# SA-BNN: State-Aware Binary Neural Network

**Chunlei Liu[1,2], Peng Chen[2], Bohan Zhuang[3], Chunhua Shen[2], Baochang Zhang[1], Wenrui Ding[1]**

[1] Beihang University
[2] The University of Adelaide
[3] Monash University
liuchunlei@buaa.edu.cn, blueardour@gmail.com, bohan.zhuang@monash.edu,
chunhua.shen@adelaide.edu.au, bczhang@buaa.edu.cn, ding@buaa.edu.cn

**Abstract**

Binary Neural Networks (BNNs) have received significant attention due to the memory and computation efficiency recently. However, the considerable accuracy gap between BNNs and their full-precision counterparts hinders BNNs to be deployed to resource-constrained platforms. One of the main reasons for the performance gap can be attributed to the frequent weight flip, which is caused by the misleading weight update in BNNs. To address this issue, we propose a state-aware binary neural network (SA-BNN) equipped with the well designed state-aware gradient. Our SA-BNN is inspired by the observation that the frequent weight flip is more likely to occur, when the gradient magnitude for all quantization states $\{-1, 1\}$ is identical. Accordingly, we propose to employ independent gradient coefficients for different states when updating the weights. Furthermore, we also analyze the effectiveness of the state-aware gradient on suppressing the frequent weight flip problem. Experiments on ImageNet show that the proposed SA-BNN outperforms the current state-of-the-arts (*e.g.,* Bi-Real Net) by more than 3% when using a ResNet architecture. Specifically, we achieve 61.7%, 65.5% and 68.7% Top-1 accuracy with ResNet-18, ResNet-34 and ResNet-50 on ImageNet, respectively.

## Introduction

Binary Neural Networks (BNNs) (Courbariaux et al. 2016; Zhang et al. 2019) have become one of the most prevailing approaches in deep learning model compression (Choi et al. 2018; Mishra et al. 2017; Gong et al. 2019; Cai et al. 2017) due to their computation efficiency. As demonstrated in (Rastegari et al. 2016), BNNs can potentially achieve $32\times$ memory compression ratio, and up to $58\times$ speed-up on CPU compared with their full-precision counterparts. However, the large accuracy drop of BNNs hampers them to be deployed in real applications (Tang, Hua, and Wang 2017; Bulat et al. 2019; Sakr et al. 2018; Han et al. 2019).

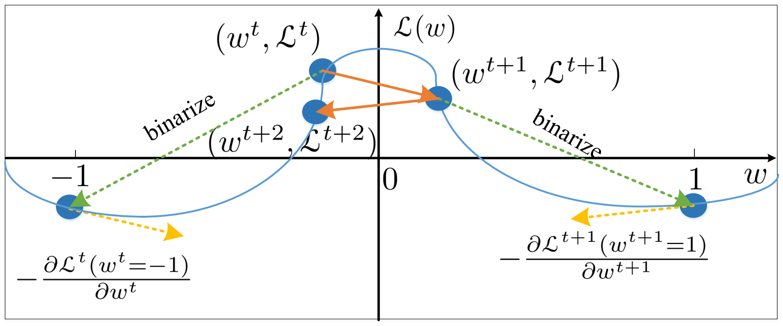
The challenge of the accuracy drop instinctively comes from the extremely limited binarization states $\{-1, 1\}$, which would cause considerable propagation errors in forward and backward procedures, and further lead to misleading weight update (Bai, Wang, and Liberty 2018). To further illustrate this issue, two sequential training iterations are shown in Figure 1. Specifically, with the $\mathrm{sign}(\cdot)$ quantizer, BNNs compute the gradient at the binarized value while performing the update at the full-precision value (Rastegari et al. 2016; Liu et al. 2018), which might mislead the training. Despite the chance to be guided into the correct direction during training, the weight update is more likely to be continuously misled and cause a frequent weight flip, leading to less efficient training and a sub-optimal model.

With respect with the aforementioned problem, we aim to make the training more efficient by suppressing the fluctuation of the weight update. Specifically, we find that existing methods (Rastegari et al. 2016; Darabi et al. 2018; Liu et al. 2018) possess the identical gradient magnitude for all quantization states $\{-1, 1\}$. According to our analysis, the frequent weight flip in this case is more likely to occur. The intuition here is about "*Can we calibrate the magnitude of the two states and make them distinctive such that they would have a different chance of weight flip?* Thus, it would also make weight flip be less frequent to occur. Inspired by this, a novel state-aware binary neural network (SA-BNN) equipped with the carefully-designed state-aware gradient is proposed in this paper. Specifically, we set separate learnable gradient coefficients for different states. Thus, the unnecessary weight update can be impeded efficiently. Moreover, we also provide a formal analysis to further justify the claim. We conduct comprehensive experiments and the results also validate that our SA-BNN provides a more reliable and effective gradient calculation than other works. Our main contributions are summarized as follows.

- We propose a state-aware binary neural network (SA-BNN) equipped with the state-aware gradient, which can provide more reliable and effective gradient calculation than prior work.
- We analyze the effectiveness of the state-aware gradient on suppressing the frequent weight flip problem and alleviate the ineffective update issue in BNNs optimization.
- We conduct extensive evaluations on ImageNet, showing that our SA-BNN achieves state-of-the-art results. In particular, the proposed SA-BNN outperforms Bi-Real Net by 5.3%, 3.3% and 6.1% in Top-1 accuracy with ResNet-18, ResNet-34 and ResNet-50, respectively.

## Related Work

There have been extensive studies on network compression and acceleration (Zhang et al. 2018a; Zhou et al. 2017, 2016;



$(w^t, \mathcal{L}^t)$
$\mathcal{L}(w)$
$(w^{t+1}, \mathcal{L}^{t+1})$
binarize
$(w^{t+2}, \mathcal{L}^{t+2})$
binarize
$-1$
$0$
$1$
$w$
$-\frac{\partial \mathcal{L}^t(w^t=-1)}{\partial w^t}$
$-\frac{\partial \mathcal{L}^{t+1}(w^{t+1}=1)}{\partial w^{t+1}}$

Figure 1: Illustration of the frequent weight flip issue caused by the misleading weight update, where $w$ and $\mathcal{L}$ indicate the full-precision weight and network loss function (blue line), respectively. Two sequential training iterations are presented. In BNNs, the full-precision tensor is binarized by $\mathrm{sign}$ function (described by the green dotted line) in the forward propagation. In the backward propagation, the gradient is computed based on the quantized value (indicated by the yellow dotted line). However, the update is conducted on the full-precision value (the orange line), which might mislead the training. From iteration $t$ to $t+1$, a misleading weight update occurs, causing a flip from $-1$ to $1$, and a similar misleading weight update that occurs from iteration $t+1$ to $t+2$ causes a flip from $1$ to $-1$. With extremely limited representation states, the weight flip happens more frequently in BNNs.

