However, the existing CNNs require massive computation and storage resources to achieve high performance, which is not hardware-friendly to resource-limited devices. The Binary neural networks (BNNs), also known as 1-bit CNN, has two key advantages: 1) it constrains weights and activations to +1 or -1 to achieve a 32x memory compression ratio; 2) it utilizes XNOR and PopCount operations to replace computationally expensive multiply-add, providing a 58x computational reduction on a CPU [1, 2, 3, 4, 5, 6, 7, 8]. Due to these attractive characteristics of BNNs, it has been regarded as one of the most essential neural network compression methods to have the potential for direct deployment on next-generation hardware [4].

* denotes corresponding author: li.liu@oulu.fi

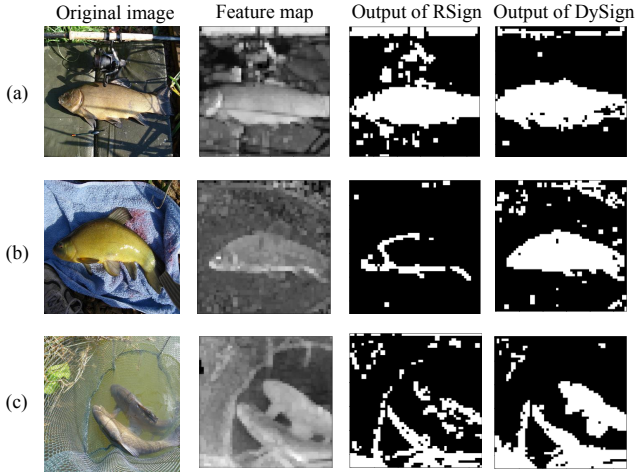

Fig. 1. The output of different samples after RSign and $DySign$. A set of fixed thresholds in RSign is insufficient to diverse samples. For (a), the fixed threshold can retain the feature information of object. For (b) and (c), the feature information significantly decrease.

Despite these advantages of BNNs, the enormous accuracy gap exists between BNNs and real-valued CNN since binarized operation limits model capacity and leads to a significant information loss of feature maps. To address this problem, Bi-RealNet [5] introduced a shortcut operation to increase value range of network, achieving remarkable performance improvement compared with XNOR-Net [9]. Based on this, Liu *et al.* [4] proved that BNNs are sensitive to activation distribution shift. A small distribution value offset close to zero would cause the binarized feature map to have a distinct appearance, leading to significant degradation of predictive accuracy. They proposed a generalization of Sign function (RSign) and PReLU function (RPReLU) to shift the distribution of feature maps. The proposed ReActNet based on MobileNetV1 [10] achieved the top-1 accuracy closed to real-valued network on ImageNet dataset [11] with lower computational cost.

However, the Sign functions in these methods apply predefined or learned **static** thresholds for adjusting activation distribution. They process the different inputs in a fixed way. The distribution of samples exists a huge gap, which means fixed shift parameters are incapable of adapting to a broad

range of samples. Shown as in Figure 1, the fixed threshold fits with image (a) but leads to the disastrous information loss for image (b) and (c). Inspired by [12], we introduce dynamic learnable shift parameters based on the input feature distribution for BNNs to enhance the feature expression ability.

Our contributions are summarized as follows:

- We propose Dynamic Binary Neural Network (DyBNN) to generate diverse shift parameters based on feature distribution of the input itself. The DyBNN incorporates channel-wise **dynamic** learnable thresholds of Sign function ($DySign$) and shift parameters of PReLU ($DyPReLU$). These parameters are both learned by a dynamic learning branch consisting of a global average pooling layer and two fully-connected layers. The global information of each channel is aggregated into the hyper function and generates a set of learning shift parameters to enhance the feature expression ability.
- We demonstrate the effectiveness of DyBNN on the ImageNet dataset. Without bells and whistles, simply replacing static RSign and RPReLU with $DySign$ and $DyPReLU$ in two networks (ReActNet and ReActNet based on ResNet18) achieves 71.2% and 67.4% top1-accuracy, outperforming strong baselines 1.8% and 1.5% by a large margin, respectively.

2. METHODOLOGY

In this section, we first introduce the BNNs and then present the influence of fixed thresholds in Sign function. Finally, we illustrate our DyBNN how to reduce the information loss and enhance the model expression ability.

