AdaBin: Improving Binary Neural Networks with Adaptive Binary Sets

Zhijun Tu[1,2], Xinghao Chen[2(✉)], Pengju Ren[1(✉)], and Yunhe Wang[2]

[1] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
tuzhijun123@stu.xjtu.edu.cn, pengjuren@xjtu.edu.cn
[2] Huawei Noah's Ark Lab
{zhijun.tu, xinghao.chen, yunhe.wang}@huawei.com

Abstract. This paper studies the Binary Neural Networks (BNNs) in which weights and activations are both binarized into 1-bit values, thus greatly reducing the memory usage and computational complexity. Since the modern deep neural networks are of sophisticated design with complex architecture for the accuracy reason, the diversity on distributions of weights and activations is very high. Therefore, the conventional sign function cannot be well used for effectively binarizing full-precision values in BNNs. To this end, we present a simple yet effective approach called AdaBin to adaptively obtain the optimal binary sets $\{b_1, b_2\}$ $(b_1, b_2 \in \mathbb{R})$ of weights and activations for each layer instead of a fixed set (*i.e.*, $\{-1, +1\}$). In this way, the proposed method can better fit different distributions and increase the representation ability of binarized features. In practice, we use the center position and distance of 1-bit values to define a new binary quantization function. For the weights, we propose an equalization method to align the symmetrical center of binary distribution to real-valued distribution, and minimize the Kullback-Leibler divergence of them. Meanwhile, we introduce a gradient-based optimization method to get these two parameters for activations, which are jointly trained in an end-to-end manner. Experimental results on benchmark models and datasets demonstrate that the proposed AdaBin is able to achieve state-of-the-art performance. For instance, we obtain a 66.4% Top-1 accuracy on the ImageNet using ResNet-18 architecture, and a 69.4 mAP on PASCAL VOC using SSD300.

Keywords: Binary Neural Networks, Adaptive Binary Sets

1 Introduction

Deep Neural Networks (DNNs) have demonstrated powerful learning capacity, and are widely applied in various tasks such as computer vision [27], natural language processing [3] and speech recognition [22]. However, the growing complexity of DNNs requires significant storage and computational resources, which

---

✉ Corresponding author

(a) FLOPs vs. ImageNet accuracy
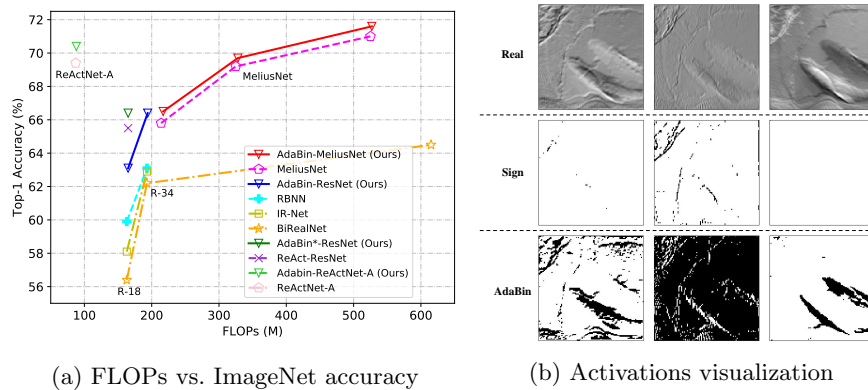

(b) Activations visualization

Fig. 1: (a) Comparisons with state-of-the-art methods. With a little extra computation, the proposed AdaBin achieves better results for various architectures such as ResNet, MeliusNet [5] and ReActNet [33]. (b) Visualization for activations of 2nd layer in ResNet-18 on ImageNet. *Real* denotes real-valued activations, *Sign* and *AdaBin* denote the binary methods of previous BNNs and ours.

makes the deployment of these deep models on embedded devices extremely difficult. Various approaches have been proposed to compress and accelerate DNNs, including low-rank factorization [44], pruning [10,19], quantization [12], knowledge distillation [11,23] and energy-efficient architecture design [9], *etc.* Among these approaches, quantization has attracted great research interests for decades, since the quantized networks with less bit-width require smaller memory footprint, lower energy consumption and shorter calculation delay. Binary Neural Networks (BNNs) are the extreme cases of quantized networks and could obtain the largest compression rate by quantizing the weights and activations into 1-bit values [18,37,41,43]. Different from the floating point matrix operation in traditional DNNs, BNNs replace the multiplication and accumulation with bit-wise operation XNOR and BitCount, which can obtain an about $64\times$ acceleration and $32\times$ memory saving [37]. However, the main drawback of BNNs is the severe accuracy degradation compared to the full-precision model, which also limits its application to more complex tasks, such as detection, segmentation and tracking.

