Fast and Accurate Binary Neural Networks based on Depth-Width Reshaping

Ping Xue[1], Yang Lu[1,2,3*], Jingfei Chang[1], Xing Wei[1,2,4], Zhen Wei[1,2,3]

[1]School of Computer Science and Information Engineering, Hefei University of Technology, Hefei, China
[2]Anhui Mine IOT and Security Monitoring Technology Key Laboratory, Hefei, China
[3]Engineering Research Center of Safety Critical Industrial Measurement and Control Technology, Ministry of Education, Hefei University of Technology, Hefei, China
[4]Intelligent Manufacturing Institute of HeFei University of Technology, Hefei, China
{xueping, cjfhfut}@mail.hfut.edu.cn, luyang.hf@126.com, weixing@hfut.edu.cn, weizhen@gocom.cn

Abstract

Network binarization (i.e., binary neural networks, BNNs) can efficiently compress deep neural networks and accelerate model inference but cause severe accuracy degradation. Existing BNNs are mainly implemented based on the commonly used full-precision network backbones, and then the accuracy is improved with various techniques. However, there is a question of whether the full-precision network backbone is well adapted to BNNs. We start from the factors of the performance degradation of BNNs and analyze the problems of directly using full-precision network backbones for BNNs: for a given computational budget, the backbone of a BNN may need to be shallower and wider compared to the backbone of a full-precision network. With this in mind, Depth-Width Reshaping (DWR) is proposed to reshape the depth and width of existing full-precision network backbones and further optimize them by incorporating pruning techniques to better fit the BNNs. Extensive experiments demonstrate the analytical result and the effectiveness of the proposed method. Compared with the original backbones, the DWR backbones constructed by the proposed method result in close to $O(\sqrt{s})$ decrease in activations, while achieving an absolute accuracy increase by up to 1.7% with comparable computational cost. Besides, by using the DWR backbones, existing methods can achieve new state-of-the-art (SOTA) accuracy (e.g., 67.2% on ImageNet with ResNet-18 as the original backbone). We hope this work provides a novel insight into the backbone design of BNNs. The code is available at https://github.com/pingxue-hfut/DWR.

1 Introduction

The pioneering study of network binarization dates back to BinaryNet (Hubara et al. 2016). BinaryNet (Hubara et al. 2016) constrains both weights and activations to {-1, +1} and exhibits high computational and storage efficiency. However, its drawback is also obvious that binarization may cause severe performance degradation. The performance degradation of binary neural networks (BNNs) can be analyzed from two perspectives: forward propagation and backward propagation. In the forward propagation, the floating-point weights and activations are binarized, which introduces the quantization error, causes information loss, and reduces the network representational capability; in the backward propagation, the derivative of the sign function is zero almost everywhere, which makes it incompatible with the backward propagation algorithm, and therefore gradient approximation is required to train BNNs, such as "Straight-Through Estimator" (STE) (Bengio, Léonard, and Courville 2013), which inevitably introduces the gradient error. In addition, inaccurate gradients may bias the optimization direction, causing difficulties in training BNNs and further degradation of the task performance. Therefore, existing studies on BNNs either proceed from the forward propagation to minimize the quantization error, reduce information loss, and enhance the representational capability of networks (Rastegari et al. 2016; Lin et al. 2020; Xue et al. 2022); or start with the backward propagation to improve the gradient approximation method and then the BNN training (Tung and Mori 2020; Martínez et al. 2020; Kim et al. 2020); or consider a combination of both the forward and backward propagation (Liu et al. 2020a; Qin et al. 2020; Liu et al. 2020b). All these approaches have improved the task performance of BNNs to some extent. However, existing BNNs are mainly implemented based on the commonly used full-precision network backbones, and then studied for optimization on this basis, but very few studies have considered whether the full-precision network backbones are appropriate for BNNs.

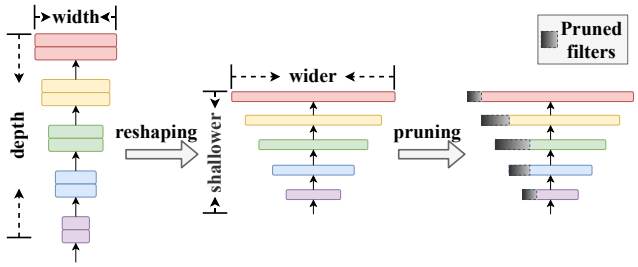

Figure 1: Overview of Depth-Width Reshaping

.

