

Pruning as a Binarization Technique

*Original*

Pruning as a Binarization Technique / Frickenstein, L., Mori, P., Balamuthu Sampath, S., Thoma, M., Fasfous, N., Rohit Vemparala, M., Frickenstein, A., Unger, C., Passerone, C., Stechele, W.. - ELETTRONICO. - (2024), pp. 2131-2140. (Conference on Computer Vision and Pattern Recognition (CVPR) Seattle, WA (USA) 17-18 June 2024) [10.1109/CVPRW63382.2024.00218].

*Availability:*

This version is available at: 11583/2989943 since: 2024-06-27T14:00:25Z

*Publisher:*

IEEE

*Published*

DOI:10.1109/CVPRW63382.2024.00218

*Terms of use:*

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

*Publisher copyright*

IEEE postprint/Author's Accepted Manuscript

©2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collecting works, for resale or lists, or reuse of any copyrighted component of this work in other works.

(Article begins on next page)

## Pruning as a Binarization Technique

Lukas Frickenstein<sup>1</sup>, Pierpaolo Mori<sup>1,3</sup>, Shambhavi Balamuthu Sampath<sup>1</sup>, Moritz Thoma<sup>1</sup>,  
Nael Fafous<sup>1</sup>, Manoj Rohit Vemparala<sup>1</sup>, Alexander Frickenstein<sup>1</sup>, Christian Unger<sup>1</sup>,  
Claudio Passerone<sup>3</sup>, Walter Stechele<sup>2</sup>

<sup>1</sup>BMW Group, Munich, Germany; <sup>2</sup>Technical University of Munich, Munich, Germany; <sup>3</sup>Politecnico Di Torino, Turin, Italy

{<firstname>.<lastname>}<sup>1</sup>@bmw.de, <sup>2</sup>@tum.de, <sup>3</sup>@polito.it,

### Abstract

Convolutional neural networks (CNNs) can be quantized to reduce the bit-width of their weights and activations. Pruning is another compression technique, where entire structures are removed from a CNN's computation graph. Multi-bit networks (MBNs) encode the operands (weights and activations) of the convolution into multiple binary bases, where the bit-width of the particular operand is equal to its number of binary bases. Therefore, this work views pruning an individual binary base in an MBN as a reduction in the bit-width of its operands, i.e. quantization. Although many binarization methods have improved the accuracy of binary neural networks (BNNs) by e.g. minimizing quantization error, improving training strategies or proposing different network architecture designs, we reveal a new viewpoint to achieve high-accuracy BNNs, which leverages **pruning as a binarization technique (PaBT)**. We exploit gradient information that exposes the importance of each binary convolution and its contribution to the loss. We prune entire binary convolutions, reducing the effective bit-widths of the MBN during the training. This ultimately results in a smooth convergence to accurate BNNs. PaBT achieves 2.9 p.p., 1.6 p.p. and 0.9 p.p. better accuracy than SotA BNNs IR-Net, LNS and SiMaN on the ImageNet dataset, respectively. Further, PaBT scales to the more complex task of semantic segmentation, outperforming ABC-Net on the CityScapes dataset. This positions PaBT as a novel high-accuracy binarization scheme, and makes it the first to expose the potential of latent-weight-free training for compression techniques.

### 1. Introduction

Convolutional neural networks (CNNs) are the de facto standard in many computer-vision applications, such as image classification [11] and semantic segmentation [3]. The major advancements in the prediction quality were brought about by employing deeper and more complex CNN archi-

tectures. This made it difficult to meet hardware (HW) execution budgets of energy, latency, and memory, particularly in embedded scenarios. As a consequence, an increased interest in neural network compression led to the development of quantization [5, 15, 17, 19, 22, 26, 32] and pruning [12, 25, 30, 31, 37] techniques. To reduce the size and complexity of the CNN, quantization reduces the bit-width of the network operands, whereas pruning removes entire structures like channels, kernels or filters. While quantization circumvents the need for complex float arithmetic operations on HW, pruning reduces the number of computations. An extreme form of quantization are binary neural networks (BNNs), where parameters are restricted to 1-bit [2, 9, 15, 17, 18, 21, 22, 24, 26]. Early works of BNNs [7, 8, 26] have shown compelling efficiency benefits, albeit with a sizeable accuracy gap compared to their full-precision counterparts on complex datasets.

Multi-bit networks (MBNs), also referred to as multiple binary base networks, address this gap by decomposing standard convolution operations into *multiple* binary convolutions, where each binary convolution of a binary weight tensor and a binary activation tensor represents a single bit-pair interaction of an overall convolution [19, 25, 37]. Thus, MBNs encode the network parameters as a weighted sum of binary bases, where the bit-width of the particular network parameter is equal to its number of binary bases. Therefore, this paper views pruning an individual binary base in an MBN as a reduction in the bit-width of its operands, i.e. quantization. Although many binarization methods have improved the accuracy of BNNs by e.g., minimizing quantization error, improving training strategies or proposing different network architecture designs, we reveal a new viewpoint to achieve high-accuracy BNNs, which leverages **pruning as a binarization technique (PaBT)**. More specifically, PaBT is a binarization technique exploiting MBNs and gradient information to prune binary convolutions. Removing all binary convolutions of a single base (weights and/or activations) corresponds to a reduction of the effective bit-width. This enables the layer-wise, adaptive quanti-

zation of the network from its over-parameterized, original architecture *during* the training process. This progressive in-train quantization through pruning ultimately results in high-accuracy BNNs. Furthermore, PaBT unlocks the potential of latent-weight-free training (BOP [13]) to be used for pruning MBNs. This makes PaBT the first in-train compression technique using BOP. The key contributions are summarized as follows:

1. We view pruning an individual binary base in an MBN as a reduction in the bit-width of its operands. Consequently, pruning all binary convolutions of a particular base *collapses* into quantization, i.e. a bit-width reduction. To the best of our knowledge, we are the first to capitalize on this viewpoint to achieve high-accuracy BNNs.
2. We present a novel binarization scheme, namely *pruning as a binarization technique* (PaBT), which jointly learns the model parameters and the importance of single binary convolutions, thereby allowing to prune entire binary convolutions in the network. This enables layer-wise, adaptive quantization of the network from its over-parameterized architecture *during* the training smoothly converging down to an accurate BNN.
3. PaBT is the first to use a latent-weight-free optimizer [13] to preserve information of pruned binary convolutions, thereby enabling the flow of information to the remaining ones throughout the compression.

Combining the contributions, PaBT achieves better accuracy than state-of-the-art (SotA) BNNs [2, 10, 15, 17, 18, 20, 21, 23, 24, 29, 33–36] on the ImageNet dataset. Furthermore, PaBT scales to the more complex task of semantic segmentation, outperforming ABC-Net [19] on the CityScapes [6] dataset.