According to the IEEE-754 standard, a 32-bit floating point number has 6.8×10^{38} unique states [1]. In contrast, a 1-bit value only has 2 states $\{b_1, b_2\}$, whose representation ability is very weak compared with that of the full-precision values, since there are only two kinds of the multiplication results of binary values as shown in Table 1a. To achieve a very efficient hardware implementation, the conventional BNN method [13] binarizes both the weights and the activations to either +1 or -1 with sign function. The follow-up approaches on BNNs have made tremendous efforts for enhancing the performance of binary network, but still restrict the binary values to a fixed set (*i.e.*, $\{-1, +1\}$ or $\{0, +1\}$) for all the

w＼a	-1	+1
-1	+1	-1
+1	-1	+1

w＼a	0	+1
-1	0	-1
+1	0	+1

w＼a	-1	+1
0	0	0
+1	-1	+1

w＼a	a_{b1}	a_{b2}
w_{b1}	$a_{b1}w_{b1}$	$a_{b2}w_{b1}$
w_{b2}	$a_{b1}w_{b2}$	$a_{b2}w_{b2}$

(a) BNN [13]   (b) SiBNN [38]   (c) SiMaN [30]   (d) AdaBin (Ours)

Table 1: The illustration on the feature representation ability of different binary schemes. The a represents the binarized input and the w represents binarized weights, respectively. $a_{b1}, a_{b2}, w_{b1}, w_{b2} \in \mathbb{R}$, which are not restricted to fixed values for different layers.

layers. Given the fact that the feature distributions in deep neural networks are very diverse, sign function can not provide binary diversity for these different distributions. To this end, we have to rethink the restriction of fixed binary set for further enhancing the capacity of BNNs.

Based on the above observation and analysis, we propose an **Ada**ptive **Bin**ary method (**AdaBin**) to redefine the binary values ($b_1, b_2 \in \mathbb{R}$) with their center position and distance, which aims to obtain the optimal binary set that best matches the real-valued distribution. We propose two corresponding optimization strategies for weights and activations. On one hand, we introduce an equalization method for the weights based on statistical analysis. By aligning the symmetrical center of binary distribution to real-valued distribution and minimizing the Kullback-Leibler divergence (KLD) of them, we can obtain the analytic solutions of center and distance, which makes the weight distribution much balanced. On the other hand, we introduce a gradient-based optimization method for the activations with a loss-aware center and distance, which are initialized in the form of sign function and trained in an end-to-end manner. As shown in Table 1, we present the truth tables of the multiplication results for binary values in different BNNs. Most previous BNNs binarize both the weights and activations into $\{-1, +1\}$ as shown in Table 1a. A few other methods [30,38] attempt to binarize weights and activations into $\{0, +1\}$, as shown in Table 1b and Table 1c. These methods result in 2 or 3 kinds of output representations. Table 1d illustrates the results of our proposed AdaBin method. The activations and weights are not fixed and could provide 4 kinds of output results, which significantly enhances the feature representation of binary networks as shown in Fig. 1b. Meanwhile, we can find that previous binary methods are the special cases of our AdaBin and we extend the binary values from ± 1 to the whole real number domain.

Furthermore, we demonstrate that the proposed AdaBin can also be efficiently implemented by XNOR and BitCount operations with negligible extra calculations and parameters, which could achieve $60.85\times$ acceleration and $31\times$ memory saving in theory. With only minor extra computation, our proposed AdaBin outperforms state-of-the-art methods for various architectures, as shown in Fig. 1a. The contributions of this paper are summarize as follow:

(1) We rethink the limitation of $\{-1, +1\}$ in previous BNNs and propose a simple yet effective binary method called AdaBin, which could seek suitable binary sets by adaptively adjusting the center and distance of 1-bit values.

(2) Two novel strategies are proposed to obtain the optimal binary sets of weights and activations for each layer, which can further close the performance gap between binary neural networks and full-precision networks.

(3) Extensive experiments on CIFAR-10 and ImageNet demonstrate the superior performance of our proposed AdaBin over state-of-the-art methods. Besides, though not tailored for object detection task, AdaBin also outperforms prior task-specific BNN methods by 1.9 mAP on PASCAL VOC dataset.

2   Related Work

Binary neural network was firstly introduced by [13]. They creatively proposed to binarize weights and activations with sign function and replace most arithmetic operations of deep neural networks with bit-wise operations. To reduce the quantization error, XNOR-Net [37] proposed a channel-wise scaling factor to reconstruct the binarized weights, which also becomes one of the most important components of the subsequent BNNs. ABC-Net [32] approximated full-precision weights with the linear combination of multiple binary weight bases and employed multiple binary activations to alleviate information loss. Inspired by the structures of ResNet [21] and DenseNet [25], Bi-Real Net [34] proposed to add shortcuts to minimize the performance gap between the 1-bit and real-valued CNN models, and BinaryDenseNet [6] improved the accuracy of BNNs by increasing the number of concatenate shortcut. IR-Net [36] proposed the Libra-PB, which can minimize the information loss in forward propagation by maximizing the information entropy of the quantized parameters and minimizing the quantization error with the constraint $\{-1, +1\}$. ReActNet [33] proposed to generalize the traditional sign and PReLU functions, denoted as RSign and RPReLU for the respective generalized functions, to enable explicit learning of the distribution reshape and shift at near-zero extra cost.

3   Binarization with Adaptive Binary Sets

In this section, we focus on how to binarize weights and activations respectively, and introduce a new non-linear module to enhance the capacity of BNNs.

We first give a brief introduction on the general binary neural networks. Given an input $a \in \mathbb{R}^{c \times h \times w}$ and weight $\mathrm{w} \in \mathbb{R}^{n \times c \times k \times k}$, then we can get the output $y \in \mathbb{R}^{n \times h' \times w'}$ by convolution operation as Eq. 1.

$$y = \mathrm{Conv}(a, \mathrm{w}). \tag{1}$$

To accelerate the inference process, previous BNNs always partition the input and weight into two clusters, -1 and $+1$ with sign function as Eq. 2.