2.1. Preliminary

For the convolution process in the BNNs, we denote the weights and features in the ℓ-layer as W^ℓ and F^ℓ. The input of the ℓ+1-layer can be expressed as:

$$F^{\ell+1} = \phi^\ell(Sign(W^\ell) \otimes Sign(F^\ell)) \tag{1}$$

$$Sign(x) = \begin{cases} +1, & x > 0, \\ -1, & x \leq 0. \end{cases} \tag{2}$$

where $\otimes$ denotes convolutional operation, the $\phi^\ell(\cdot)$ denotes the nonlinear operation in the ℓ-layer such as ReLU, PReLU, and BN layer. To reduce the memory saving and computational cost, the BNNs constrain the W^ℓ and F^ℓ to -1 or +1 by Sign function.

However, BNNs are sensitive to distribution shift since the activations are constrained to +1 and -1. A slight distribution shift in the input feature map can significantly affect the output of Sign function. To address this problem, the ReActNet [4] utilized learnable channel-wise thresholds to shift the feature maps, which is defined as RSign:

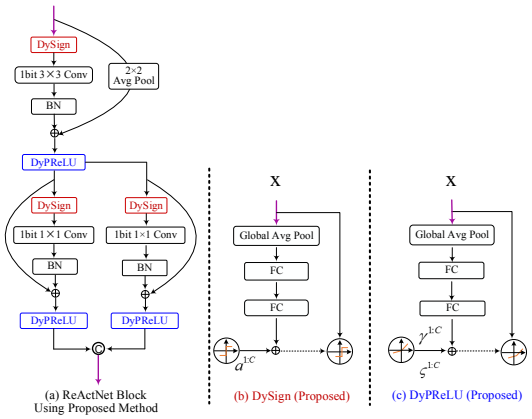

Fig. 2. The basic modules of proposed DyBNN (built on the ReActNet).

$$RSign(x_i) = \begin{cases} +1, & x_i > a_i, \\ -1, & x_i \leq a_i. \end{cases} \tag{3}$$

where x_i denotes the element of i-th channel in input feature map, a_i denotes the threshold of i-th channel, which illustrates that the threshold can vary for different channels. The PReLU function is also processed by this operation to reshape the feature map, which can be expressed as:

$$RPReLU(x_i) = \begin{cases} x_i - \gamma_i + \zeta_i, & x_i > \gamma_i, \\ \beta_i(x_i - \gamma_i) + \zeta_i, & x_i \leq \gamma_i. \end{cases} \tag{4}$$

where γ_i and ζ_i denote the learnable shift parameters for reshaping the distribution. By introducing the activation distribution shift and reshape, ReActNet achieves a significant increase in image classification.

2.2. Dynamic binary neural network

Although ReActNet achieves remarkable performance, the feature information loss still negatively affects the performance of the BNNs since the RSign adopts a **static** channel-wise thresholds on the input feature maps. Each sample contains a specific feature expression, which means that fixed thresholds cannot be adapted to a broad range of samples (See Fig 1). The threshold parameters should be adaptively generated to shift the feature maps based on the characteristics of feature.

Based on the aforementioned statement, we propose the Dynamic Learning Sign Function ($DySign$) to shift the feature maps. For a given input feature map X with C channels, the $DySign$ is defined as a function with learnable parameters $f(x)$, which adapts to the input X (See Fig 2). The hyper function $f(x)$ computes the threshold for each ith-channel feature X^i, which can be expressed as:

$$\alpha = f(X) = f_2(f_1(\frac{1}{HW}\sum_{H,W} X)) \tag{5}$$

where H and W denote the height and width of input feature map X, $f_1 \in \mathbb{R}^{C\times\frac{C}{16}}$ and $f_2 \in \mathbb{R}^{\frac{C}{16}\times C}$ denote two fully-connected (FC) layers. The $DySign$ binarized the feature maps based on the parameters $\alpha^{1:C}$ from $f(x)$ as following:

$$DySign(x_i) = \begin{cases} +1, & x_i > \alpha_i, \\ -1, & x_i \leq \alpha_i. \end{cases} \alpha_i \in \alpha^{1:C} \tag{6}$$

where α^i denotes the threshold of i-th channel, which is the i-th element of output vector from $f(x)$. The $DySign$ adopts a SEBlock [13] to learn a set of channel-wise thresholds from the input feature maps for Sign function. The block of $DySign$ can be observed in Fig 2. The process is "$input \rightarrow GAP \rightarrow FClayer \rightarrow FClayer$". To avoid over-fit and reduce extra computational cost, the reduction ratio between two linear layers is set as 16. This hyper function encodes the global information of input F to determine appropriate thresholds of each channel. $DySign$ reduces the feature information loss in BNNs and enables significantly more representation power than using standard Sign function.