Network depth refers to the number of layers of a network, while the width represents the number of channels per layer. It is well-known that without considering overfitting, increasing either the width or the depth of a full-precision

*Corresponding author.

network usually enhances the network representational capability and thus the task performance of the network. Existing studies have found that the gains of increasing the depth polynomially are comparable to those of increasing the width exponentially (Delalleau and Bengio 2011; Eldan and Shamir 2016; Lu et al. 2017), i.e., given a computational budget (e.g., floating-point operations, FLOPs), increasing the depth generally yields greater network performance gains than increasing the width does. Therefore, existing full-precision networks may prefer deep and narrow backbones in terms of the computational cost. However, the binarization of weights and activations in BNNs introduces the quantization error and gradient error, which leads to performance degradation. In addition, the errors may accumulate layer by layer as the network goes deeper (Qin et al. 2020; Kim et al. 2020; Ye, Wang, and Zhang 2021). It could reduce the gains from increasing depth, leading to a change in the relationship between the gains obtained from increasing depth and increasing width, thus making the backbones of full-precision networks (depth/width configuration) potentially unsuitable for BNNs.

To address the above issues, we analyze the effect of binarization on the gain relationship between the depth and width of networks from theoretical and empirical perspectives and improving the task performance of BNNs by reshaping the depth and width of the existing full-precision network backbones to make them more suitable for BNNs. To flexibly constrain the computational complexity and optimize the network backbones, we introduce pruning techniques into BNNs to further improve the network performance (see the overview in Figure 1). Notably, this study aims to reconstruct network backbones more suitable for BNNs to obtain strong baselines, which is orthogonal to existing BNN optimization methods and could be coupled with existing advanced methods to obtain better task performance.

The main contributions can be summarized as follows.

1) To the best of our knowledge, we are the first to analyze the potential problems that full-precision network backbones are directly used for BNNs starting from the causes of performance degradation of BNNs, and on the basis of the analysis, reshaping existing full-precision network backbones to better fit BNNs, providing a novel insight into the backbone design of BNNs.

2) We analyze the changes in the gain relationship between the depth and width in BNNs relative to full-precision networks from theoretical and empirical perspectives.

3) We incorporate network binarization and pruning techniques to obtain new baseline backbones for BNNs that are more efficient and accurate.

4) Extensive experiments demonstrate the effectiveness of the proposed method. Furthermore, by coupling with existing BNN optimization methods, new state-of-the-art (SOTA) accuracy is achieved.

2 Related Work

Network Binarization: Network binarization dramatically reduces model storage and speeds up network inference. However, naive binarization of the model may cause severe performance degradation. The main reasons for the performance degradation of BNNs are that weight and activation binarization introduce the quantization error and gradient error, which causes information loss, reduces the network representational capability, and makes network training difficult. Therefore, most works try to solve the above problems to close the performance gap between BNNs and their full-precision counterparts. XNOR-Net (Rastegari et al. 2016) introduces the real-valued scaling factors to reduce the quantization error. XNOR-Net++ (Bulat and Tzimiropoulos 2019) makes further improvements by fusing the separated weight and activation scaling factors into one. Real-to-Bin (Martínez et al. 2020) and RB-Net (Liu et al. 2022) obtain the activation scaling factors via SE (Hu et al. 2020). Bi-Real Net (Liu et al. 2020a) connects the real-valued activation of adjacent layers to enhance the model capability. IR-Net (Qin et al. 2020), BBG (Shen et al. 2020), and Equal Bits (Li, Pintea, and van Gemert 2022) maximize the information entropy by adjusting the weight distribution to reduce information loss. UAD (Kim et al. 2021), ReActNet (Liu et al. 2020b), and SD-BNN (Xue et al. 2022) improve the diversity by adjusting activation distribution. ANE (Zhang et al. 2021) uses a neural encoder (NE) instead of the sign function to binarize the weights to alleviate the gradient error problem. DGRL (Ye, Wang, and Zhang 2021) combines knowledge distillation to optimize BNN training. Real-to-Bin (Martínez et al. 2020) and ReActNet (Liu et al. 2020b) divided BNN training into two phases to stabilize the training process, which greatly improved the task performance of BNNs. Although great progress has been made in network binarization, these methods are mainly based on existing full-precision network backbones for binarization and do not consider the problem of backbones applicability. Unlike the existing methods, we analyze the changes in the applicability of BNNs to network backbones compared to that of full-precision networks starting from the problems raised by binarization itself, and reshaping network backbones by incorporating pruning methods to obtain faster and more accurate BNNs.