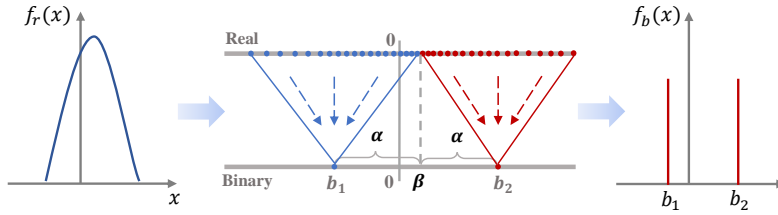$$\mathrm{Sign}(x) = \begin{cases} b_1 = -1, & x < 0 \\ b_2 = +1, & x \geq 0 \end{cases}. \tag{2}$$

Fig. 2: AdaBin quantizer. The middle represents the mapping from floating point distribution $f_r(x)$ to binary distribution $f_b(x)$. b_1 and b_2 are the two clusters, α and β are the distance and center, respectively.

Then the floating-point multiplication and accumulation could be replaced by bit-wise operation XNOR (denoted as $\odot$) and BitCount as Eq. 3, which will result in much less overhead and latency.

$$y = \text{BitCount}(a_b \odot w_b). \tag{3}$$

In our method, we do not constrain the binarized values to a fixed set like $\{-1, +1\}$. Instead we release b_1 and b_2 to the whole real number domain and utilize the proposed AdaBin quantizer, which could adjust the center position and distance of the two clusters adaptively as Eq. 4. In this way, the binarized distribution can best match the real-valued distribution:

$$\mathcal{B}(x) = \begin{cases} b_1 = \beta - \alpha, & x < \beta \\ b_2 = \beta + \alpha, & x \geq \beta \end{cases}, \tag{4}$$

where the α and β are the half-distance and center of the binary values b_1 and b_2. Fig. 2 shows the binarization of AdaBin, as we can see that, the data on the left of the center will be clustered into b_1 and the data on the right of the center will be clustered into b_2. The distance α and center β will change with different distributions, which help partition the floating point data into two optimal clusters adaptively. For the binarization of weights and activations, we exploit the same form of AdaBin but different optimization strategies.

3.1   Weight Equalization

Low-bit quantization greatly weaken the feature extraction ability of filter weights, especially for 1-bit case. Previous BNNs exploit different methods to optimize the binarized weights. XNOR-Net [37] minimizes the mean squared error (MSE) by multiplying a scale factor, and IR-Net [36] obtains the maximum information entropy by weight reshaping and then conduct the same operation as XNOR-Net. However, these methods can not get accurate quantization error between binarized data and real-valued data due to the following limitations.

Firstly, the center position of previous binarized values $\{-1, +1\}$ is always 0, which is not aligned with the center of original real-valued weights. Secondly, MSE is a simple metric to evaluate the quantization error but do not consider the distribution similarity between binarized data and real-valued data. On the contrary, the Kullback-Leibler divergence (KLD) is a measure on probability distributions [28] and is more accurate to evaluate the information loss than MSE. Therefore, we propose to minimize the KLD to achieve a better distribution-match. We apply the AdaBin for weights binarization as Eq. 5:

$$\mathrm{w}_b = \mathcal{B}(\mathrm{w}) = \begin{cases} \mathrm{w}_{b1} = \beta_w - \alpha_w, & \mathrm{w} < \beta_w \\ \mathrm{w}_{b2} = \beta_w + \alpha_w, & \mathrm{w} \geq \beta_w \end{cases}, \tag{5}$$

where α_w and β_w are distance and center of binarized weights, the binary elements of w_b in the forward is $\beta_w - \alpha_w$ and $\beta_w + \alpha_w$. And the KLD of real-valued distribution and binary distribution can be represented as Eq. 6.

$$D_{KL}(P_r||P_b) = \int_{x \in \mathrm{w} \& \mathrm{w}_b} P_r(x) \log \frac{P_r(x)}{P_b(x)} dx, \tag{6}$$

where the $P_r(x)$ and $P_b(x)$ denote the distribution probability of real-valued weights and binarized weights. In order to make the binary distribution more balanced, we need to align its symmetrical center (position of mean value) to the real-valued distribution, so that Eq. 7 can be obtained.

$$\beta_w = \mathbb{E}(\mathrm{w}) \approx \frac{1}{c \times k \times k} \sum_{m=0}^{c-1} \sum_{j=0}^{k-1} \sum_{i=0}^{k-1} \mathrm{w}_{m,j,i}. \tag{7}$$

Therefore, we can further infer that $P_b(\mathrm{w}_{b1}) = P_b(\mathrm{w}_{b2}) = 0.5$. Since there is no convinced formula of weight distribution for neural networks, it is difficult to calculate the Kullback-Leibler divergence explicitly. However, the weights in such networks typically assume a bell-shaped distribution with tails [2,4,45], and the both sides are symmetrical on the center, then we can get the α_w as Eq. 8, the detailed proof is in the supplementary.

$$\alpha_w = \frac{\|\mathrm{w} - \beta_w\|_2}{\sqrt{c \times k \times k}}, \tag{8}$$

where $\| \cdot \|_2$ denotes the ℓ_2-norm. In our method, the distance α_w and center β_w are channel-wise parameters for weight binarization, and updated along the real-valued weights during the training process. As shown in Figure ??, without distribution reshaping and the constraint that the center of binary values is 0, AdaBin could equalize the weights to make the binarized distribution best match the real-valued distribution. During the inference, we can decompose the binary weights matrix into 1-bit storage format as following:

$$\mathrm{w}_b = \alpha_w b_w + \beta_w, b_w \in \{-1, +1\}. \tag{9}$$

So that the same as the previous BNNs, our method can also achieve about 32× memory saving.