## 2. Related Work

**Binary Neural Networks.** Many methods on network binarization explore multiple remedies to improve the prediction capabilities, which can be classified as (1) gradient approximation techniques, (2) scaling factor formulations, (3) BNN architecture design. **Gradient approximation** techniques solve the problem of restricted gradient flows caused by discrete operations in BNNs when training with standard optimizers like stochastic gradient descent (SGD). Some works [2, 15, 17, 19, 21, 24, 26, 34] use variants of straight-through-estimator (STE) [1] to train latent weights, while others completely move away from latent representations and directly use gradients’ moving average [13]. Popular works introduced **scaling factors** [2, 17, 18, 26] to minimize the reconstruction error between real-valued parameters and their binary counterparts. This improves the representational power of BNNs at the cost of increasing the number of floating-point operations (see Tab. 1). Other recent studies [4, 21, 22] investigate different **network archi-**

**tectures** to ease the binarization process. Bi-Real Net [21] shows increased representational capability by connecting real-valued activations to binary activations of the consecutive binary convolution, i.e. a double residual connection compared to a normal residual block [11]. ReactNet [22] proposes structural changes on MobileNetV1 [14], adding parameter-free shortcuts and replacing the group convolution by regular convolution. Although many binarization methods have improved the accuracy of BNNs by exploring multiple remedies as explained above, we reveal a new viewpoint to achieve high-accuracy BNNs, which leverages pruning as a binarization technique (PaBT). The presented work differs in the three categories as follows. (1) We do not approximate gradients, as we use the gradients’ moving average [13] for updating weights *and* gradient information to decide the compression strategy, i.e. determine which binary convolution can be pruned (see Tab. 2). (2) We remove the need for additional scaling factors compared SotA BNN/MBNs (see Tab. 1). (3) We do not introduce any new network architecture complexities or additional floating-point operations. However, such network architectures can be orthogonally combined with PaBT if desired (see Tab. 3).

**Multi-Bit Quantization.** MBQ compresses CNNs through quantizing weights and activations into multiple binary bases [19, 25, 37]. ABC-Net [19] introduces scaling factors per binary bases for weights and activations,  $\alpha$  and  $\beta$  respectively (see Tab. 1). ALQ [25] quantizes weights in an exhaustive manner through many iterations of a nested optimization problem, while activations simply follow the fixed-point quantization approach from [5]. With DMBQ [37], the authors propose to consider the distribution prior of weights to optimize the quantization aspect through a look-up table, to mitigate the computational load and sub-optimal quantization schemes of prior works. In contrast, our work differentiates in the following aspects: (1) PaBT employs a one-shot prune-and-train scheme, removing the need for nested iterations and fine-tuning steps or the look-up table approaches. (2) Although other works on MBQ explore compression of CNNs through decomposing network parameters into multiple binary bases [19], and gradually reduce the bit-width to a fixed average bit-width [25, 37], PaBT compresses weights *and* activations simultaneously *down to 1-bit*, thus positioning it as a novel training scheme in the domain of BNNs.

## 3. Pruning as a Binarization Technique

In the following, we introduce the core concept of *pruning as a binarization technique* (PaBT). In Sec. 3.1, we present our viewpoint on quantized numerical representations. We make analogies between standard quantization-aware and multi-bit training techniques to link the concept of pruning binary bases with bit-width reduction. In Sec. 3.2, we

further push the flexibility of the trainable scaling factors of MBNs by switching from base-oriented scaling factors to operation-oriented scaling factors. Additionally, we remove the need to maintain latent full-precision weights for MBNs, as discussed in Sec. 3.3. Finally, in Sec. 3.4 we bring the previously presented concepts together by deriving the importance scores of all binary convolutions and pruning the least contributing ones. This allows PaBT to progressively reduce the effective bit-width of an initial over-parameterized network down to a BNN.

### 3.1. Revisiting Quantized Representations

**Multi-Bit and Integer Quantization Domains.** We reintroduce two CNN quantization domains, namely multi-bit and integer quantization. An  $L$ -layer CNN architecture consists of convolutional layers  $l \in \{1, \dots, L\}$ . Generally, the output activation  $A^l \in \mathbb{R}^{X_o \times Y_o \times C_o}$  of a convolution operation is computed from the input activation  $A^{l-1} \in \mathbb{R}^{X_i \times Y_i \times C_i}$  and the weight  $W^l \in \mathbb{R}^{K_x \times K_y \times C_i \times C_o}$ . Here,  $X_i, Y_i, C_i$  and  $C_o$  represent the dimensions of the width, height, number of input and output channels of activations respectively, while  $K_x$  and  $K_y$  represent the kernel dimensions of the weights. An MBN convolution operation involves a binary activation tensor  $A_{bin,n}^{l-1} \in \mathbb{B}^{X_i \times Y_i \times C_i}$  and a binary weight tensor  $W_{bin,m}^l \in \mathbb{B}^{K_x \times K_y \times C_i \times C_o}$  with  $\mathbb{B} = \{0, 1\}$ , followed by a multiplication of base scaling factors  $\alpha_m$  for weights and  $\beta_n$  for activations.  $n \in \{1, \dots, N\}$  and  $m \in \{1, \dots, M\}$  denote the individual binary bases of activations and weights of a layer respectively, in an  $M \times N$  MBN. Eq. 1 captures the approximation of a full-precision convolution as an MBN convolution, where a convolution of every binary base  $n$  is performed against every binary base  $m$ , and accumulated at a later stage.

$$\sum_{m=1}^M \sum_{n=1}^N \alpha_m \beta_n \text{BinaryConv}(W_{bin,m}^l, A_{bin,n}^{l-1}) \approx \text{Conv}(W^l, A^{l-1}) = A^l \quad (1)$$

Switching to the quantized convolution domain QuantConv, the scaling factor  $s_{int}$  maps the results of the convolution onto the true range of values, see Eq. 2.

$$s_{int} \text{QuantConv}(W_{int}^l, A_{int}^{l-1}) \approx \text{Conv}(W^l, A^{l-1}) = A^l \quad (2)$$

$W_{int}^l$  and  $A_{int}^{l-1}$  are integer weight and activation tensors, composed of elements  $x_{int}$ . In standard quantization-aware training (QAT) [5, 38], the integer value  $x_{int}$  in a CNN is represented by a high-precision floating-point value  $x_f$  to maintain smooth training and fine updates during backpropagation. The mapping between  $x_{int}$  and  $x_f$  takes place between standard round and clip operations as  $x_{int} = \text{round}(\text{clip}(x_f, -c, +c)/s_{int})$ , where  $c$  is the clipping threshold. Note that an integer value  $x_{int}$  represented

in bits also has bit-positional scaling factors  $s_{pos}$  applied to each bit position  $b_{pos}$ . Eq. 3 shows an example of an integer with  $N = 3$  (i.e. 3-bits) broken down to binary bits and position scaling factors  $b_{pos}$  and  $s_{pos}$ .