Yang et al. 2020; Shen et al. 2019), including quantization (Faraone et al. 2018; Cai et al. 2017; Jung et al. 2019; Zhuang et al. 2018, 2019), pruning (He, Zhang, and Sun 2017), and compact network design (Zhang et al. 2018b). We focus on the binarization in this work. In general, most binarization algorithms (Bethge et al. 2020; Martinez et al. 2020a) aim at tackling three main problems.

The first category targets at designing accurate binarization approximation methods (Gu et al. 2019a,b; Rastegari et al. 2016; Bulat and Tzimiropoulos 2019; Li et al. 2017). In BNNs, both weights and activations are constrained to $\{-1, 1\}$ using sign function. XNOR-Net (Rastegari et al. 2016) uses $\{-\alpha, \alpha\}$ instead of $\{-1, 1\}$, where $\alpha$ is a channel-wise full-precision scales. Since sign function is non-differentiable, HardTanh (Courbariaux et al. 2016), clip (Rastegari et al. 2016), swishsign (Darabi et al. 2018) and piecewise polynomial function (Liu et al. 2018) are used as differentiable approximation of the sign function for better backward propagation. Besides, Real-to-Binary Net (Martinez et al. 2020a) proposes to use a data-driven method to calculate the scaling factor for the activations after 1-bit convolution, through which the performance of BNNs can be large improved.

The second category aims at improving the capacity of the model. Bi-Real Net (Liu et al. 2018) proposes to increase the number of shortcut connections in ResNet. CBCN (Liu et al. 2019) designs a simple and unique variation process to increase the filter diversity in BNNs. Moreover, GroupNet (Zhuang et al. 2019) is proposed to approximate the full-precision convolution by aggregating several binary branches.

The third category focuses on exploring the efficient training methods (Chen et al. 2017; Mishra and Marr 2017; Liu et al. 2020b; Hou, Yao, and Kwok 2016). BinaryDuo (Kim et al. 2020) proposes a new training scheme for binary activation networks in which two binary activations are coupled into a ternary activation during training. Martinez et al. (Martinez et al. 2020b) devise a sequence of teacher-student pairs to progressively bridge the architecture gap between real and binary networks.

Most previous works mainly focus on designing advanced binarization methods. The issue of frequent weight flip is less studied explicitly. To solve this problem, Helwegen et al. (Helwegen et al. 2019) introduce an optimizer specifically designed for BNNs by directly updating the state of binarized weights. ProxQuant (Bai, Wang, and Liberty 2018) formulates the quantized network training as a regularized learning problem instead and optimizes it via the prox-gradient method. These methods consider the weight updating consistently for different states, which may cause the frequent forth-and-back weight flip issue. To improve the BNNs optimization, we propose SA-BNN in this paper, which learns independent gradient coefficients for each state for more efficient training.

## Method

### Preliminary and Motivation

BNNs refer to the CNN models with binary weights and binary activations, specifically through a $\mathrm{sign}(\cdot)$ function. For a convolutional layer, the forward pass can be written as

$$x^{l+1} = (\hat{x}^l * \hat{w}^l)\alpha = (\mathrm{sign}(x^l) * \mathrm{sign}(w^l))\alpha, \qquad (1)$$

where $x^l$ and $w^l$ indicate the elements of full-precision activation $X^l$ and weight $W^l$ in the $l$-th layer, respectively, $\hat{x}^l, \hat{w}^l \in \{-1, 1\}$ represent the corresponding binarized values, $\alpha$ is a full-precision scaling factor, and $*$ represents the convolution operator. Correspondingly, the backward pass can be written as

$$\frac{\partial \mathcal{L}}{\partial x^l} = \frac{\partial \mathcal{L}}{\partial \hat{x}^l}\frac{\partial \hat{x}^l}{\partial x^l}, \qquad \frac{\partial \mathcal{L}}{\partial w^l} = \frac{\partial \mathcal{L}}{\partial \hat{w}^l}\frac{\partial \hat{w}^l}{\partial w^l}, \qquad (2)$$

where $\mathcal{L}$ is the network loss. Specifically, the triangle-shaped derivative (Liu et al. 2018) and square-shaped derivative

(Rastegari et al. 2016) are widely used options for $\frac{\partial \hat{x}^l}{\partial x^l}$ and $\frac{\partial \hat{w}^l}{\partial w^l}$ to approximate the gradients. Notably, these methods treat the gradients of the two states in BNNs equally and do not distinctively maintain the weight updating. However, our analysis shows that this weight updating manner might aggravate the frequent weight flip issue when considerable noise exists in the mini-batch gradients. It motivates us to employ state-aware gradient in BNNs to discourage unnecessary weight updating for efficient training.

## State-Aware Binary Neural Network

As aforementioned, we propose the following state-aware gradient to stabilize the optimization:

$$\frac{\partial \mathcal{L}}{\partial x} = \begin{cases} \frac{\partial \mathcal{L}}{\partial \hat{x}}(\tau_{-1}\frac{\partial \hat{x}}{\partial x}) & \text{if } \hat{x} = -1 \\ \frac{\partial \mathcal{L}}{\partial \hat{x}}(\tau_{1}\frac{\partial \hat{x}}{\partial x}) & \text{otherwise} \end{cases}, \quad (3)$$

where $\tau_{-1}, \tau_1 \in \mathbb{R}$ are learnable coefficients, which are introduced on the activation gradients to distinctively treat the two states in BNNs. We do not apply the distinguishable parameters $\tau = \{\tau_{-1}, \tau_1\}$ on weight gradient ($\frac{\partial \mathcal{L}}{\partial w}$), since weights themselves are learnable in training process. It is equivalent to regard the state-aware coefficients $\tau$ and weight parameters as a whole. Therefore, we do not consider state-aware gradient on weights, and instead focus on that on activation in the following. According to Eq. (3), we actually leverage an extra scale factor on the activation gradients for each binarization state to impose a mild constraint on the weight updating. When the two scale factors are equal ($\tau_{-1} = \tau_1$), it reduces to the traditional weight updating with state-consistent gradients. Otherwise, it is the proposed state-aware gradient based BNNs. Next, we analyze the difference between these two mechanisms.