Network Pruning: Network pruning is another popular practice in model compression, which focuses on reducing the model size by removing unimportant or redundant parameters from the network. According to the pruning granularity, it can be categorized into element-wise, group-wise, strip-wise, filter-wise, channel-wise, and layer-wise (Liu, Zhang, and Lv 2022), the key to network pruning is the importance evaluation of network parameters. This line is orthogonal to network binarization. Inspired by pruning techniques, we use them as an architecture search method to further constrain the network computational complexity and optimize the network backbones.

Network Scaling: Another line close to this work is network scaling. Network scaling refers to scaling a base convolutional neural network to endow it with greater computational complexity and representational capability (Dollár, Singh, and Girshick 2021). Typical scaling strategies include scaling depth (He et al. 2016), width (Zagoruyko and Komodakis 2016), resolution (Huang et al. 2019), and compound scaling (i.e., scaling depth, width, and resolution si-

multaneously) (Tan and Le 2019). The main difference is that network scaling target full-precision networks and aim to increase the given computational budget in exchange for the maximum performance gains, whereas we analyze theoretically and empirically the impact of binarization on the performance gains from adjusting network width and depth, and obtains more suitable baseline backbones for BNNs by reshaping the existing full-precision network backbones and incorporating pruning techniques while maintaining or reducing the computational cost.

3 Preliminaries

3.1 Binary Neural Networks

The basic operation in full-precision convolutional neural networks (CNNs) is:

$$z = \omega_r \otimes a_r \tag{1}$$

where ω_r denotes the real-valued weights, a_r the real-valued input activations, and $\otimes$ the real-valued convolution. Generally, the convolution with batch normalization (BN) and nonlinear activation forms the convolutional layer. A CNN is a stack of a task-related algorithm $T(\cdot)$ (e.g., classification, object detection) and l convolutional layers. Given the input I, the output of the full-precision network is:

$$O_r = T(R^l(\cdots R^2(R^1(I))\cdots)) \tag{2}$$

where $R^l(\cdot)$ denotes the lth real-valued convolutional layer transform with real-valued parameters. A BNN is similar to the full-precision network, with the main difference being the use of 1-bit weights and activations, so its basic operation becomes:

$$\begin{gathered} \omega_b = sign(\omega_r),\ a_b = sign(a_r) \\ z = \omega_b \oplus a_b \end{gathered} \tag{3}$$

where ω_b and a_b denote the binary weights and input activations, respectively, $\oplus$ the binary convolution, and $sign(\cdot)$ the binarization function used to convert the real-valued weights and activations into binary ones, and the function form is:

$$sign(x) = \begin{cases} +1, & if\ x \geq 0 \\ -1, & otherwise \end{cases} \tag{4}$$

For the input I, the output of the BNN becomes:

$$O_b = T(B^l(\cdots B^2(B^1(I))\cdots)) \tag{5}$$

where $B^l(\cdot)$ denotes the lth binary convolutional layer transform with binary parameters.

3.2 Network Complexity

Increasing either the network depth or width can enhance the network representational capability and improve the task performance, but it also increases the network complexity, so the effect of adjusting depth/width on network complexity needs to be considered. Network complexity mainly refers to the number of operations (OPs) and parameters, while (Liu et al. 2021) and (Dollár, Singh, and Girshick 2021) show that the number of activations plays a major role in determining the speedups on modern memory-bandwidth limited hardware, so the number of activations is taken into account as a measure of network complexity as well. According to the characteristics of convolutional networks, following the statistical approach of (Dollár, Singh, and Girshick 2021), then the effect of scaling depth and width on network complexity is shown in Table 1, where w and d denote the width and depth of the network, respectively, r^2 denotes the spatial resolution $r \times r$, s the factor of scaling depth/width.