$$5 = 101 = 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 = \sum_{i=0}^{N-1} s_{pos_i} \times b_{pos_i} \quad (3)$$

**Linking Multi-Bit and Integer Quantization.** We formulate an analogy between the two presented quantization domains by linking Eq. 1 and Eq. 2. The scaling factors  $\alpha_m$  and  $\beta_n$  in Eq. 1 are bit-wise specific scaling factors of the binary convolutions involving bit- $m$  and bit- $n$ . All the  $M \times N$  binary convolutions constitute the overall Conv in an MBN. In an integer quantized convolution (Eq. 2),  $s_{int}$  is a *restricted* form of  $\alpha_m, \beta_n$ , where the whole quantized convolution must have one quantization scaling factor  $s_{int}$  in addition to their bit-positional scaling factor  $s_{pos}$  of its integer operands in the tensors  $W_{int}^l$  and  $A_{int}^{l-1}$ .

$$s_{int} \times s_{pos\ m} \times s_{pos\ n} \rightarrow \alpha_m \times \beta_n \quad (4)$$

The left side of Eq. 4 shows the scaling factors involved *at the bit-level* in a quantized integer arithmetic operation between a bit at position  $s_{pos\ m}$  in an element of the tensor  $W_{int}^l$  and a bit at position  $s_{pos\ n}$  in an element of the tensor  $A_{int}^{l-1}$ . Note that  $s_{int}$  is trainable in fixed-point QAT methods. However,  $s_{pos\ m}$  and  $s_{pos\ n}$  are dictated by the position of the weight bit and the activation bit involved in the computation, and cannot be chosen freely or learned directly. This *restricts* the choice of scaling factors at the bit-level in standard integer quantized convolution. However,  $\alpha_m$  and  $\beta_n$ , are both trainable and can be unique for each bit- $m$  and bit- $n$ . In Eq. 4, we can conclude that scaling factors of integer quantized convolution can be represented by MBNs, but not the other way around. This is the result of  $\alpha_m$  and  $\beta_n$  having bit-level flexibility to hold any value necessary to learn the neural network task at hand, making multi-bit quantization a more general case of standard integer quantization techniques. This fundamental understanding links MBNs [19, 25, 37] that train  $\alpha$  and  $\beta$  with QAT works [5] that train  $s_{int}$ , and allows us in the next sections to view pruning binary bases as bit-width reductions.

### 3.2. Operation-Oriented Scaling Factors

Consider a layer  $l$  in an MBN with  $M=3$  and  $N=3$ . The combinations of all possible scaling factors per MBN layer is shown in Eq. 5, where  $\alpha$  and  $\beta$  are learned for each base [19, 25, 37]. Although this is much more flexible than  $s_{int}$  as explained in Sec. 3.1, we still see that  $\alpha$  and  $\beta$  combinations are fixed. In Eq. 5, the learned  $\alpha_1$  has to be compatible with  $\beta_1, \beta_2, \beta_3$ . Therefore, the network must tweak

each base scaling factor such that it is compatible with every other scaling factor it interacts with in a binary convolution (every  $n$  against every  $m$ ).

$$\{\alpha_1\beta_1, \alpha_1\beta_2, \alpha_1\beta_3, \alpha_2\beta_1, \dots, \alpha_3\beta_3\} \rightarrow \{\gamma_{11}, \gamma_{12}, \gamma_{13}, \gamma_{21}, \dots, \gamma_{33}\} \quad (5)$$

To further improve the flexibility associated with base-oriented scaling factors  $\alpha_m$  and  $\beta_n$ , we replace them by *operation-oriented* scaling factors  $\gamma_{mn}$  for MBNs, as shown in Eq. 5, offering the following two benefits. (1) As the name implies, each *operation-oriented* scaling factor is dedicated to one specific binary convolution between a binary activation base  $n$  and a binary weight base  $m$ . The MBN can now independently learn optimal  $\gamma_{mn}$  for every binary convolution. (2) Using  $\gamma_{mn}$  grants frictionless access to the underlying binary convolution operations *during* the training. This helps PaBT derive the importance scores of binary convolutions with respect to the final loss term from gradient information. After binarization with PaBT **only one  $\gamma$  remains** for the entire convolutional layer. Sec. 3.4 further details on operation-level pruning.

Finally, we compare the number of introduced scaling factors of different BNNs. In Tab. 1, we note that XNOR-Net [26] and XNOR-Net++ [2] introduce scaling factors for convolution layers according to their dimensions. ABC-Net [19] introduces per base scaling factors along the  $N$  activation bases and  $M$  weight bases. Although PaBT uses operation-oriented scaling factors *during* training to enable a layer-wise, adaptive quantization of the network, PaBT ultimately results in a BNN, with only one remaining scaling factor per layer. In summary, **PaBT requires the least amount of scaling factors compared to the discussed BNNs**, while having a better accuracy (see Tab. 3).

Table 1. Comparison of scaling factor formulations and number (#) of scaling factors per layer with other BNN/MBNs.

Model	Scaling Factor Formulation	# of Scaling Factors per Layer
XNOR-Net [26]	$\alpha \in \mathbb{R}^{C_o}, \beta \in \mathbb{R}^{C_o}$	$2 \cdot C_o$
XNOR-Net++ [2]	$\alpha \in \mathbb{R}^{C_o}, \beta \in \mathbb{R}^{X_o}, \gamma \in \mathbb{R}^{Y_o}$	$C_o + X_o + Y_o$
ABC-Net [19]	$\alpha \in \mathbb{R}^M, \beta \in \mathbb{R}^N$	$M + N$
PaBT (Ours)	$\gamma \in \mathbb{R}$	<b>1</b>

### 3.3. Latent-Free Training of Multi-Bit Networks

For PaBT, we apply the binary optimizer (BOP) [13] concept to MBNs for the first time. This enables training binary weight bases of the MBN without maintaining latent full-precision weights and gradient approximation [1]. To flip binary weights, BOP considers the *consistency* and *strength* of gradient signals. The *consistency* captures past gradient information by maintaining the exponential moving average

of weight gradients  $\mu_t$  at training step  $t$ , see Eq. 6.