3.2   Gradient-based Activation Binarization

Activation quantization is a challenging task with low bit-width, and has much more impacts to the final performance than weight. HWGQ [8] proposed to address this challenge by applying a half-wave Gaussian quantization method, based on the observation that activation after Batch Normalization tends to have a symmetric, non-sparse distribution, that is close to Gaussian and ReLU is a half-wave rectifier. However, recent BNNs [35] proposed to replace the ReLU with PReLU [20], which could facilitate the training of binary networks. So that HWGQ can not be further applied because of this limitation. Besides, the distribution of real-valued activations is not as stable as weights, which keeps changing for different inputs. Therefore we can not extract the center and distance from the activations as Eq. 7 and Eq. 8, which brings extra cost to calculate them and will greatly weaken the hardware efficiency of binary neural networks during inference. In order to get the optimal binary activation during training, we propose a gradient-based optimization method to minimize the accuracy degradation arising from activation binarization. Firstly, we apply the AdaBin quantizer to activations as Eq. 10.

$$a_b = \mathcal{B}(a) = \begin{cases} a_{b1} = \beta_a - \alpha_a, & a < \beta_a \\ a_{b2} = \beta_a + \alpha_a, & a \geq \beta_a \end{cases}, \tag{10}$$

where α_a and β_a are the distance and center of binarized activations, and the binary set of a_b in the forward is $\{\beta_a - \alpha_a, \beta_a + \alpha_a\}$. To make the binary activations adapt to the dataset as much as possible during the training process, we set α_a and β_a as learnable variables, which could be optimized via backward gradient propagation as total loss decreases. In order to ensure that the training process can converge, we need to clip out the gradient of large activation values in the backward as Eq. 11.

$$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial \mathcal{L}}{\partial a_b} * \mathbb{1}_{|\frac{a-\beta_a}{\alpha_a}| \leq 1}, \tag{11}$$

where $\mathcal{L}$ denotes the output loss, a is the real-valued activation and a_b is the binarized activation, $\mathbb{1}_{|x| \leq 1}$ denotes the indicator function that equals to 1 if $|x| \leq 1$ is true and 0 otherwise. This functionality can be achieved by a composite function of hard tanh and sign, thus we rewrite the Eq. 10 as following:

$$a_b = \alpha_a \times \text{Sign}(\text{Htanh}(\frac{a - \beta_a}{\alpha_a})) + \beta_a. \tag{12}$$

For simplicity, we denote $g(x) = \text{Sign}(\text{Htanh}(x))$, then we can get the gradient of α_a and β_a as Eq. 13 in the backward:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \alpha_a} &= \frac{\partial \mathcal{L}}{\partial a_b}\frac{\partial a_b}{\partial \alpha_a} = \frac{\partial \mathcal{L}}{\partial a_b}(g(\frac{a - \beta_a}{\alpha_a}) - \frac{a}{\alpha_a}g'(\frac{a - \beta_a}{\alpha_a})), \\ \frac{\partial \mathcal{L}}{\partial \beta_a} &= \frac{\partial \mathcal{L}}{\partial a_b}\frac{\partial a_b}{\partial \beta_a} = \frac{\partial \mathcal{L}}{\partial a_b}(1 - g'(\frac{a - \beta_a}{\alpha_a})), \end{aligned} \tag{13}$$
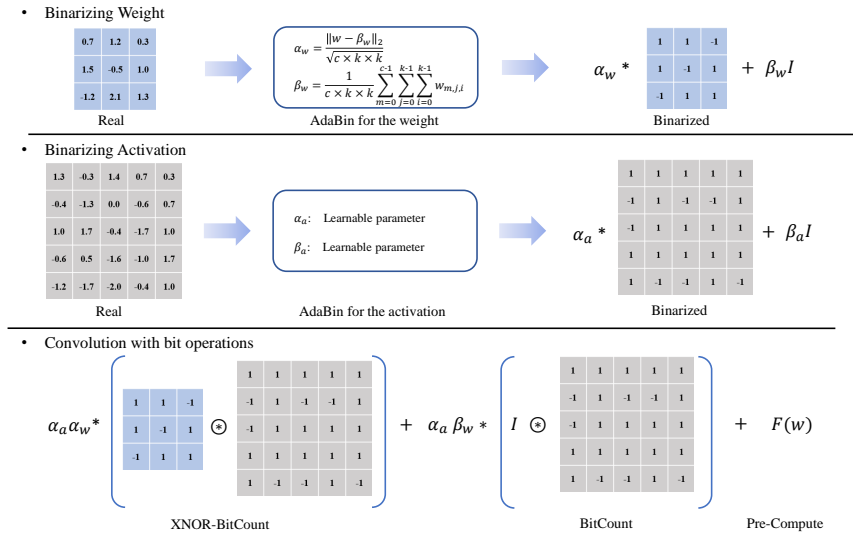
Fig. 3: Binary convolution process. The I represents the identity matrix, and $F(w)$ represents the extra computation with w, which could be pre-computed during the inference.

where $g'(x)$ is the derivative of $g(x)$. We set the initial values of center position β_a and distance α_a to 0 and 1, so that the initial effect of our binary quantizer is equivalent to the sign function[13,34,36]. Then these two parameters of different layers are dynamically updated via gradient descent-based training, and converge to the optimal center and distance values, which is much different from the unified usage of the sign function in the previous BNNs. During inference, the α_a and β_a of all the layers are fixed, then we can binarize the floating point activations into 1-bit as followings:

$$a_b = \alpha_a b_a + \beta_a, b_a \in \{-1, +1\}, \tag{14}$$

where the b_a is the 1-bit storage form and obtained online with input data. Compared with the sign function of previous BNNs, AdaBin will take a little overhead but could significantly improve the feature capacity of activations with the adaptive binary sets for each layer.