**Proposition 1** *The state-aware gradients ($|\tau_{-1}| \neq |\tau_1|$) can suppress frequent weight flip effectively compared with the corresponding state-consistent gradients ($|\tau_{-1}| = |\tau_1|$), leading to more stable training.*

Based on the gradient chain rule, the weight updating procedure can be described as

$$\begin{aligned} w^{l,t+1} &= w^{l,t} - \eta\frac{\partial \mathcal{L}}{\partial w^{l,t}} \\ &= w^{l,t} - \eta\frac{\partial \mathcal{L}}{\partial \hat{x}^{l+1,t}}(\tau^{l+1,t}\frac{\partial \hat{x}^{l+1,t}}{\partial x^{l+1,t}})\frac{\partial x^{l+1,t}}{\partial \hat{w}^{l,t}}\frac{\partial \hat{w}^{l,t}}{\partial w^{l,t}} \\ &= w^{l,t} - \eta\frac{\partial \mathcal{L}}{\partial \hat{x}^{l+1,t}}(\tau^{l+1,t}\frac{\partial \hat{x}^{l+1,t}}{\partial x^{l+1,t}})\hat{x}^{l,t}\frac{\partial \hat{w}^{l,t}}{\partial w^{l,t}} \\ &= w^{l,t} - \tau^{l+1,t}b^{l,t}, \end{aligned} \quad (4)$$

where $\eta$ is the learning rate, $t$ represents the $t$-th iteration, and $b^{l,t} = \eta\frac{\partial \mathcal{L}}{\partial \hat{x}^{l+1,t}}\frac{\partial \hat{x}^{l+1,t}}{\partial x^{l+1,t}}\hat{x}^{l,t}\frac{\partial \hat{w}^{l,t}}{\partial w^{l,t}}$. For simplicity, we ignore the layer index superscript $l$ in the following analysis. According to Eq. (4), to enable a weight flip (namely let $\text{sign}(w^{t+1}) \neq \text{sign}(w^t)$), it requires to satisfy the constraints $\text{sign}(\tau^t b^t) = \text{sign}(w^t)$ and $|\tau^t b^t| > |w^t|$, where $|\cdot|$ represents the magnitude of the input. We assume the initial state $\text{sign}(w^t) = -1$, and the process is similar for the initial state $\text{sign}(w^t) = 1$.

1) If $|\tau_{-1}| = |\tau_1|$, the flip probability from the iteration $t$ to $t+1$ is

$$P(\text{sign}(w^t) \neq \text{sign}(w^{t+1})) = N_{|w^t|}/N, \quad (5)$$

where $N_{|w^t|}$ represents the total number of $b^t$ satisfying $\text{sign}(\tau_1^t b^t) = \text{sign}(w^t)$ and $|\tau_1^t b^t| > |w^t|$, and $N$ represents the total number of $b$. Similarly, the flip probability from the iteration $t+1$ to $t+2$ is

$$P(\text{sign}(w^{t+1}) \neq \text{sign}(w^{t+2})) = N_{|w^{t+1}|}/N, \quad (6)$$

where $N_{|w^{t+1}|}$ represents the total number of $b^{t+1}$ satisfying $\text{sign}(\tau_{-1}^{t+1} b^{t+1}) = \text{sign}(w^{t+1})$ and $|\tau_{-1}^{t+1} b^{t+1}| > |w^{t+1}|$. Thus, the sequential flip probability from the iteration $t$ to $t+2$ is

$$\begin{aligned} &P((\text{sign}(w^t) \neq \text{sign}(w^{t+1})) \cap (\text{sign}(w^{t+1}) \neq \text{sign}(w^{t+2}))) \\ &= (N_{|w^t|}N_{|w^{t+1}|})/N^2. \end{aligned} \quad (7)$$

2) If $|\tau_{-1}| < |\tau_1|$, it remains the same flip probability from the iteration $t$ to $t+1$ as Eq. (5). However, when considering the flip probability from iteration $t+1$ to $t+2$, the number of $b^{t+1}$ that satisfying $|\tau_{-1}^{t+1} b^{t+1}| > |w^{t+1}|$ in this case is less than that in the case of $|\tau_{-1}| = |\tau_1|$.

Therefore, the state-aware gradient (*i.e.,* $|\tau_{-1}| < |\tau_1|$) has lower probability of sequential weight flip compared with the conventional state-consistent methods (*i.e.,* $|\tau_{-1}| = |\tau_1|$),

$$P(A^t \cap A^{t+1} || \tau_{-1}| < |\tau_1|) < P(A^{t+1} \cap A^{t+2} || \tau_{-1}| = |\tau_1|), \quad (8)$$

where $A^t$ represents $\text{sign}(w^t) \neq \text{sign}(w^{t+1})$.

3) If $|\tau_1| < |\tau_{-1}|$, the process is similar to 2). The state-aware gradient also has lower probability of sequential weight flip as

$$P(A^t \cap A^{t+1} || \tau_1| < |\tau_{-1}|) < P(A^{t+1} \cap A^{t+2} || \tau_{-1}| = |\tau_1|). \quad (9)$$

Based on the above analysis, we propose an efficient yet simple solution to realize the state-aware gradient:

$$x^{l+1} = \begin{cases} (\text{sign}(\tau_{-1}^l x^l) * \text{sign}(w^l))\alpha & \text{if } \hat{x} = -1 \\ (\text{sign}(\tau_1^l x^l) * \text{sign}(w^l))\alpha & \text{otherwise} \end{cases}. \quad (10)$$

Compared to Eq. (1), we simply multiply the scale $\tau$ on the activation based on its state. Note that the learnable coefficients $\tau$ are per-channel granularity in our paper. In this way, our SA-BNN is established in exchange for a small increase in computational complexity (only an extra point-wise product between $\tau$ and $x$).

## Discussion

In this section, we first discuss the difference of SA-BNN with related methods in (Helwegen et al. 2019; Bai, Wang, and Liberty 2018), and then further analyze the effectiveness of the proposed SA-BNN.