Dim	Scaling		OPs	Parameters	Activations
none	d	w	dw^2r^2	dw^2	dwr^2
d	sd	w	sdw^2r^2	sdw^2	$sdwr^2$
w	d	$\sqrt{s}w$	sdw^2r^2	sdw^2	$\sqrt{s}dwr^2$

Table 1: Complexity of scaling depth/width

As can be seen from Table 1, constraining OPs can simultaneously constrain the complexity of the other two items. Therefore, in this work, the complexity is measured in terms of OPs when evaluating the complexity budget:

$$C = \psi(w, d) \propto dw^2 \tag{6}$$

It is worth noting that the OPs of a full-precision network is the FLOPs, whereas the OPs of a BNN contains both FLOPs and binary operations (BOPs). In line with ReActNet (Liu et al. 2020b), the OPs is computed using $OPs = BOPs/64 + FLOPs$ for BNNs.

3.3 Motivation

Feature representation is critical for deep learning. Generally increasing either the depth or width of a network would enhance the model capability. (Pascanu, Montufar, and Bengio 2013) shows that for a constant input n_0, the representational capability of a shallow model with kn hidden neurons is $\mathrm{O}(k^{n_0}n^{n_0})$, whereas that of a deep model with k layers of n hidden neurons per layer is $\Omega(\lfloor n/n_0 \rfloor^{k-1} n^{n_0})$, i.e., given a computational budget, the deep model outperforms the shallow model. (Delalleau and Bengio 2011; Eldan and Shamir 2016; Lu et al. 2017) further demonstrate the idea and show that increasing the depth polynomially yields comparable model performance gains as increasing the width exponentially. Let G be the performance gains (e.g., classification accuracy) of the model,

$$\begin{cases} G = f(w, d), \\ w \in [w_{\min}, w_{\max}] \\ d \in [d_{\min}, d_{\max}] \end{cases} \tag{7}$$

where $[w_{\min}, w_{\max}]$ and $[d_{\min}, d_{\max}]$ are reasonable ranges of width and depth, respectively. Then in a full-precision network, G has the following properties:

a) In the domain of definition, $f(w, d) > 0$, and increasing either the width or the depth yields performance gains,

$$\begin{cases} f'_d = \frac{\partial f(w,d)}{\partial d} > 0 \\ f'_w = \frac{\partial f(w,d)}{\partial w} > 0 \end{cases} \tag{8}$$

b) Normally, given a computational budget, increasing the depth will yield greater gains than that of increasing the width,

$$\frac{\partial f(w, d)}{\partial d} > \frac{\partial f(w, d)}{\partial w} \tag{9}$$

c) Performance gains slow down as depth/width grows (Tan and Le 2019),

$$\begin{cases} f''_d = \frac{\partial f'_d(w,d)}{\partial d} < 0 \\ f''_w = \frac{\partial f'_w(w,d)}{\partial w} < 0 \end{cases} \tag{10}$$

d) When width remains constant, the performance gains obtained by increasing the depth gradually slow down until saturation. Thus, there is an equilibrium point $d = d_0$, such that the gains from further increases in depth are not greater than the gains from increasing the width,

$$\begin{cases} \frac{\partial f(w,d_0)}{\partial d} = \frac{\partial f(w,d_0)}{\partial w} \\ \frac{\partial f(w,d_0+\varepsilon)}{\partial d} < \frac{\partial f(\sqrt{\frac{d_0+\varepsilon}{d_0}}w,d_0)}{\partial w} \end{cases} \tag{11}$$

where $\varepsilon > 0$.

In a BNN, if only the change in representation precision is considered, i.e., the decrease from 2^{32} of a full-precision network to 2^1, which significantly reduces the network representational capability and leads to a degradation in task performance, but the backbone of the full-precision network should still be appropriate for the BNN, where the gain relationship between depth and width should remain unchanged,

$$G_b = f_b(w,d) < f(w,d) \tag{12}$$

and $f_b(w,d)$ has similar properties with $f(w,d)$. However, network binarization not only reduces the representation precision of each neuron but also introduces the quantization error and gradient error that accumulate layer by layer along the network depth (Qin et al. 2020; Kim et al. 2020; Ye, Wang, and Zhang 2021). Specifically, during the forward propagation, the quantization error is transmitted from the current layer to the next layer and accumulates along depth,