$$\mu_t = (1 - \varphi)\mu_{t-1} + \varphi g_{w,t} \quad (6)$$

$\mu_t$  considers the exponential moving average of weight gradients  $\mu_{t-1}$  of the training step  $t - 1$ , the weight gradients  $g_{w,t}$  of the training step  $t$  and an adaptivity rate  $\varphi$ . If the *strength* of gradient signals  $|\mu_t^i|$ , surpasses the threshold  $\tau$ , the binary weight  $w^i$  is flipped, see Eq. 7.

$$w_t^i = \begin{cases} -w_{t-1}^i & \text{if } |\mu_t^i| \geq \tau \text{ and } \text{sign}(\mu_t^i) = \text{sign}(w_{t-1}^i), \\ w_{t-1}^i & \text{otherwise.} \end{cases} \quad (7)$$

Using BOP, we push the accuracy improvement of MBNs (see Tab. 2). We provide an ablation study on the training hyperparameters, adaptivity rate and threshold, for MBN training with BOP in the supplementary Sec. S1. Utilizing the concept of *consistency* and *strength* of gradient signals, i.e. the gradients' history, to update binary weight bases not only helps improve the prediction capability of MBNs, but also allows information flow from pruned binary convolutions to remaining ones in PaBT. The information flow is an essential stepping stone for PaBT to smoothly prune bits down to high-accuracy BNNs, see Sec. 5.1.

### 3.4. Operation-Level Pruning

In the following, we detail the flow of PaBT as visualized in Fig. 1. Starting from (i) *importance scores*, we explain the three fundamental functions of *derive()*, *sort()* and *impose()*. Using the importance scores [25], we elaborate (ii) the *binary convolution pruning* process. Finally, by (iii) *pruning binary bases* we achieve a reduction in the effective bit-width of the MBN, which enables smooth convergence down to the final BNN.

**(i) Importance Scores.** For an *operation-oriented* scaling factor  $\gamma_{mn}$ , PaBT *derives* the importance score  $f_{\gamma_{mn}}$  for the respective binary convolution operation. This enables the technique to assess the contribution of each binary convolution operation to the loss during the training. In detail, all operation-oriented scaling factors  $\gamma_{mn}$  are trained with AMSGrad [27] providing the first and second order momentum  $u$  and  $v$ , respectively. In the context of PaBT, reusing the long-term momentum information allows for computing the importance score  $f_{\gamma_{mn}}$ , see Eq. 8.

$$f_{\gamma_{mn}} = -\kappa u_{\gamma_{mn}} \gamma_{mn} + \frac{1}{2} v_{\gamma_{mn}} (\gamma_{mn})^2 \quad (8)$$

Here,  $\kappa$  represents a weighting factor, while  $u_{mn}$  and  $v_{mn}$  represent the first and second order momentum of the respective scaling factor  $\gamma_{mn}$ . *Sorting* the importance scores of all binary convolutions in the MBN reveals the least contributing ones. The derived importance scores steer the pruning of binary convolutions *imposed* via pruning

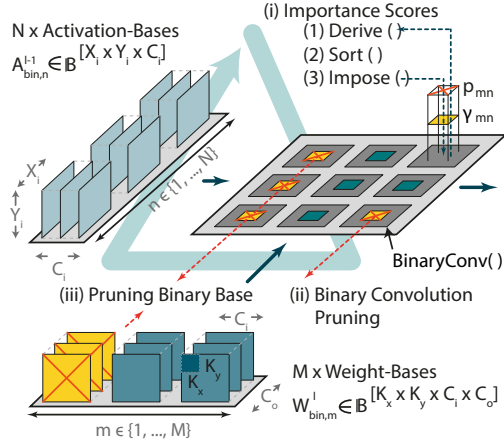


Figure 1. Visual representation of PaBT for one layer: Starting from (i) *importance scores*, the functions *derive()*, *sort()* and *impose()* are performed. Using the importance scores, (ii) a *binary convolution pruning* decision can be made. By (iii) *pruning binary bases* we achieve a reduction in the effective bit-width of the MBN, progressively resulting in a BNN.

masks. To prove the effectiveness of the proposed importance score-based pruning approach of PaBT, we compare it to a variant of PaBT with randomly chosen pruning decisions in Sec. 5.2.

**(ii) Binary Convolution Pruning.** A binary pruning mask  $P_l \in \{0, 1\}^{M \times N}$  of layer  $l$ , as shown in Eq. 9, is applied to the binary convolution operation grid visualized in Fig. 1.

$$P_l = \begin{bmatrix} p_{11} & \dots & p_{1N} \\ \vdots & \ddots & \vdots \\ p_{M1} & \dots & p_{MN} \end{bmatrix} \quad (9)$$

When the element  $p_{mn} \in P_l$  is set to zero, the output of the corresponding binary convolution is blocked, i.e. pruned. Eq. 10 shows the MBN convolution of PaBT in the training phase, with its pruning mask element  $p_{mn}$ , the operation-oriented scaling factor  $\gamma_{mn}$  and the binary convolution  $\text{BinaryConv}()$ , between the binary weight tensor  $W^l_{bin,m}$  and binary activation tensor  $A^{l-1}_{bin,n}$  of layer  $l$ .

$$\sum_{m=1}^M \sum_{n=1}^N p_{mn} \odot \gamma_{mn} \text{BinaryConv}(W^l_{bin,m}, A^{l-1}_{bin,n}) \approx \text{Conv}(W^l, A^{l-1}) = A^l \quad (10)$$

In-train pruning introduces sudden changes in loss. However, due to BOP keeping information of previous gradient updates through the momentum, the change introduced by the pruning does not suddenly disrupt the weight updates. Instead, PaBT gradually adapts to the compressed structure of the MBN through this momentum information, allowing

it to compensate for lost bit interactions and preserve prediction capabilities. This highlights the advantage of PaBT combining the concept of BOP and MBNs for the first time as our experiments have shown that using other standard optimizers did not succeed in learning high accuracy BNNs.

**(iii) Pruning Binary Bases.** At every pruning step, the least contributing binary convolution is pruned. This reduces the total number of binary convolution operations in the MBN, but does not yet result in lowering the effective bit-width. This is due to the fact, that the bases of the pruned binary convolution may still be used in the remaining ones, as shown in the top right of Fig. 1. After several pruning steps, all binary convolutions in which a particular base is used are removed. At this point, the pruning method *collapses* into quantization by resulting in a bit-width reduction due to the entire base removal (yellow tensor in Fig. 1). After several iterations of progressive quantization during the training, one final binary convolution remains per layer.