In particular, Helwegen et al. (Helwegen et al. 2019) argue that latent weights are not necessary for gradient-based optimization of BNNs, and they directly update the state of binarized weights with:

$$w^t = \begin{cases} -w^{t-1} & \text{if } |g^t| \geq \beta \text{ and } \text{sign}(g^t) = \text{sign}(w^{t-1}) \\ w^{t-1} & \text{otherwise} \end{cases}, \tag{11}$$

where $g^t$ is exponential moving average of gradient ($g^t = (1-\gamma)g^{t-1} + \gamma\frac{\partial\mathcal{L}}{\partial w^t}$, where $\gamma$ is the adaptive rate) and $\beta$ is a manually defined threshold to control the weight flipping. From Eq. (11), we can learn that it is easy for the weight to flip when threshold $\beta$ is small and hard to flip when threshold $\beta$ is large. From this perspective, threshold $\beta$ in (Helwegen et al. 2019) is consistent with the coefficients $\tau$ in our method. However, the method in (Helwegen et al. 2019) suppresses the weight flip equally for different states, while SA-BNN treats different binarization states distinctively by employing an independent coefficient for each state. Moreover, different from the handcrafted hyper-parameters $\beta$, the coefficients $\tau$ are learnable, which avoids a very careful tuning during the optimization procedure.

Bai et al. (Bai, Wang, and Liberty 2018) propose the ProxQuant by formulating the quantized network training as a regularized learning problem and optimizing it via the prox-gradient method. Specifically, ProxQuant has access to additional gradient information at non-quantized points, which avoids the misleading weight update in the training. Different from the ProxQuant which suppresses the frequent weight flip by designing a dedicated optimizer, SA-BNN provide a more elegant and simple way which can be easily integrated in existing methods and is able to achieve better performance.

In addition, due to the non-differentiability of sign function in the binarization process, most existing works employ a surrogate for the gradients (Rastegari et al. 2016; Darabi et al. 2018; Liu et al. 2018), in which the gradients are forced to be 0 for values outside $[-1, +1]$. However, once the value falls outside the truncation interval, it cannot be used to update the corresponding weights anymore. This strategy greatly limits the training ability of backward propagation (Qin et al. 2019) when too many values are clipped. In contrast, our SA-BNN has the ability to preserve more gradients through learnable coefficients, thus alleviating the unreliable gradients in BNNs optimization.

## Experiments

### Dataset and Implementation Details

We perform experiments on large-scale dataset ImageNet (ILSVRC12) (Russakovsky et al. 2015), which contains approximately 1.2 million training images and $50K$ validation images from 1000 categories. In our experiments, we employ $224 \times 224$ random crop and center crop for training and inference, respectively. We use ResNet as our backbone, including ResNet-18, ResNet-34 and ResNet-50 (He et al. 2016). We use Adam (Kingma and Ba 2014) with the momentum of 0.9 and set the weight-decay to be 0.

For SA-BNNs with backbone ResNet-18, we run the training algorithm for 95 epochs with a batch size of 256. The learning rate starts from 0.001 and is decayed twice by multiplying 0.1 at the $75^{th}$ and the $85^{th}$ epoch. Besides, for SA-BNNs with backbone ResNet-34, the training process includes 90 epochs and the batch size is set to 256. The learning rate starts from 0.001 and is multiplied by 0.1 at the $60^{th}$ and the $80^{th}$ epoch, respectively. Moreover, for SA-BNNs with backbone ResNet-50, we run the training algorithm for 70 epochs with a batch size of 64. The learning rate starts from 0.0005 and is decayed twice by multiplying 0.1 at the $40^{th}$ and the $60^{th}$ epoch.

### Ablation Study

**Initial Coefficients.** In this section, we investigate the influence of the initialization of $\tau = \{\tau_{-1}, \tau_1\}$ on the final performance. First, $\tau_{-1}$ and $\tau_1$ are constantly set to be 0.4 and 1 for ResNet-18 empirically. We obtain an improvement of 5.8% over the model without the state-aware method (SA) (59.2% *v.s.* 53.4%), which justifies the effectiveness of state-aware gradient. Furthermore, we unleash $\tau$ to be learnable for better performance by setting their initial values to be 0.10, 0.25, 0.40, 0.75, and 1.0. The results are listed in Table 1, where "N" represents $\tau_{-1}$ is learnable while $\tau_1$ is constantly set to be 1 ($\tau_1 = 1$); "P" represents $\tau_1$ is learnable while $\tau_{-1}$ is constantly set to be 1 ($\tau_{-1} = 1$); "S" represents both $\tau_{-1}$ and $\tau_1$ are learnable, and we keep $\tau_{-1} = \tau_1$ in the whole training process; "D" represents both $\tau_{-1}$ and $\tau_1$ are learnable and have the same initial value, but independently updated in the training process. From the results in Table **??**, we observe that the results of $\tau_{-1} \neq \tau_1$ (*i.e.,* "N", "P", "D") outperform those of $\tau_{-1} = \tau_1$ (*i.e.,* "S") by a large margin, which demonstrate that the performance can be improved when the magnitude is different for binary states. Besides, we conduct an additional experiment to validate the performance influence of different initial value settings (0.40 for $\tau_{-1}$, and 1.0 for $\tau_1$) for both learnable $\tau_{-1}$ and $\tau_1$, and observe the accuracy is 60.8%.

We further plot the distributions of learnable coefficients $\tau_{-1}$ on the $14^{th} - 17^{th}$ layer of ResNet-18 with the initial values of 0.4 while fixing $\tau_1 = 1$ in Figure 2. From the results, we observe that most coefficients fall into the range of $[0, 1]$ and the distributions of $\tau_{-1}$ are extremely distinctive in different layers (*e.g.,* the coefficients are distributed around 0 in the $17^{th}$ layer while being around 0.5 in the $16^{th}$ layer), which shows the learnable way can make $\tau$ be adaptive for different layers.