We also handle the RPReLU function [4] in the same way. The shift parameters $\gamma^{1:C}$ and $\zeta^{1:C}$ are learnable based on input feature maps ($DyPReLU$). The hyper function $f(\cdot)$ generates the shift parameters for each channel in input feature maps. This process can dynamically shift and reshape feature the distribution, which is an effective and simple way to increase model capacity.

Essentially, the $DySign$ can learn the most suitable thresholds $\alpha^{1:C}$ to binarize the input feature map. The threshold parameters can be dynamically adjusted for different input feature maps, which can effectively limit the feature information loss after binarization. For $DyPReLU$, the $\gamma^{1:C}$ and $\zeta^{1:C}$ can be easily understood as these parameters are dynamically generated to obtain better output distribution. By introducing these functions, the aforementioned problem risen by static parameters can be eliminated. The BNNs can retain more object information and learn more meaningful features. In the experiment section, we will show that dynamic learning distribution is an effective and straightforward way to boost the performance of BNNs.

2.3. Model architecture

For model architecture, The ResNet18 [14] and MobileNetV1 [10] are built as the backbones following ReActNet [4]. In the ReActNet (Shown as in Figure 2), the 3×3 depthwise and 1×1 pointwise convolution are replaced by standard 3×3 and 1×1 convolution. The duplication and concatenation operations are designed for addressing the channel number difference. In our case, we simply replace the RSign and RPReLU functions in ReActNet with our $DySign$ and $DyPReLU$. The ReActNet can be regarded as the baseline in our experiment.

2.4. Computational complexity analysis

Following the calculation method in [15, 4], we calculate total operations (OPs), which consists of binary operations (BOPs) and floating point operations (FLOPs). The OPs can be obtained as:

$$OPs = \frac{BOPs}{64} + FLOPs \tag{7}$$

For our DyBNN, we do not introduce extra binary convolutional operations. Thus, the BOPs is same as ReActNet. The increased computational consumption mainly comes from floating-point operations in $DySign$ and $DyPReLU$, including one global average pooling layer and two fully-connected layers. For reducing introduced computational cost, we set the reduction ratio between two fully-connected layers as 16, which can limit the introduced model parameters and float operations. We denote the size of input feature map as $C \times H \times W$. The FLOPs for each RSign and RPReLU increase $C + \frac{C^2}{8}$ and $2 \times (C + \frac{C^2}{8})$. *The extra computational cost is small compared with the total cost.*

3. EXPERIMENT

To evaluate the performance of dynamic distribution learning on BNNs, we conduct experiments on ImageNet dataset [11]. In this section, we first introduce the training dataset and details. We then report the accuracy and OPs of DyBNN and compare them with state of the art methods. We analyze the impact of $DySign$ and $DyPReLU$ in the ablation study shown in Section 3.4.

3.1. Datasets and implementation details

The ILSVRC12 ImageNet classification dataset [11] is utilized to evaluate proposed method. The ILSVRC12 ImageNet classification dataset contains 1.2 million training images and 50,000 validation images across 1000 classes, which is more challenging than small datasets.

The training strategy utilizes the two-step training strategy described in [15]. In the first step, the network is trained from scratch with binary activations and real-valued weights. In the second step, the network inherits the weight from the first step and trained with binary activations and weights. For both steps, we follow the training scheme in [4]. We use Adam optimizer and a linear learning rate decay to optimize model. The initial learning rate is 5e-4, and batchsize is set to 256. We also train a quick version of DyBNN with the one-step training strategy.