## 4. Algorithm

The PaBT method is detailed in Alg. 1. Given an  $L$ -layer MBN with bases  $M$  and  $N$  for all layers, we summarize all its parameters in  $\theta = \{\theta_w, \theta_{BN}, \theta_{SF}\}$ , where  $\theta_w$  contains all its weights,  $\theta_{BN}$  all its batch norm parameters and  $\theta_{SF}$  all its scaling factors  $\gamma_{mn}$  for all layers. We choose  $t_{start}$  and  $t_{end}$  as the train steps at which the pruning process starts and ends respectively. We provide further details on the hyperparameters  $t_{start}$  and  $t_{end}$  in the supplementary Sec. S2.  $\psi$  determines the pruning ratio over all binary convolutions of the MBN, thus defining the pruning duty for PaBT down to BNN configurations from the previously defined over-parameterized MBN. We initialize the weight parameters  $\theta_w$  from a pre-trained full-precision model. All elements of the pruning mask  $P_l^{M \times N}$  are initialized to 1, to start with an unpruned, over-parameterized model. The optimizers BOP and AMSGrad are initialized with their parameters, as introduced in previous sections. We start by computing the total number of binary convolutions to be pruned as  $prune_{total}$ , equally distributing the pruning duty  $\psi$  across available training steps. Then, we compute the number of training steps between two consecutive pruning decisions,  $t_{prune}$ . At line 4 of Alg. 1, the PaBT training starts. From lines 5-12 training updates are performed, as described in Sec. 3.3. At line 13, we check if the training step  $t$  is within the pruning phase  $t_{start}$  and  $t_{end}$ . If we are within the pruning phase, in line 14, we check if sufficient steps have passed since the last pruning decision by accounting for  $t_{prune}$ . From line 15-21, the main pruning decision steps of (1) *derive*, (2) *sort* and (3) *impose* are applied as previously shown in Fig. 1 and discussed in Sec. 3.4. After a pruning decision has been made, we assign the corresponding element  $p_{lmn}$  of the respective pruning mask  $P^l$  to zero.

---

**Algorithm 1: Pruning as a binarization technique.**

---

**Input:** Train data  $(x, y)$ ,  $L$ -layer MBN( $\cdot$ ) with parameters  $\theta = \{\theta_w, \theta_{BN}, \theta_{SF}\}$  and  $M, N$  bases,  $t_{start}, t_{end}, \psi$ .

**Init :**  $\theta_w \leftarrow \text{sign}(\theta_w) \in \{-1, 1\}$ ,  $P_l^{M \times N} \leftarrow 1$ , AMSGrad  $(\eta, u, v, \beta_1, \beta_2)$ , BOP  $(\tau, \varphi, \mu)$ .

- 1: Compute total number of convolutions to be pruned:
- 2:  $\text{prune}_{total} = L \times M \times N \times \psi$
- 3: Compute  $t_{prune}$  step size:  $t_{prune} = \frac{t_{end} - t_{start}}{\text{prune}_{total}}$
- 4: **for** Train step  $t = 1$  to  $T$  **do**
- 5:   Compute gradient:
- 6:    $g_t \leftarrow \nabla \mathcal{L}(\text{MBN}(x_t; \theta_t), y_t)$
- 7:   Update momentum:
- 8:    $u_t, v_t, \hat{v}_t$  following AMSGrad.
- 9:    $\mu_t$  following BOP.
- 10:   Update parameter:
- 11:    $\theta_{SF, t+1}$  and  $\theta_{BN, t+1}$  with AMSGrad.
- 12:    $\theta_w, t+1$  with BOP.
- 13:   **if**  $t_{start} \leq t \leq t_{end}$  **then**
- 14:     **if**  $t \bmod t_{prune} == 0$  **then**
- 15:       Derive(): Compute importance scores  $f_{\theta_{SF}}$  with Eq. 8
- 17:       Sort(): Sort and select Top-1  $f_{\theta_{SF}}$  in ascending order:
- 19:        $[l, m, n] \leftarrow f_{\theta_{SF}}.\text{index}(\min())$
- 20:       Impose(): Impose pruning decision:
- 21:        $p_{lmn} \leftarrow 0$
- 22:     **end if**
- 23:   **end if**
- 24: **end for**

---

## 5. Experiments

This section presents the results of applying PaBT on ResNet-20, ResNet-18 and DeepLabv3+ [3] evaluated on CIFAR-10 [16], ImageNet [28] and CityScapes [6] respectively. If not otherwise mentioned, the hyper-parameters are adopted from the base implementations. Note that, we follow the conventional approach of not quantizing first and last layers of the CNN [19, 21, 26]. More details about bringing PaBT to the task of semantic segmentation are provided in the supplementary Sec. S3. The bit-width of weights and activations is denoted as  $I_W$  and  $I_A$  respectively.

### 5.1. Exploring Scaling Factors, Optimizer and Pruning Capabilities

We present the investigation of base-oriented ( $\alpha, \beta$ ) and operation-oriented ( $\gamma$ ) scaling factors ( $\theta_{SF}$ ) proposed for PaBT. Different optimizer settings and PaBT-based binarization of MBNs for different network architectures and

Table 2. Influence of the scaling factors  $\theta_{SF}$ , the used optimizer and operation level pruning, in terms of number of bit-operations (Bit-OPs) and Top-1 accuracy. Note that solutions with  $\alpha$  and  $\beta$  represent the implementation from [19].

Model/ Dataset	$\theta_{SF}$	Optimizer (Parameter Scope)	Operation Pruning	Bit-Width		Top-1 [%]	
				$I_W$	$I_A$		
ResNet-20 CIFAR-10	-	ADAM ( $\theta$ )	✗	8	8	92.4	
	$\alpha, \beta$	ADAM ( $\theta$ )	✗	3	3	89.61	
	$\gamma$	ADAM ( $\theta$ )	✗			89.63	
	$\gamma$	ADAM ( $\theta_{BN, SF}$ ), BOP ( $\theta_W$ )	✗	90.00			
	$\alpha, \beta$	ADAM ( $\theta$ )	✗	1	1	83.85	
	$\gamma$	ADAM ( $\theta$ )	✗			83.93	
	$\gamma$	ADAM ( $\theta_{BN, SF}$ ), BOP ( $\theta_W$ )	✗	84.27			
	$\gamma$	AMSGrad ( $\theta_{BN, SF}$ ), BOP ( $\theta_W$ )	✓	1	1	<b>86.59</b>	
	ResNet-18 ImageNet	-	ADAM ( $\theta$ )	✗	8	8	69.3
		$\alpha, \beta$	AMSGrad ( $\theta$ )	✗	3	3	61.99
$\gamma$		AMSGrad ( $\theta$ )	✗	62.69			
$\gamma$		AMSGrad ( $\theta_{BN, SF}$ ), BOP ( $\theta_W$ )	✗	62.86			
$\alpha, \beta$		AMSGrad ( $\theta$ )	✗	1	1	55.44	
$\gamma$		AMSGrad ( $\theta$ )	✗			55.55	
$\gamma$		AMSGrad ( $\theta_{BN, SF}$ ), BOP ( $\theta_W$ )	✗	56.02			
$\gamma$	AMSGrad ( $\theta_{BN, SF}$ ), BOP ( $\theta_W$ )	✓	1	1	<b>57.51</b>		