**Training Stability.** To better understand our SA-BNNs, we follow (Helwegen et al. 2019; Bai, Wang, and Liberty 2018) to calculate the flipping state at each epoch and set the ratio of sequential weight flip as

$$\pi^t = \frac{\sum_w A^t \wedge \cdots \wedge A^{t+m-1}}{||\text{sign}(W)||_1}, \tag{12}$$

where $A^t$ represents $\text{sign}(w^t) \neq \text{sign}(w^{t+1})$ and $m$ is the examined epoch interval. In Figure 3, we plot the ratios of sequential weight flip under the case of $m = 2$, $m = 3$, and $m = 4$ for the $10^{th}$ and $14^{th}$ layer of ResNet-18. The results show our methods (the solid lines) have lower flip ratios compared with BNNs without SA (the dotted lines) in both $10^{th}$ and $14^{th}$ layer, which validate the effectiveness

|   | 0.10 | 0.25 | 0.40 | 0.75 | 1.0 |   | 0.10 | 0.25 | 0.40 | 0.75 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| N | 58.8 | 60.0 | 60.6 | 60.6 | 59.2 | P | 59.6 | 59.6 | 60.4 | 60.6 | 59.4 |
| S | 57.5 | 55.7 | 56.4 | 55.8 | 55.6 | D | 58.6 | 59.6 | 60.5 | 60.1 | 59.9 |

Table 1: Top-1 accuracy (%) with different initial values on ResNet-18.

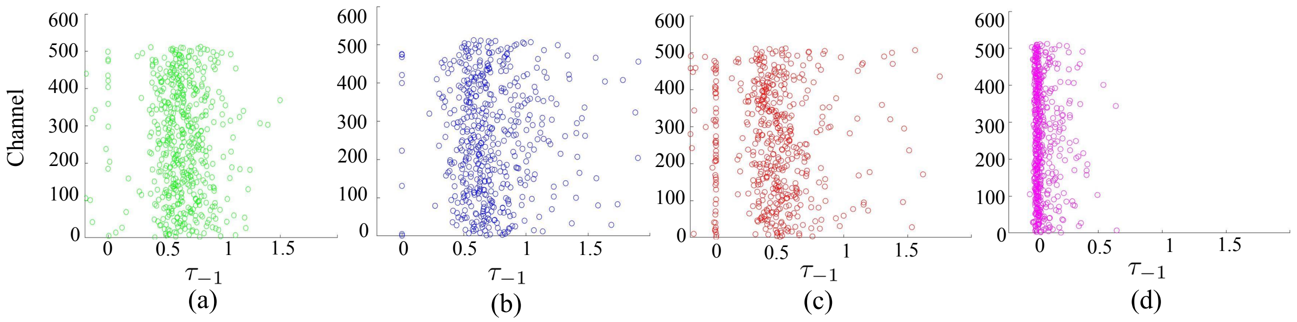


Figure 2: The distributions of learnable $\tau_{-1}$ on ResNet-18 while fixing $\tau_1 = 1$. (a), (b), (c), (d) represent the $14^{th} - 17^{th}$ layer in ResNet-18, respectively.

of our SA-BNNs on suppressing the consecutive weight flip. Besides, we can also find that ratios of sequential weight flip are gradually decreased as the training converges. Note it cannot be updated without flipping at all while frequent flipping will make training unstable, so we aim to adjust weight flip probability to improve training stability and efficiency rather than inhibiting weight flip.

**Effective Gradients in BNNs.** In this paper, we choose the triangle-shaped derivative (Liu et al. 2018) for $\frac{\partial \hat{x}}{\partial x}$. In this case, the gradient is forced to 0 for values outside $[-1, +1]$, where we regard these values as the clipped values. Furthermore, the clipped ratio can be defined as $N_c/N_g$, where $N_c$ represents the number of clipped values and $N_g$ represents the number of all values. Then, we calculate the clipped ratios in Figure 4. Due to the increasing accumulative error in the propagation, clipping too much gradient information is not conducive to obtain better performance. From the results in Figure 4, we observe that the clipped gradient ratios can be reduced drastically (*e.g.,* $0.69$ *v.s.* $0.02$) in the $13^{th}$ layer in ResNet-18 when equipped with SA, which validate that our SA can help BNNs preserve more effective gradient information, thus suppressing the unreliable gradients in BNNs optimization. Note similar conclusions can be obtained with rectangular-shaped derivative.

**Training Techniques.** In this section, we show how our method can alleviate the optimization challenges of BNNs by 1) state-aware method (SA), 2) a BN layer before binarized activation (pbn) (Zhang et al. 2018a) and 3) additional shortcuts (sc) (Liu et al. 2020a). To study how these techniques benefit BNNs individually and collectively, we train BNNs with a combination of these techniques on ImageNet in Table 2. We implement Bi-Real Net as our baseline (base) in this section according to the official source code with the same parameters settings with SA-BNNs.

1) Comparing the columns of "Base" and "Base+SA", SA-BNNs improve the accuracy of baseline by $7.2\%$, $4.6\%$, and $12.7\%$ on ResNet-18, ResNet-34, and ResNet-50, respectively, which validates the effectiveness of our state-aware method.

2) Comparing the columns of "Base+SA" and "Base+SA+pbn", the pbn strategy obtains an improvement of about $1\%$ on three networks. The reason is attributed to that the pbn which act as a feature map normalization can maximize the information entropy of the binarized activation, thus further reducing the binarization error to ensure a higher diversity (Qin et al. 2019).

3) Comparing the columns of "Base+SA+pbn" and "Base+SA+pbn+sc", the networks with additional shortcuts outperform their counterparts with an improvement of up to $0.5\%$. In BNNs, additional shortcuts retain more network representational capability, thus achieving better accuracy.

4) Putting together the above three training strategies, our final models achieve approximately $89\%$, $89\%$, and $92\%$ of the accuracy of their corresponding full-precision networks, but with a huge amount of speedup and computation cost saving.

## Accuracy Comparison with State-of-the-Art

While the ablation study demonstrates the effectiveness of our SA-BNN, it is also necessary to compare with other state-of-the-art methods to further evaluate the overall performance. We carry out a comparative study with six methods: IR-Net (Qin et al. 2019), Bop (Helwegen et al. 2019), CI-Net (Wang et al. 2019), BONN (Gu et al. 2019b), Bi-Real Net (Liu et al. 2018), and XNOR-Net (Rastegari et al. 2016) on ResNet-18, ResNet-34 and ResNet-50 in Table 3. These six works are representative methods of binarizing both weights and activations for CNNs and achieve state-of-the-art results.