3.2. Experiment on ImageNet

We compare the DyBNN with other state-of-the-art binarization methods in Table 1. Compared with ReActNet and

ReActNet-ResNet18, the DyBNN and DyBNN_ResNet18 achieve 1.8% and 1.5% increase, respectively. Due to the introduced global average pooling layer and fully-connected layer, the FLOPs increases slightly. For DyBNN, the OPs is 0.02×10^8 higher than ReAcNet (See Tabel 2), which is acceptable. With the limited computational cost increased, the DyBNN can outperform previous methods by a large margin, which illustrates the effectiveness of dynamic learnable shift parameters in BNNs.

Table 1. Compare of the top-1 accuracy with state-of-art methods. The W/A denote the number of bits in weight and activation quantization. The blue font denotes the comparing result with DyBNN and ReActNet based on ResNet18. The red font denotes the comparing result based on MobileNetV1.

Binary Method	W/A	Acc Top-1 (%)	Acc Top-5 (%)
BNN[6]	1/1	42.2	67.1
ABC-Net[16]	1/1	42.7	67.6
XNOR-Net[9]	1/1	51.2	69.3
DoReFa[17]	1/2	53.4	-
Bi-RealNet-18[5]	1/1	56.4	79.5
XNOR++[18]	1/1	57.1	79.9
IR-Net[19]	1/1	58.1	80.0
BONN[20]	1/1	59.3	81.6
NoisySupervision[21]	1/1	59.4	81.7
RBNN[22]	1/1	59.8	81.9
Real-to-Binary Net[15]	1/1	65.4	86.2
ReActNet-ResNet18[4]	1/1	65.9	86.5
ReActNet[4]	1/1	69.4	88.6
AdamBNN[23]	1/1	70.5	89.1
DyBNN-ResNet18(ours)	1/1	67.4(↑ 1.5)	87.4
DyBNN(ours)	1/1	71.2(↑ 1.8)	89.8

3.3. One-step Training

The DyBNN utilizes the two-step training strategy with 512 epochs in total, which is time-consuming. To simplify the training process, we also evaluate the quick version of DyBNN in one-step training. The result can be observed in Tabel 2. We evaluate DyBNN under following strategies: 1) distillation with distribution loss in [4]; 2) no distillation with CrossEntropy loss. Following other training details described in 3.1, DyBNN achieved 2.3% and 2.0% higher than ReActNet under the two training strategies mentioned above, which illustrated that DyBNN is an effective and straightforward way to boost performance of BNNs. We also observe that DyBNN achieves higher accuracy and faster convergence (See Fig 3). In this paper, The calculation of FLOPs contains BN, PReLU layers, so our reported OPs is higher than 0.87×10^8 in [4].

3.4. Impacts of DySign and DyPReLU

In this section, we analyze the individual effect of $DySign$. We conduct the experiment in the two-step training strategy following Section 3.1. The experimental result is shown in Table 3. DyBNN achieves 70.1% top-1 accuracy on the ImageNet without $DyPReLU$. Comparing with ReActNet, the accuracy increases by 0.7%, illustrating the effectiveness of our proposed $DySign$. $DySign$ promotes model performance by reducing feature information loss. Furthermore, $DyPReLU$ increases model expression ability by reshaping activation distribution of feature maps, which further improves model accuracy.

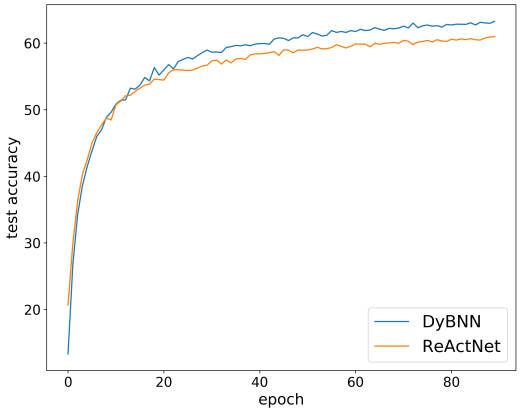

Fig. 3. The accuracy of DyBNN and ReActNet on ImageNet.

Table 2. The test result of a quick version for DyBNN. The ✓ denotes the model is trained with distributional loss and the real-valued ResNet34 is set as teacher model. No ✓ denotes the model is directly trained with Cross Entropy loss.