datasets, with respect bit-width and Top-1 accuracy are presented in Tab. 2. The effectiveness of adopting BOP in PaBT is shown by comparing to experiments where the network parameters are optimized by ADAM or AMSGrad. Further experiments using BOP for weight parameters ( $\theta_w$ ) optimization in combination with ADAM or AMSGrad for batch norm parameters ( $\theta_{BN}$ ) and scaling factors ( $\theta_{SF}$ ) optimization are carried out. **(1) Results with  $\gamma$  show improvements from our proposed operation-oriented scaling factors. (2) Results combining AMSGrad and BOP show our improvements of introducing the latent-weight-free training approach. (3) Results with operation pruning activated (✓) show improvements of our overall PaBT method.**

Interestingly, switching from one global optimizer for  $\theta$  to two optimizer scopes and updating  $\theta_w$  with BOP results in only slight improvements throughout all experiments, demonstrated by an increase of 0.47 p.p. in Top-1 for the 1-bit solution on ImageNet. Consistently, PaBT produces dominating solutions (Top-1) through pruning of an over-parameterized MBN down to 1-bit, compared to directly training a 1-bit network with latent-weight-free optimization and operation-oriented scaling factors, on all experiments. This manifests in a boost of 1.49 p.p. in Top-1 for ResNet-18 on ImageNet. The experiments shown in this section demonstrate the superiority of PaBT in learning from its over-parameterized, original network architecture and smooth flow of information onto remaining binary con-

volutions after a pruning step (as indicated by experiments highlighted by (✓) for operation pruning). It is interesting to note, that replacing latent-weight-free training with latent-weight representation during pruning did not succeed in maintaining accuracy of the resulting network.

## 5.2. Does Pruning the Right Operations Matter?

In Fig. 2, we demonstrate the effectiveness and training behaviour of PaBT on ImageNet. We highlight the relevance of the importance scores, as proposed in Sec. 3.4, by visualizing the training behaviour of two experiments which vary in the pruning criterion (PaBT-I and PaBT-R), compared to the baseline, all averaged over three runs. The dotted lines indicate the pruning start ( $t_{\text{start}}$ ) and end ( $t_{\text{end}}$ ) for the PaBT experiments. PaBT-R prunes binary convolution operations randomly, while PaBT-I prunes them based on importance scores. The baseline ResNet-18 MBN with  $M=N=1$  is trained in a latent-weight-free manner using BOP [13] and achieves a Top-1 accuracy of 56.02%, indicated in green. Improving on that result, PaBT-R is used to train and prune an over-parameterized ResNet-18 with  $M=N=3$  down to a ResNet-18 with  $M=N=1$ , where the pruned binary convolution operations are selected randomly across all layers and operations (indicated in red), resulting in a Top-1 accuracy of 57.04%. Lastly, PaBT-I utilizes the importance scores as the pruning criterion, where the training behavior is indicated in blue and produces a Top-1 of 57.51%. This shows the benefit of the *importance scores* over the *random* pruning criterion. In summary, the above empirically supports two claims: (1) PaBT network binarization from an over-parameterized model down to BNNs outperforms training models directly from BNN architectures (both red/blue outperform green) and (2) using gradient information to assess the importance of binary convolutions in PaBT (i.e. making good pruning decisions) is better than randomly removing binary convolutions (blue outperforms red). Note that all three configurations involve contributions proposed by this work in Sec. 3.

## 5.3. Comparison to State-of-the-Art

We compare PaBT with various binarization techniques [2, 10, 15, 17–21, 23, 24, 26, 29, 33–36], all aiming to produce highly accurate BNNs. Tab. 3 compares our PaBT with SotA BNNs in terms of Top-1/mIoU, where PaBT prunes binary convolutions of an over-parameterized MBN with  $M=N=3$  down to the BNN configuration. PaBT outperforms all other BNN techniques on ResNet-20 architecture, and achieves on-par results with RBNN [17]. To showcase the versatility of PaBT to be combined with orthogonal network architecture complexities (recall related work, e.g. Bi-Real Net [21], ReactNet [22]), we take the Bi-Real structure as an example along with the normal ResNet [11] structure for the ImageNet dataset, indicated by <sup>2</sup> and <sup>1</sup> respectively.

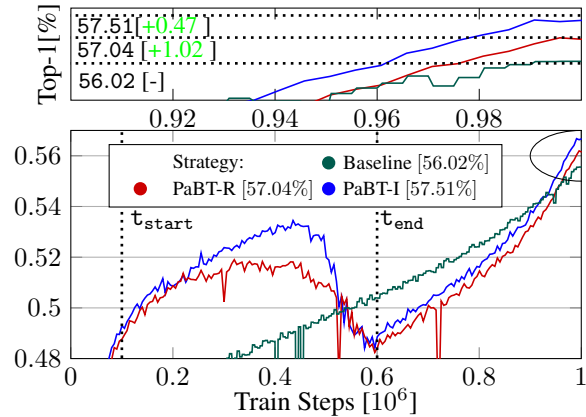


Figure 2. Comparison of the training behavior of PaBT relying on *random selection* (red) and *importance score* (blue) as a pruning criterion to determine the pruned binary convolution of an over-parameterized ResNet-18 MBN with  $M=N=3$  down to a BNN. This achieves the same computational complexity (Bit-OPs) as a ResNet-18 MBN with  $M=N=1$  (green) directly learned with latent-weight-free training.

On the more complex ImageNet-trained ResNet-18, PaBT results are again outperforming all other techniques. This manifests in 3.3 p.p. and 0.4 p.p. better accuracy than UAD [15] and XNOR++ [2] using normal ResNet structure. The trend holds for works with Bi-Real structure, where PaBT achieves 2.9 p.p., 1.6 p.p. and 0.9 p.p. better accuracy than SotA BNNs IR-Net [24], LNS [10] and SiMaN [18] on ImageNet, respectively. Semantic segmentation is notoriously difficult for BNNs, nonetheless the PaBT BNN variant of DeepLabv3+ still achieves a 3.72 p.p. mIoU improvement over an equivalent ABC-Net MBN with  $M=N=1$ . This showcased the scalability to larger models, as we quantized the ResNet-18 backbone *as well as* the decoder layers as they hold the majority of computational complexity using PaBT. It is important to note that the results presented, not only outperform SotA but also (1) do not rely on gradient approximations, (2) do not introduce additional scaling factors and (3) are orthogonal to network architecture complexities (proven on normal [<sup>1</sup>] and Bi-Real [<sup>2</sup>] structure).