The comparison in Table 3 clearly demonstrates that our SA-BNNs outperform other networks by a considerable mar-

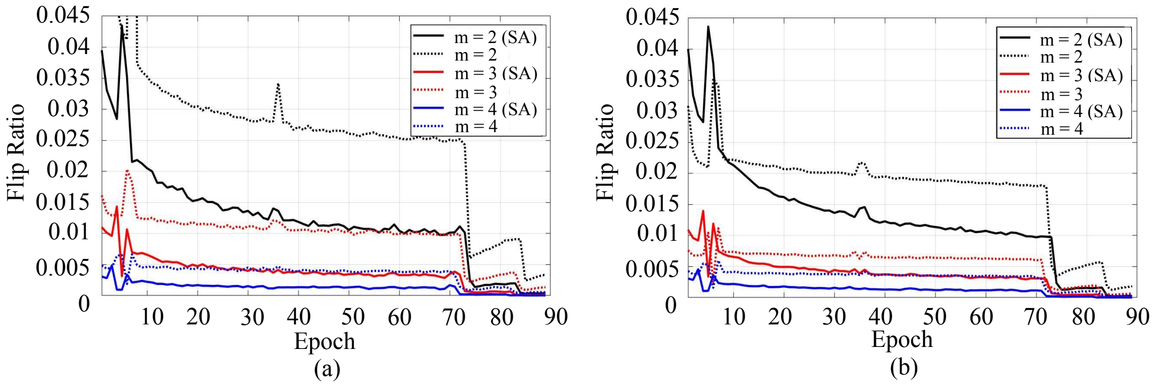


Figure 3: The ratios of sequential weight flip at each epoch with/without SA. (a) and (b) represent the flip ratios at the $10^{th}$ and $14^{th}$ layer with ResNet-18, respectively.

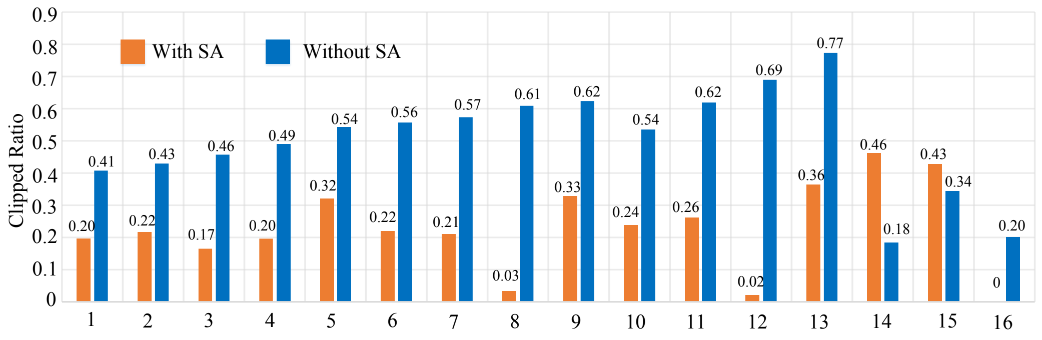


Figure 4: The statistical results of clipped gradient ratios. Orange, blue represent the clipped ratios with and without SA. On the horizontal axis, 1-16 represents 16 intermediate convolutional layers in ResNet-18.

| | Base | | Base+SA | | Base+SA+pbn | | Base+SA+pbn+sc | | Full-precision | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Top-1 | Top-5 | Top-1 | Top-5 | Top-1 | Top-5 | Top-1 | Top-5 | Top-1 | Top-5 |
| ResNet-18 | 53.4 | 76.9 | 60.6 | 82.7 | 61.3 | 82.9 | 61.7 | 82.8 | 69.3 | 89.2 |
| ResNet-34 | 58.5 | 80.6 | 63.1 | 84.3 | 65.0 | 85.3 | 65.5 | 85.8 | 73.3 | 91.3 |
| ResNet-50 | 54.8 | 77.6 | 67.5 | 86.1 | 68.5 | 87.4 | 68.7 | 87.4 | 74.7 | 92.1 |

Table 2: Top-1 and Top-5 accuracy (%) with different combinations of the proposed techniques on three different network backbones.

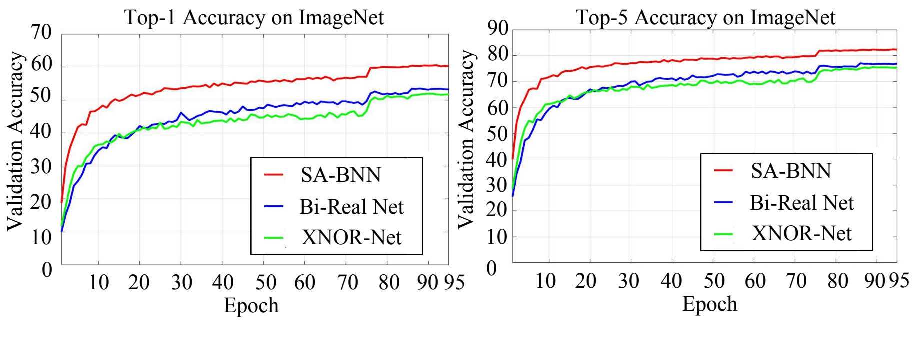


Figure 5: Validation accuracy curves of SA-BNN, Bi-Real Net and XNOR-Net with ResNet-18 backbone on ImageNet.

| | | SA-BNN | IR-Net | Bop | CI-Net | BONN | Bi-Real Net | XNOR-Net | FP |
|---|---|---|---|---|---|---|---|---|---|
| ResNet-18 | Top-1 | 61.7 | 58.1 | 56.6 | 59.9 | 59.3 | 56.4 | 51.2 | 69.3 |
| | Top-5 | 82.8 | 80.0 | 79.4 | 84.2 | 81.6 | 79.5 | 73.2 | 89.2 |
| ResNet-34 | Top-1 | 65.5 | 62.9 | – | 64.9 | – | 62.2 | – | 73.3 |
| | Top-5 | 85.8 | 84.1 | – | 86.6 | – | 83.9 | – | 91.3 |
| ResNet-50 | Top-1 | 68.7 | – | – | – | – | 62.6 | 63.1 | 74.7 |
| | Top-5 | 87.4 | – | – | – | – | 83.9 | 83.6 | 92.1 |