3.3   Non-linearity

Prior methods [35] propose to use Parametric Rectified Linear Unit (PReLU) [20] as it is known to facilitate the training of binary networks. PReLU adds an adaptively learnable scaling factor in the negative part and remain unchanged in the positive part. However, we empirically found that the binary values with

Networks	Methods	W/A	Acc. (%)
ResNet-18	Full-precision	32/32	94.8
	RAD [15]		90.5
	IR-Net [36]		91.5
	RBNN [31]	1/1	92.2
	ReCU [42]		92.8
	AdaBin (Ours)		**93.1**
ResNet-20	Full-precision	32/32	91.7
	DoReFa [46]		79.3
	DSQ [16]		84.1
	IR-Net [36]	1/1	86.5
	RBNN [31]		87.8
	AdaBin (Ours)		**88.2**
VGG-Small	Full-precision	32/32	94.1
	LAB [24]		87.7
	XNOR-Net [37]		89.8
	BNN [13]		89.9
	RAD [15]	1/1	90.0
	IR-Net [36]		90.4
	RBNN [31]		91.3
	SLB[43]		92.0
	AdaBin (Ours)		**92.3**

Table 2: Comparisons with state-of-the-art methods on CIFAR-10. W/A denotes the bit width of weights and activations.

our proposed AdaBin are almost all positive in very few layers, which invalidate the non-linearity of PReLU. Therefore, to further enhance the representation of feature maps, we propose to utilize Maxout [17] for the stronger non-linearity in our AdaBin, which is defined as Eq. 15.

$$f_c(x) = \gamma_c^+ \text{ReLU}(x) - \gamma_c^- \text{ReLU}(-x), \tag{15}$$

where x is the input of the Maxout function, γ_c^+ and γ_c^- are the learnable coefficient for the positive part and negative part of the c-th channel, respectively. Following the setting of PReLU, the initialization of γ_c^+ and γ_c^- are 1 and 0.25.

3.4 Binary Convolution for AdaBin

The goal of BNNs is to replace the computationally expensive multiplication and accumulation with XNOR and BitCount operations. Although the binary sets are not limited to $\{-1, +1\}$, our method can still be accelerated with bit-wise operations by simple linear transformation. As shown in Fig. 3, we can binarize the weights and get the 1-bit matrix b_w offline via Eq. 9, and binarize the activations to get the 1-bit activations b_a online via Eq. 14, then decompose

the binary convolution into three items. The first term is the same as the previous BNNs, and the second term only needs to accumulation for one output channel, which can be replaced by BitCount. The third term $F(w)$ could be pre-computed in the inference process. For $n = c = 256$, $k = 3$, $w' = h' = 14$, compared with the binary convolution of IR-Net [37], our method only increases 2.74% operations and 1.37% parameters, which are negligible compared to the total complexity and could achieve 60.85× acceleration and 31× memory saving in theory, the detailed analysis is shown in the supplementary material.

4   Experiments

In this section, we demonstrate the effectiveness of our proposed AdaBin via comparisons with state-of-the-art methods and extensive ablation experiments.

4.1   Results on CIFAR-10

We train AdaBin for 400 epochs with a batch size of 256, where the initial learning rate is set to 0.1 and then decay with CosineAnnealing as IR-Net[36]. We adopt SGD optimizer with a momentum of 0.9, and use the same data augmentation and pre-processing in [21] for training and testing. We compare AdaBin with BNN [13], LAB [24], XNOR-Net [37], DoReFa [46], DSQ [16], RAD [15], IR-Net [36], RBNN [31], ReCU [42] and SLB[43]. Table 2 shows the performance of these methods on CIFAR-10. AdaBin obtains 93.1% accuracy for ResNet-18 architecture, which outperforms the ReCU by 0.3% and reduces the accuracy gap between BNNs and floating-point model to 1.7%. Besides, AdaBin obtains 0.4% accuracy improvement on ResNet-20 compared to the current best method RBNN, and gets 92.3% accuracy while binarizing the weights and activations of VGG-small into 1-bits, which outperforms SLB by 0.3%.

4.2   Results on ImageNet

We train our proposed AdaBin for 120 epochs from scratch and use SGD optimizer with a momentum of 0.9. We set the initial learning rate to 0.1 and then decay with CosineAnnealing following IR-Net[36], and utilize the same data augmentation and pre-processing in [21]. In order to demonstrate the generality of our method, we conduct experiments on two kinds of structures. The first group is the common architectures that are widely used in various computer vision tasks, such as AlexNet [27] and ResNet [21]. Another kind is the binary-specific structures such as BDenseNet [7], MeliusNet [5] and ReActNet [33], which are designed for BNNs and could significantly improve the accuracy with the same amount of parameters as common structures.