**Discussion.** Last, we want to compare achievable results of SotA BNN works, categorized by the underlying SotA BNN baseline architectures. Tab. 4 highlights the range of achievable results of SotA BNN methods building on top of the SotA BNN baseline architectures, namely standard ResNet [11]/XNOR-Net [26] (51.2 – 57.5), Bi-Real Net [21] (56.4 – 61.0) and ReactNet [22] (65.5+). Fig. 3 visualizes the major differences of the BNN baseline architectures in the downsampling layers. Note the *full precision convolution* (CONV) used in both Bi-Real Net and ReactNet (highlighted in red) and additional modules (e.g. BiRealSign, ReactSign) *surrounding* the

Table 3. Comparison of PaBT with SotA model compression approaches. <sup>[1]</sup> and <sup>[2]</sup> indicate results using ResNet with normal structure [11] and with Bi-Real structure [21].

Model/ Dataset	Method	$I_W/I_A$	Top-1/mIoU [%]
ResNet-20 CIFAR-10	ResNet-20 [11]	8/8	92.48
	ABC-Net [19]	1/1	83.85
	XNOR-Net [26]	1/1	83.98
	MBN-Net + BOP (Ours)	1/1	84.27
	IR-Net [24]	1/1	85.50*
	LNS [10]	1/1	85.70*
	LCR [29]	1/1	86.00*
	FDA-BNN [35]	1/1	86.20*
	XNOR-Net [26] + BOP [13]	1/1	86.30
	RBNN [17]	1/1	86.51*
<b>PaBT (Ours)</b>	1/1	<b>86.59</b>	
ResNet-18 ImageNet	ResNet-18 [11]	8/8	69.01
	XNOR-Net <sup>1</sup> [26]	1/1	51.20
	MAD <sup>1</sup> [23]	1/1	52.00*
	ABC-Net <sup>1</sup> [19]	1/1	52.23
	BNN-UAD <sup>1</sup> [15]	1/1	54.20*
	XNOR-Net <sup>1</sup> [26] + BOP [13]	1/1	55.08
	MBN-Net <sup>1</sup> + BOP (Ours)	1/1	55.86
	Bi-Real Net <sup>2</sup> [21]	1/1	56.40*
	XNOR++ <sup>1</sup> [2]	1/1	57.10*
	BNN-UAD <sup>2</sup> [15]	1/1	57.20*
	IR-Net <sup>2</sup> [24]	1/1	58.10*
	BONN <sup>2</sup> [36]	1/1	59.30*
	LNS <sup>2</sup> [10]	1/1	59.40*
	RBCN <sup>2</sup> [20]	1/1	59.50*
	LCR <sup>2</sup> [29]	1/1	59.60*
	Si-BNN <sup>2</sup> [33]	1/1	59.70*
	RBNN <sup>2</sup> [17]	1/1	59.90*
	SiMaN <sup>2</sup> [18]	1/1	60.10*
	FDA-BNN <sup>2</sup> [35]	1/1	60.20*
	ReSTE <sup>2</sup> [34]	1/1	60.88*
PaBT <sup>1</sup> (Ours)	1/1	57.51	
<b>PaBT<sup>2</sup> (Ours)</b>	1/1	<b>61.02</b>	
DeepLabv3+ CityScapes	DeepLabv3+ [3]	8/8	68.53
	ABC-Net [19]	1/1	50.95
	MBN-Net + BOP (Ours)	1/1	51.1
	PaBT (Ours)	1/1	<b>54.67</b>

binary convolutions (bConv) (highlighted in green). PaBT outperforms SotA BNN works on compared architectures, XNOR-Net and Bi-Real Net. Notably, all BNN works building on top of ReactNet [22] generally have higher baselines, due to structural changes surrounding the binary convolution block. Differently, PaBT contributes inside the binary convolution block in Fig. 3 making it orthogonal to other techniques as proven on Bi-Real/XNOR-Net structures.

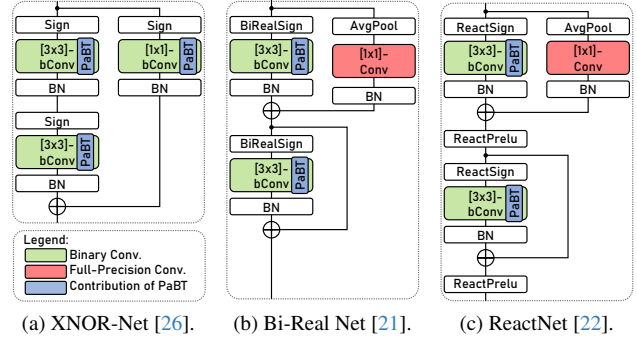


Figure 3. Structure of downsampling layers in BNN baseline architectures.

Table 4. Comparison of SotA BNN works, categorized by used BNN baseline architectures, namely standard XNOR-Net [26], Bi-Real Net [21] and ReactNet [22]. Benchmark for ResNet-18 trained on ImageNet. [-] no result is reported.

Method	Published	XNOR-Net [51.20]	Bi-Real Net [56.40]	ReactNet [65.50]
RBCN [20]	IJCAI19	-	59.50	-
XNOR++ [2]	BMVC19	57.10	-	-
IR-Net [24]	CVPR20	56.90	58.10	66.50
LNS [10]	PMLR20	-	59.40	-
RBNN [17]	NeurIPS20	-	59.90	-
MAD [23]	BMVC21	52.00	57.90	66.50
BNN-UAD [15]	CVPR21	54.20	57.20	-
FDA-BNN [35]	NeurIPS21	-	60.20	66.00
SiMaN [18]	TPAMI22	-	60.10	66.10
BONN [36]	IJCV22	-	59.30	-
ReSTE [34]	ICCV23	-	60.88	-
<b>PaBT (Ours)</b>		<b>57.51</b>	<b>61.02</b>	-

## 6. Conclusion

We presented a viewpoint on pruning in the domain of multi-bit networks to effectively achieve in-train binarization. With this viewpoint, we proposed a novel compression method which leverages pruning as a binarization technique (PaBT) by exploiting gradient information and revealing the importance of each binary convolutions and its contribution to the loss. Additionally, we applied the concept of BOP to MBNs for the first time, which allowed us to improve their baseline accuracies and further facilitate the flow of information in our in-train pruning technique. By pruning all binary convolutions of a particular base, we reduced the effective bit-width of the MBN being trained, at which point pruning collapsed into quantization during the training, ultimately resulting in the gradual convergence down to the final BNN. PaBT outperforms SotA BNNs on the ImageNet dataset and scales to the more complex task of semantic segmentation. This positions PaBT as a novel high-accuracy, binarization technique, and makes it the first to expose the potential of latent-weight-free training for compression.