Table 3: Comparison on Top-1 and Top-5 accuracy (%) of SA-BNN with other state-of-the-art binarization methods, including IR-Net (Qin et al. 2019), Bop (Helwegen et al. 2019), CI-Net (Wang et al. 2019), BONN (Gu et al. 2019b), Bi-Real Net (Liu et al. 2018), and XNOR-Net (Rastegari et al. 2016). "FP" means full-precision.

| | | Memory Usage | Memory Saving | FLOPs | Speedup |
|---|---|---|---|---|---|
| ResNet-18 | SA-BNN | 33.7 Mbit | $11.10\times$ | $1.69 \times 10^8$ | $10.71\times$ |
| | Bi-Real Net | 33.6 Mbit | $11.14\times$ | $1.63 \times 10^8$ | $11.06\times$ |
| | XNOR-Net | 33.7 Mbit | $11.10\times$ | $1.67 \times 10^8$ | $10.86\times$ |
| | Full-precision | 374.1 Mbit | – | $1.81 \times 10^9$ | – |
| ResNet-34 | SA-BNN | 44.3 Mbit | $15.74\times$ | $2.01 \times 10^8$ | $18.21\times$ |
| | Bi-Real Net | 43.7 Mbit | $15.97\times$ | $1.93 \times 10^8$ | $18.99\times$ |
| | XNOR-Net | 43.9 Mbit | $15.88\times$ | $1.98 \times 10^8$ | $18.47\times$ |
| | Full-precision | 697.3 Mbit | – | $3.66 \times 10^9$ | – |

Table 4: Memory usage and FLOPs calculation in our method.

gin in terms of the Top-1 accuracy. Note that the results of the other six works are quoted directly from the corresponding references. Specifically, the proposed SA-BNN with backbone ResNet-18 outperforms its counterpart Bi-Real Net by $5.3\%$ and achieves a roughly $2\%$ relative improvement over CI-Net. Similar improvements can be observed for ResNet-34 and ResNet-50 networks. In Figure 5, we plot the validation accuracy curves of XNOR-Net, Bi-Real Net, and SA-BNN (without the contribution of pbn and sc). All networks are implemented under the same hyper-parameter setting. It clearly shows that, by learning distinctive gradient coefficients for binarization states, our method converges faster and better than XNOR-Net and Bi-Real Net. Therefore, SA-BNN is more competitive than other state-of-the-art binary networks.

## Efficiency and Memory Usage Analysis

We further analyze the memory usage saving and speedup in Table 4. Following (Rastegari et al. 2016; Liu et al. 2018), we keep the weights and activations in the first convolutional and the last fully-connected layers to be full-precision. The memory usage $M$ is calculated as

$$M = 32 \times N_f + N_b, \tag{13}$$

where $N_f$ and $N_b$ is the number of full-precision and binary weights, respectively. Besides, the FLOPs is calculated as

$$F = N_{cf} + N_{cb}/64, \tag{14}$$

where $N_{cf}$ and $N_{cb}$ is the number of full-precision and binary multiplication, respectively. For ResNet-18 and ResNet-34, the proposed SA-BNNs reduce the memory usage by $11.10\times$ and $15.74\times$, respectively, in comparison with the corresponding full-precision networks. In addition, our SA-BNNs can be speedup to $10.71\times$ and $18.21\times$, respectively, for backbone ResNet-18 and ResNet-34. Note Bi-Real Net has lower FLOPs compared with XNOR-Net because the scaling factor for binarization in Bi-Real Net is absorbed by the BatchNorm layer at inference time. Compared with XNOR-Net which has not absorbed the scaling factor, we obtain about $10\%$ accuracy improvement on ResNet-18 with small additional FLOPs cost.

## Conclusion

In this paper, we have proposed a state-aware binary neural network (SA-BNN) to improve the performance of BNNs. Different from the standard BNNs, SA-BNN utilizes a simple state-aware gradient to significantly improve network optimization. By employing independent gradient coefficients for different states when updating the weight, SA-BNN effectively suppresses the frequent weight flip problem. Furthermore, we have also analyzed the effectiveness of the state-aware gradient on suppressing frequent weight flip and alleviated the ineffective update in BNNs. Experimental results have demonstrated that the proposed SA-BNN shows substantially superiority over the state-of-the-art methods.

## Acknowledgments

This work was in part supported by Science and Technology Innovation 2030-Key Project of "New Generation Artificial Intelligence" under Grant 2020AAA0108200, National Natural Science Foundation of China (U20B2042), and National Natural Science Foundation of China (62076019). Chunlei Liu, Peng Chen and Bohan Zhuang contribute equally. Wenrui Ding is the corresponding author. This work was done when Chunlei Liu was visiting The University of Adelaide. Chunhua Shen and his employer received no financial support for the research, authorship, and/or publication of this article.

## References

Bai, Y.; Wang, Y.-X.; and Liberty, E. 2018. Proxquant: Quantized neural networks via proximal operators. *arXiv: Comp. Res. Repository* .

Bethge, J.; Bartz, C.; Yang, H.; Chen, Y.; and Meinel, C. 2020. MeliusNet: Can binary neural networks achieve MobileNet-level accuracy? *arXiv: Comp. Res. Repository* .

Bulat, A.; Kossaifi, J.; Tzimiropoulos, G.; and Pantic, M. 2019. Matrix and tensor decompositions for training binary neural networks. *arXiv: Comp. Res. Repository* .

Bulat, A.; and Tzimiropoulos, G. 2019. XNOR-Net++: Improved binary neural networks. *arXiv: Comp. Res. Repository* .

Cai, Z.; He, X.; Sun, J.; and Vasconcelos, N. 2017. Deep learning with low precision by half-wave gaussian quantization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 5918–5926.

Chen, G.; Choi, W.; Yu, X.; Han, T.; and Chandraker, M. 2017. Learning efficient object detection models with knowledge distillation. In *Proc. Advances in Neural Inf. Process. Syst.*, 742–751.

Choi, J.; Wang, Z.; Venkataramani, S.; Chuang, P. I.-J.; Srinivasan, V.; and Gopalakrishnan, K. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv: Comp. Res. Repository* .

Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or-1. *arXiv: Comp. Res. Repository* .

Darabi, S.; Belbahri, M.; Courbariaux, M.; and Nia, V. P. 2018. BNN+: Improved binary network training. *arXiv: Comp. Res. Repository* .