Networks	Methods	W/A	Top-1 (%)	Top-5 (%)
AlexNet	Full-precision	32/32	56.6	80.0
	BNN [13]		27.9	50.4
	DoReFa [46]		43.6	-
	XNOR [37]	1/1	44.2	69.2
	SiBNN [38]		50.5	74.6
	AdaBin (Ours)		**53.9**	**77.6**
ResNet-18	Full-precision	32/32	69.6	89.2
	BNN [13]		42.2	-
	XNOR-Net [37]		51.2	73.2
	Bi-Real [34]		56.4	79.5
	IR-Net [36]	1/1	58.1	80.0
	Si-BNN [38]		59.7	81.8
	RBNN [31]		59.9	81.9
	SiMaN [30]		60.1	82.3
	ReCU[42]		61.0	82.6
	AdaBin (Ours)		**63.1**	**84.3**
	IR-Net* [36]		61.8	83.4
	Real2Bin [35]	1/1	65.4	86.2
	ReActNet* [33]		65.5	86.1
	AdaBin* (Ours)		**66.4**	**86.5**
ResNet-34	Full-precision	32/32	73.3	91.3
	ABC-Net [32]		52.4	76.5
	Bi-Real [34]		62.2	83.9
	IR-Net [36]		62.9	84.1
	SiBNN [38]	1/1	63.3	84.4
	RBNN [31]		63.1	84.4
	ReCU[42]		65.1	85.8
	AdaBin (Ours)		**66.4**	**86.6**

Table 3: Comparison with state-of-the-art methods on ImageNet for AlexNet and ResNets. W/A denotes the bit width of weights and activations. * means using the two-step training setting as ReActNet.

Common structures. We show the ImageNet performance of AlexNet, ResNet-18 and ResNet-34 on Table 3, and compare AdaBin with recent methods like Bi-Real [34], IR-Net [36], SiBNN [38], RBNN [31], ReCU[42], Real2Bin [35] and ReActNet [33]. For AlexNet, AdaBin could greatly improve its performance on ImageNet, outperforming the current best method SiBNN by 3.4%, and reducing the accuracy gap between BNNs and floating-point model to only 2.7%. Besides, AdaBin obtains a 63.1% Top-1 accuracy with ResNet-18 structure, which only replaces the binary function and non-linear module of IR-Net [36] with the adaptive quantizer and Maxout but gets 5.0% improvement and outperforms the current best method ReCU by 2.1%. For ResNet-34, AdaBin obtain 1.3% performance improvement compared to the ReCU while binarizing the weights and activations into 1-bits. Besides, we also conduct experiments on ResNet-18 following the training setting as ReActNet. With the two step training strat-

Networks	Methods	OPs ($\times 10^8$)	Top-1 (%)
BDenseNet28 [7]	Origin	2.09	62.6
	AdaBin	2.11	**63.7** (+1.1)
MeliusNet22 [5]	Origin	2.08	63.6
	AdaBin	2.10	**64.6** (+1.0)
MeliusNet29 [5]	Origin	2.14	65.8
	AdaBin	2.17	**66.5** (+0.7)
MeliusNet42 [5]	Origin	3.25	69.2
	AdaBin	3.28	**69.7** (+0.5)
MeliusNet59 [5]	Origin	5.25	71.0
	AdaBin	5.27	**71.6** (+0.6)
ReActNet-A [33]	Origin	0.87	69.4
	AdaBin	0.88	**70.4** (+1.0)

Table 4: Comparisons on ImageNet for binary-specific structures.

egy, AdaBin could get 66.4% top-1 accuracy, which obtains 0.9% improvement compared to ReActNet.

Binary-specific structures. Table 4 shows the performance comparison with BDenseNet, MeliusNet and ReActNet. For BDenseNet28, AdaBin could get 1.1% improvement with the same training setting, which costs negligible extra computational operations. Similarly, when AdaBin is applied to MeliusNet, an advanced version of BDenseNet, it outperforms the original networks by 1.0%, 0.7%, 0.5% and 0.6%, respectively, demonstrating that AdaBin could significantly improve the capacity and quality of binary networks. Besides, we also train the ReActNet-A structure with our AdaBin, following the same training setting with ReActNet [33]. As we can see that, AdaBin could get 1.0% performance improvement with the similar computational operations. Our method could explicitly improve the accuracy of BNNs with a little overhead compared to state-of-the-art methods, as shown in Fig. 1a.

4.3   Results on PASCAL VOC

Table 5 presents the results of object detection on PASCAL VOC dataset for different binary methods. We follow the training strategy as BiDet [34]. The backbone network was pre-trained on ImageNet [14] and then we finetune the whole network for the object detection task. During training, we used the data augmentation techniques in [40], and the Adam optimizer [26] was applied. The learning rate started from 0.001 and decayed twice by multiplying 0.1 at the 160-th and 180-th epoch out of 200 epochs. Following the setting of BiDet [40], we evaluate our proposed AdaBin on both the normal structure and the structure with real-valued shortcut. We compare them with general binary methods BNN [13], XNOR-Net [37] and BiReal-Net [34], and also compare with BiDet [40] and AutoBiDet [39], which are specifically designed for high-performance binary

Methods	W/A	#Params(M)	FLOPs(M)	mAP
Full-precision	32/32	100.28	31750	72.4
TWN [29]	2/32	24.54	8531	67.8
DoReFa [46]	4/4	29.58	4661	69.2
BNN [13]	1/1	22.06	1275	42.0
XNOR-Net [37]	1/1	22.16	1279	50.2
BiDet [40]	1/1	22.06	1275	52.4
AutoBiDet [39]	1/1	22.06	1275	53.5
AdaBin (Ours)	1/1	22.47	1280	**64.0**
BiReal-Net [34]	1/1	21.88	1277	63.8
BiDet* [40]	1/1	21.88	1277	66.0
AutoBiDet* [39]	1/1	21.88	1277	67.5
AdaBin* (Ours)	1/1	22.47	1282	**69.4**

Table 5: The comparison of different methods on PASCAL VOC for object detection. W/A denotes the bit width of weights and activations. * means the the proposed method with extra shortcut for the architectures [40].

detectors. And for reference, we also show the results of the multi-bit quantization method TWN [29] and DoReFa [46] with 4 bit weights and activations. Compared with the previous general BNNs, the proposed AdaBin improves the BNN by 22.0 mAP, XNOR by 13.8 mAP and Bi-Real Net by 5.6 mAP. Even for the task-specific optimization method BiDet, they are 11.6 mAP and 2.6 mAP lower than our method with two structures, and the improved AutoBiDet still lower than AdaBin by 10.5 mAP and 1.9 mAP. Besides, AdaBin with shortcut structure could outperform TWN and DoReFa, which demonstrates that our could significantly enable the binary neural networks to complex tasks.