## References

- [1] Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *ArXiv*, abs/1308.3432, 2013. [2](#), [4](#)
- [2] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2019. [1](#), [2](#), [4](#), [7](#), [8](#)
- [3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. [1](#), [6](#), [8](#)
- [4] Tianlong Chen, Zhenyu Zhang, Xu Ouyang, Zechun Liu, Zhiqiang Shen, and Zhangyang Wang. "bnn - bn = ?": Training binary neural networks without batch normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4619–4629, 2021. [2](#)
- [5] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I.-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized Clipping Activation for Quantized Neural Networks. *ArXiv*, abs/1805.06085, 2018. [1](#), [2](#), [3](#)
- [6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [2](#), [6](#)
- [7] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2015. [1](#)
- [8] 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](#)
- [9] Sieger Falkena, Hadi Jamali-Rad, and Jan van Gemert. Lab: Learnable activation binarizer for binary neural networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 6425–6434, 2023. [1](#)
- [10] Kai Han, Yunhe Wang, Yixing Xu, Chunjing Xu, Enhua Wu, and Chang Xu. Training binary neural networks through learning with noisy supervision. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 4017–4026. PMLR, 2020. [2](#), [7](#), [8](#)
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. [1](#), [2](#), [7](#), [8](#)
- [12] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. [1](#)
- [13] Koen Helwegen, James Widdicombe, Lukas Geiger, Zechun Liu, Kwang-Ting Cheng, and Roeland Nusselder. Latent weights do not exist: Rethinking binarized neural network optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2019. [2](#), [4](#), [7](#), [8](#)
- [14] 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 preprint arXiv:1704.04861*, 2017. [2](#)
- [15] Hyungjun Kim, Jihoon Park, Changhun Lee, and Jae-Joon Kim. Improving accuracy of binary neural networks using unbalanced activation distribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7862–7871, 2021. [1](#), [2](#), [7](#), [8](#)
- [16] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *University of Toronto*, 2012. [6](#)
- [17] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Yan Wang, Yongjian Wu, Feiyue Huang, and Chia-Wen Lin. Rotated binary neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7474–7485. Curran Associates, Inc., 2020. [1](#), [2](#), [7](#), [8](#)
- [18] Mingbao Lin, Rongrong Ji, Zihan Xu, Baochang Zhang, Fei Chao, Chia-Wen Lin, and Ling Shao. Siman: Sign-to-magnitude network binarization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. [1](#), [2](#), [7](#), [8](#)
- [19] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards Accurate Binary Convolutional Neural Network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. [1](#), [2](#), [3](#), [4](#), [6](#), [8](#)
- [20] Chunlei Liu, Wenrui Ding, Xin Xia, Yuan Hu, Baochang Zhang, Jianzhuang Liu, Bohan Zhuang, and Guodong Guo. Rectified binary convolutional networks for enhancing the performance of 1-bit dcnn. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 854–860. International Joint Conferences on Artificial Intelligence Organization, 2019. [2](#), [8](#)
- [21] Zechun Liu, Baoyuan Wu, Wenhao Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. 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)*, pages 722–737, 2018. [1](#), [2](#), [6](#), [7](#), [8](#)
- [22] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision (ECCV)*, 2020. [1](#), [2](#), [7](#), [8](#)
- [23] Ying Nie, Kai Han, and Yunhe Wang. Multi-bit adaptive distillation for binary neural networks. In *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*, page 61. BMVA Press, 2021. [2](#), [7](#), [8](#)
- [24] 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. In *Proceedings of the IEEE/CVF Conference on*

- Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 7, 8
- [25] Zhongnan Qu, Zimu Zhou, Yun Cheng, and Lothar Thiele. Adaptive loss-aware quantization for multi-bit networks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7985–7994, 2020. 1, 2, 3, 4
- [26] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. 1, 2, 4, 6, 7, 8
- [27] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2018. 4
- [28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 6
- [29] Yuzhang Shang, Dan Xu, Bin Duan, Ziliang Zong, Liqiang Nie, and Yan Yan. Lipschitz continuity retained binary neural network. In *Computer Vision – ECCV 2022*, pages 603–619, Cham, 2022. Springer Nature Switzerland. 2, 7, 8
- [30] Manoj-Rohit Vemparala, Nael Fasfous, Alexander Frickenstein, Mhd Ali Moraly, Aquib Jamal, Lukas Frickenstein, Christian Unger, Naveen-Shankar Nagaraja, and Walter Stechele. L2pf-learning to prune faster. In *International Conference on Computer Vision and Image Processing*, pages 249–261. Springer, 2020. 1
- [31] Manoj-Rohit Vemparala, Nael Fasfous, Alexander Frickenstein, Sreetama Sarkar, Qi Zhao, Sabine Kuhn, Lukas Frickenstein, Anmol Singh, Christian Unger, Naveen-Shankar Nagaraja, Christian Wressnegger, and Walter Stechele. Adversarial robust model compression using in-train pruning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 66–75, 2021. 1
- [32] Manoj Rohit Vemparala, Nael Fasfous, Lukas Frickenstein, Alexander Frickenstein, Anmol Singh, Driton Salihu, Christian Unger, Naveen Shankar Nagaraja, and Walter Stechele. Hardware-aware mixed-precision neural networks using in-train quantization. In *BMVC*, page 60, 2021. 1
- [33] Peisong Wang, Xiangyu He, Gang Li, Tianli Zhao, and Jian Cheng. Sparsity-inducing binarized neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):12192–12199, 2020. 2, 7, 8
- [34] X. Wu, D. Zheng, Z. Liu, and W. Zheng. Estimator meets equilibrium perspective: A rectified straight through estimator for binary neural networks training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 17009–17018, Los Alamitos, CA, USA, 2023. IEEE Computer Society. 2, 8
- [35] Yixing Xu, Kai Han, Chang Xu, Yehui Tang, Chunjing Xu, and Yunhe Wang. Learning frequency domain approximation for binary neural networks. In *Neural Information Processing Systems (NeurIPS)*, 2021. 8
- [36] Junhe Zhao, Sheng Xu, Baochang Zhang, Jiaxin Gu, David Doermann, and Guodong Guo. Towards compact 1-bit cnns via bayesian learning. *International Journal of Computer Vision*, 130, 2022. 2, 7, 8
- [37] Sijie Zhao, Tao Yue, and Xuemei Hu. Distribution-aware adaptive multi-bit quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9281–9290, 2021. 1, 2, 3
- [38] 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. 3