Faraone, J.; Fraser, N.; Blott, M.; and Leong, P. H. 2018. Syq: Learning symmetric quantization for efficient deep neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 4300–4309.

Gong, R.; Liu, X.; Jiang, S.; Li, T.; Hu, P.; Lin, J.; Yu, F.; and Yan, J. 2019. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proc. IEEE Int. Conf. Comp. Vis.*, 4852–4861.

Gu, J.; Li, C.; Zhang, B.; Han, J.; Cao, X.; Liu, J.; and Doermann, D. 2019a. Projection convolutional neural networks for 1-bit cnns via discrete back propagation. In *Proc. AAAI Conf. Artificial Intell.*, volume 33, 8344–8351.

Gu, J.; Zhao, J.; Jiang, X.; Zhang, B.; Liu, J.; Guo, G.; and Ji, R. 2019b. Bayesian Optimized 1-Bit CNNs. In *Proc. IEEE Int. Conf. Comp. Vis.*, 4909–4917.

Han, Y.; Deng, C.; Zhao, B.; and Tao, D. 2019. State-aware anti-drift object tracking. *IEEE Transactions on Image Processing* 28(8): 4075–4086.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 770–778.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *Proc. IEEE Int. Conf. Comp. Vis.*, 1389–1397.

Helwegen, K.; Widdicombe, J.; Geiger, L.; Liu, Z.; Cheng, K.-T.; and Nusselder, R. 2019. Latent weights do not exist: Rethinking binarized neural network optimization. In *Proc. Advances in Neural Inf. Process. Syst.*, 7531–7542.

Hou, L.; Yao, Q.; and Kwok, J. T. 2016. Loss-aware binarization of deep networks. *arXiv: Comp. Res. Repository* .

Jung, S.; Son, C.; Lee, S.; Son, J.; Han, J.-J.; Kwak, Y.; Hwang, S. J.; and Choi, C. 2019. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 4350–4359.

Kim, H.; Kim, K.; Kim, J.; and Kim, J.-J. 2020. BinaryDuo: Reducing Gradient Mismatch in Binary Activation Network by Coupling Binary Activations. *arXiv: Comp. Res. Repository* .

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv: Comp. Res. Repository* .

Li, Z.; Ni, B.; Zhang, W.; Yang, X.; and Gao, W. 2017. Performance guaranteed network acceleration via high-order residual quantization. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2584–2592.

Liu, C.; Ding, W.; Xia, X.; Zhang, B.; Gu, J.; Liu, J.; Ji, R.; and Doermann, D. 2019. Circulant binary convolutional networks: Enhancing the performance of 1-bit dcnns with circulant back propagation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2691–2699.

Liu, Z.; Luo, W.; Wu, B.; Yang, X.; Liu, W.; and Cheng, K.-T. 2020a. Bi-real net: Binarizing deep network towards real-network performance. *Int. J. Comput. Vision* 128(1): 202–219.

Liu, Z.; Shen, Z.; Savvides, M.; and Cheng, K.-T. 2020b. ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions. *arXiv: Comp. Res. Repository* .

Liu, Z.; Wu, B.; Luo, W.; Yang, X.; Liu, W.; and Cheng, K.-T. 2018. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proc. Eur. Conf. Comp. Vis.*, 722–737.

Martinez, B.; Yang, J.; Bulat, A.; and Tzimiropoulos, G. 2020a. Training binary neural networks with real-to-binary convolutions. In *Proc. Int. Conf. Learn. Representations*.

Martinez, B.; Yang, J.; Bulat, A.; and Tzimiropoulos, G. 2020b. Training binary neural networks with real-to-binary convolutions. In *Proc. Int. Conf. Learn. Representations*.

Mishra, A.; and Marr, D. 2017. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *arXiv: Comp. Res. Repository* .

Mishra, A.; Nurvitadhi, E.; Cook, J. J.; and Marr, D. 2017. WRPN: wide reduced-precision networks. *arXiv: Comp. Res. Repository* .

Qin, H.; Gong, R.; Liu, X.; Wei, Z.; Yu, F.; and Song, J. 2019. IR-Net: Forward and backward information retention for highly accurate binary neural networks. *arXiv: Comp. Res. Repository* .

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. Eur. Conf. Comp. Vis.*, 525–542. Springer.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision* 115(3): 211–252.

Sakr, C.; Choi, J.; Wang, Z.; Gopalakrishnan, K.; and Shanbhag, N. 2018. True gradient-based training of deep binary activated neural networks via continuous binarization. In *Proc. IEEE Int. Conf. Acoustics, Speech & Signal Process.*, 2346–2350. IEEE.

Shen, M.; Han, K.; Xu, C.; and Wang, Y. 2019. Searching for accurate binary neural architectures. In *Proc. IEEE Int. Conf. Comp. Vis. Workshops*, 0–0.

Tang, W.; Hua, G.; and Wang, L. 2017. How to train a compact binary neural network with high accuracy? In *Proc. AAAI Conf. Artificial Intell.*

Wang, Z.; Lu, J.; Tao, C.; Zhou, J.; and Tian, Q. 2019. Learning channel-wise interactions for binary convolutional neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 568–577.

Yang, Z.; Wang, Y.; Han, K.; Xu, C.; Xu, C.; Tao, D.; and Xu, C. 2020. Searching for low-bit weights in quantized neural networks. *Proc. Advances in Neural Inf. Process. Syst.* 33.

Zhang, D.; Yang, J.; Ye, D.; and Hua, G. 2018a. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proc. Eur. Conf. Comp. Vis.*, 365–382.

Zhang, J.; Pan, Y.; Yao, T.; Zhao, H.; and Mei, T. 2019. dabnn: A super fast inference framework for binary neural networks on arm devices. In *Proc. ACM Int. Conf. Multimedia*, 2272–2275.

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018b. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 6848–6856.

Zhou, A.; Yao, A.; Guo, Y.; Xu, L.; and Chen, Y. 2017. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv: Comp. Res. Repository* .

Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; and Zou, Y. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv: Comp. Res. Repository* .

Zhuang, B.; Shen, C.; Tan, M.; Liu, L.; and Reid, I. 2018. Towards effective low-bitwidth convolutional neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 7920–7928.

Zhuang, B.; Shen, C.; Tan, M.; Liu, L.; and Reid, I. 2019. Structured binary neural networks for accurate image classification and semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 413–422.