4.4   Ablation Studies

Effect of AdaBin quantizer.   We conduct the experiments by starting with a vanilla binary neural networks, and then add the AdaBin quantizer of weights and activations gradually. To evaluate different methods fairly, we utilize PReLU for these experiments, which is equal to the Maxout function only with γ^- for negative part. The results are shown in Table 6a, we can see that when combined with existing activation binarization by sign function, our equalization method for binarizing weights could get 0.6% accuracy improvement. Besides, when we free the α_w and β_w to two learnable parameters which are trained in an end-to-end manner as activation, it only get 86.7% accuracy and is much poorer than AdaBin (the last two row). We find that its Kullback-Leibler divergence is also less than AdaBin, which shows the KLD is much important to 1-bit quantization. When keeping the weight binarization as XNOR-Net [37], the proposed gradient-based optimization for binarizing activations could get 1.6% accuracy improvement, as shown in the 3rd row. Combining the proposed weight equalization and activation optimization of AdaBin boosts the accuracy by 2% over

W_set	A_set	Non-linearity	Acc.(%)
$\{-\alpha, +\alpha\}$	$\{-1, +1\}$	PReLU	85.7
$\{w_{b1}, w_{b2}\}$	$\{-1, +1\}$	PReLU	86.3
$\{-\alpha, +\alpha\}$	$\{a_{b1}, a_{b2}\}$	PReLU	87.3
$\{w_{b1}, w_{b2}\}$	$\{a_{b1}, a_{b2}\}$	PReLU	87.7
$\{w_{b1}, w_{b2}\}^{*}$	$\{a_{b1}, a_{b2}\}$	Maxout	86.7
$\{w_{b1}, w_{b2}\}$	$\{a_{b1}, a_{b2}\}$	Maxout	**88.2**

(a) Binary quantizer

Scale factors	Top-1 (%)	Top-5 (%)
None	53.2	77.2
γ^{+}	62.8	83.9
γ^{-}	62.9	84.1
γ^{-}, γ^{+}	**63.1**	**84.3**

(b) γ in Maxout

Table 6: (a) Ablation studies of AdaBin for ResNet-20 on CIFAR-10. * means the α_w and β_w are learnable parameters to the binary sets. (b) The ablation studies of Maxout on ImageNet, the scale factor with γ^{-} equals to PReLU.

vanilla BNN (the 1st vs. 4th row), which shows that AdaBin quantizer could significantly improve the capacity of BNNs.

Effect of γ in Maxout.   In addition, we evaluate four activation functions on ImageNet. The first is none, denoting it is an identity connection. The second is Maxout that only with γ^{+} for positive part, the third is Maxout only with γ^{-} for negative part and the last one is the complete Maxout as Eq. 15. As shown in Table 6b, the coefficient of γ^{+} and γ^{-} improve the accuracy by 9.6% and 9.7% individually. The activation function with both coefficients gets the best performance, which justifies the effectiveness of Maxout.

5   Conclusion

In this paper, we propose an adaptive binary method (AdaBin) to binarize weights and activations with optimal value sets, which is the first attempt to relax the constraints of the fixed binary set in prior methods. The proposed AdaBin could make the binary weights best match the real-valued weights and obtain more informative binary activations to enhance the capacity of binary networks. We demonstrate that our method could also be accelerated by XNOR and BitCount operations, achieving 60.85$\times$ acceleration and 31$\times$ memory saving in theory. Extensive experiments on CIFAR-10 and ImageNet show the superiority of our proposed AdaBin, which outperforms state-of-the-art methods on various architectures, and significantly reduce the performance gap between binary neural networks and real-valued networks. We also present extensive experiments for object detection, which demonstrates that our method can naturally be extended to more complex vision tasks.