Network	distillation	OPs	Acc Top-1(%)
ReActNet		0.97×10^8	61.0
ReActNet	✓		64.3
DyBNN		0.99×10^8	**63.3**
DyBNN	✓		**66.3**

Table 3. The impact of *DySign* and *DyPReLU*.

Network	DySign	DyPReLU	Acc Top-1(%)
ReActNet			69.4
DyBNN	✓		70.1
DyBNN	✓	✓	71.2

4. CONCLUSION

In this paper, we have introduced the dynamic learning method into binary neural networks and propose DyBNN. DyBNN dynamically generates adaptive parameters to shift and reshape distribution activations. The experimental results show that DyBNN can reduce the feature information loss and significantly improve the performance of BNNs with a limited computational cost increase.

Acknowledgemens This work was partially supported by the Academy of Finland under grant 331883 and the National Natural Science Foundation of China under Grant 61872379. The authors also wish to acknowledge CSC IT Center for Science, Finland, for computational resources.

5. REFERENCES

[1] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe, "Binary neural networks: A survey," *PR*, vol. 105, pp. 107281, 2020. 1

[2] Jiaxin Gu, Ce Li, Baochang Zhang, Jungong Han, Xianbin Cao, Jianzhuang Liu, and David Doermann, "Projection convolutional neural networks for 1-bit cnns via discrete back propagation," *ICLR*, 2019. 1

[3] Daniel Soudry, Itay Hubara, and Ron Meir, "Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights," *NeurIPS*, 2014. 1

[4] Zechun Liu, Zhiqiang Shen, Marios Savvides, , and Kwang-Ting Cheng, "Reactnet: Towards precise binary neural net-work with generalized activation functions," *ECCV*, pp. 143–159, 2020. 1, 2, 3, 4

[5] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Chen, "Bi-real net: Enhancing the perfor-mance of 1-bit cnns with improved representational capabil-ity and advanced training algorithm," *ECCV*, pp. 722–737, 2018. 1, 4

[6] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," *arXiv preprint arXiv:1602.02830*, 2016. 1, 4

[7] Ziwei Wang, Jiwen Lu, Ziyi Wu, and Jie Zhou, "Learning efficient binarized object detectors with information compression," *IEEE TPAMI*, 2021. 1

[8] Ziwei Wang, Jiwen Lu, and Jie Zhou, "Learning channel-wise interactions for binary convolutional neural networks," *IEEE TPAMI*, vol. 43, pp. 3432–3445, 2021. 1

[9] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *ECCV*, pp. 525–542, 2016. 1, 4

[10] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017. 1, 3

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, San jeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, pp. 211–252, 2015. 1, 3

[12] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu, "Dynamic relu," *ECCV*, pp. 351–367, 2020. 2

[13] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation net-works," *CVPR*, pp. 7132–7141, 2018. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," *CVPR*, pp. 770–778, 2016. 3

[15] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tz-imiropoulos, "Training binary neural networks with real-to-binary convolutions," *ICLR*, 2019. 3, 4

[16] Xiaofan Lin, Cong Zhao, and Wei Pan, "Towards accurate binary convolutional neural network," *NeurIPS*, pp. 345–353, 2017. 4

[17] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016. 4

[18] Adrian Bulat and Georgios Tzimiropoulos, "Xnor-net++: Improved binary neural networks," *BMVC*, 2019. 4

[19] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song, "Forward and backward information retention for accurate binary neural networks," *CVPR*, pp. 2250–2259, 2020. 4

[20] Jiaxin Gu, Junhe Zhao, Xiaolong Jiang, Baochang Zhang, Jianzhuang Liu, Guodong Guo, and Rongrong Ji, "Bayesian optimized 1-bit cnns," *ICCV*, pp. 4909–4917, 2019. 4

[21] Kai Han, Yunhe Wang, Yixing Xu, Chunjing Xu, Enhua Wu, and Chang Xu, "Training binary neural networks through learning with noisy supervision," *ICML*, pp. 4017–4026, 2020. 4

[22] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin, "Rotated binary neural network," *NeurIPS*, 2020. 4

[23] Zechun Liu, Zhiqiang Shen, Shichao Li, Koen Helwegen, Dong Huang, and Kwang-Ting Cheng, "How do adam and training strategies help bnns optimization?," *ICML*, 2021. 4