References

1. Ieee standard for binary floating-point arithmetic. ANSI/IEEE Std 754-1985 pp. 1–20 (1985). https://doi.org/10.1109/IEEESTD.1985.82928
2. Anderson, A.G., Berg, C.P.: The high-dimensional geometry of binary neural networks. arXiv preprint arXiv:1705.07199 (2017)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
4. Baskin, C., Liss, N., Schwartz, E., Zheltonozhskii, E., Giryes, R., Bronstein, A.M., Mendelson, A.: Uniq: Uniform noise injection for non-uniform quantization of neural networks. ACM Transactions on Computer Systems (TOCS) **37**(1-4), 1–15 (2021)
5. Bethge, J., Bartz, C., Yang, H., Chen, Y., Meinel, C.: Meliusnet: An improved network architecture for binary neural networks. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1439–1448 (2021)
6. Bethge, J., Yang, H., Bornstein, M., Meinel, C.: Back to simplicity: How to train accurate bnns from scratch? arXiv preprint arXiv:1906.08637 (2019)
7. Bethge, J., Yang, H., Bornstein, M., Meinel, C.: Binarydensenet: developing an architecture for binary neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. pp. 0–0 (2019)
8. Cai, Z., He, X., Sun, J., Vasconcelos, N.: Deep learning with low precision by half-wave gaussian quantization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5918–5926 (2017)
9. Chen, H., Wang, Y., Xu, C., Shi, B., Xu, C., Tian, Q., Xu, C.: Addernet: Do we really need multiplications in deep learning? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1468–1477 (2020)
10. Chen, X., Zhang, Y., Wang, Y.: Mtp: Multi-task pruning for efficient semantic segmentation networks. arXiv preprint arXiv:2007.08386 (2020)
11. Chen, X., Zhang, Y., Wang, Y., Shu, H., Xu, C., Xu, C.: Optical flow distillation: Towards efficient and stable video style transfer. In: European Conference on Computer Vision. pp. 614–630. Springer (2020)
12. Choukroun, Y., Kravchik, E., Yang, F., Kisilev, P.: Low-bit quantization of neural networks for efficient inference. In: ICCV Workshops. pp. 3009–3018 (2019)
13. Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830 (2016)
14. Deng, J., Dong, W., Socher, R., Li, L., Kai Li, Li Fei-Fei: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. pp. 248–255 (2009). https://doi.org/10.1109/CVPR.2009.5206848
15. Ding, R., Chin, T.W., Liu, Z., Marculescu, D.: Regularizing activation distribution for training binarized deep networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11408–11417 (2019)
16. 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: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4852–4861 (2019)
17. Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout networks. In: International conference on machine learning. pp. 1319–1327. PMLR (2013)

18. Han, K., Wang, Y., Xu, Y., Xu, C., Wu, E., Xu, C.: Training binary neural networks through learning with noisy supervision. In: International Conference on Machine Learning. pp. 4017–4026. PMLR (2020)
19. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural networks. arXiv preprint arXiv:1506.02626 (2015)
20. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2015)
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
22. Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine **29**(6), 82–97 (2012)
23. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
24. Hou, L., Yao, Q., Kwok, J.T.: Loss-aware binarization of deep networks. arXiv preprint arXiv:1611.01600 (2016)
25. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
26. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25**, 1097–1105 (2012)
28. Kullback, S., Leibler, R.A.: On information and sufficiency. The annals of mathematical statistics **22**(1), 79–86 (1951)
29. Li, F., Zhang, B., Liu, B.: Ternary weight networks. arXiv preprint arXiv:1605.04711 (2016)
30. Lin, M., Ji, R., Xu, Z., Zhang, B., Chao, F., Xu, M., Lin, C.W., Shao, L.: Siman: Sign-to-magnitude network binarization. arXiv preprint arXiv:2102.07981 (2021)
31. Lin, M., Ji, R., Xu, Z., Zhang, B., Wang, Y., Wu, Y., Huang, F., Lin, C.W.: Rotated binary neural network. Advances in Neural Information Processing Systems **33** (2020)
32. Lin, X., Zhao, C., Pan, W.: Towards accurate binary convolutional neural network. arXiv preprint arXiv:1711.11294 (2017)
33. Liu, Z., Shen, Z., Savvides, M., Cheng, K.T.: Reactnet: Towards precise binary neural network with generalized activation functions. In: European Conference on Computer Vision. pp. 143–159. Springer (2020)
34. Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., Cheng, K.T.: Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In: Proceedings of the European conference on computer vision (ECCV). pp. 722–737 (2018)
35. Martinez, B., Yang, J., Bulat, A., Tzimiropoulos, G.: Training binary neural networks with real-to-binary convolutions. arXiv preprint arXiv:2003.11535 (2020)
36. 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2250–2259 (2020)

37. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: European conference on computer vision. pp. 525–542. Springer (2016)

38. Wang, P., He, X., Li, G., Zhao, T., Cheng, J.: Sparsity-inducing binarized neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 12192–12199 (2020)

39. Wang, Z., Lu, J., Wu, Z., Zhou, J.: Learning efficient binarized object detectors with information compression. IEEE Transactions on Pattern Analysis & Machine Intelligence (01), 1–1 (jan 5555). https://doi.org/10.1109/TPAMI.2021.3050464

40. Wang, Z., Wu, Z., Lu, J., Zhou, J.: Bidet: An efficient binarized object detector. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2049–2058 (2020)

41. Xu, Y., Han, K., Xu, C., Tang, Y., Xu, C., Wang, Y.: Learning frequency domain approximation for binary neural networks. In: NeurIPS (2021)

42. Xu, Z., Lin, M., Liu, J., Chen, J., Shao, L., Gao, Y., Tian, Y., Ji, R.: Recu: Reviving the dead weights in binary neural networks. arXiv preprint arXiv:2103.12369 (2021)

43. Yang, Z., Wang, Y., Han, K., Xu, C., Xu, C., Tao, D., Xu, C.: Searching for low-bit weights in quantized neural networks. arXiv preprint arXiv:2009.08695 (2020)

44. Yu, X., Liu, T., Wang, X., Tao, D.: On compressing deep models by low rank and sparse decomposition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7370–7379 (2017)

45. Zhang, Z., Shao, W., Gu, J., Wang, X., Luo, P.: Differentiable dynamic quantization with mixed precision and adaptive resolution. In: International Conference on Machine Learning. pp. 12546–12556. PMLR (2021)

46. 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 preprint arXiv:1606.06160 (2016)