$$\begin{cases} a_b^{i+1} = B^{i+1}(a_b^i) = B^{i+1}(a_r^i - e^i) \\ E_Q = \sum_{i=1}^{l} |a_r^i - a_b^i| \end{cases} \tag{13}$$

where e^i denotes the residual between the real-valued activation a_r^i and the binary activation a_b^i at the ith layer; similarly, during the backward propagation, the gradient error at the deep layer is also transmitted to the shallow layer and accumulates along depth according to the chain rule. For simplicity, the quantization error and gradient error are fused as the quantization loss, and since both are related to the depth of networks, which can be expressed as:

$$L_Q = E_Q + E_G = \varphi(d) \tag{14}$$

The quantization loss accumulates along depth,

$$L'_Q = \frac{\partial \varphi(d)}{\partial d} > 0 \tag{15}$$

Intuitively, the larger the quantization loss, the more severe the task performance degradation of the model, and thus the relationship between the quantization loss and the performance gains of the model should be as:

$$\begin{cases} G_L = -g(L_Q) = -g(\varphi(d)) = -g_L(d) \\ G'_L = -\frac{\partial g_L(d)}{\partial d} < 0 \end{cases} \tag{16}$$

Therefore, the actual performance gains of a BNN would be:

$$G_b^t = G_b + G_L = f_b(w,d) - g_L(d) \tag{17}$$

From Eq. (9) and (11), we know that given the computational budget $C_0 = \psi_r(w,d_0)$, the optimal configuration of depth/width of the full-precision network is $\mathrm{N}_r[w,d_0]$, and the corresponding computational budget of the BNN is $C_n = \psi_b(w,d_0)$. Whereas for a BNN, if

$$\frac{\partial f_b(w,d_0)}{\partial d} = \frac{\partial f_b(w,d_0)}{\partial w} \tag{18}$$

due to the quantization loss, the difference in performance gains between increasing depth and width at this point is:

$$\begin{aligned} \Delta =& G_b^t(w,d_0+\varepsilon_0) - G_b^t(\sqrt{\tfrac{d_0+\varepsilon_0}{d_0}}w,d_0) \\ =& [f_b(w,d_0+\varepsilon_0) - g_L(d_0+\varepsilon_0)] \\ & -[f_b(\sqrt{\tfrac{d_0+\varepsilon_0}{d_0}}w,d_0) - g_L(d_0)] \\ =& [f_b(w,d_0+\varepsilon_0) - f_b(w,d_0)] \\ & -[f_b(\sqrt{\tfrac{d_0+\varepsilon_0}{d_0}}w,d_0) - f_b(w,d_0)] \\ & -[g_L(d_0+\varepsilon_0) - g_L(d_0)] \end{aligned} \tag{19}$$

Let $\lim(\varepsilon_0 \to 0)$, from Eq. (16) and (18), it is clear that $\Delta < 0$. Thus, the gains from increasing the width is greater than that from increasing the depth, i.e.,

$$\frac{\partial G_b^t(w,d_0)}{\partial d} < \frac{\partial G_b^t(w,d_0)}{\partial w} \tag{20}$$

The equilibrium point of the depth of the BNN is shifted back compared to the full-precision network. Therefore, there exists a new equilibrium point $d = d_n < d_0$ such that:

$$\frac{\partial G_b^t(\sqrt{\frac{d_0}{d_n}}w,d_n)}{\partial d} = \frac{\partial G_b^t(\sqrt{\frac{d_0}{d_n}}w,d_n)}{\partial w} \tag{21}$$

It indicates that for a given computational budget, when the optimal configuration of depth/width for the full-precision network is $\mathrm{N}_r[w,d_0]$, the optimal configuration for the corresponding BNN may be $\mathrm{N}_b[\sqrt{\frac{d_0}{d_n}}w,d_n]$. In short, the quantization loss caused by binarization partially offsets the performance gains from increasing depth, resulting in **BNNs that may favor shallower and wider backbones relative to the deep and narrow backbones of full-precision networks.**

4 Method

In this section, motivated by the previous analysis, we present the construction of fast and accurate binary neural networks based on Depth-Width Reshaping (DWR). First, the naive reshaping strategy is elaborated to directly reshape the depth and width of the existing full-precision network backbones to obtain more suitable backbones for BNNs. Then the reshaping method incorporating the pruning technique is proposed to further optimize the backbones to achieve strong baseline backbones for BNNs.

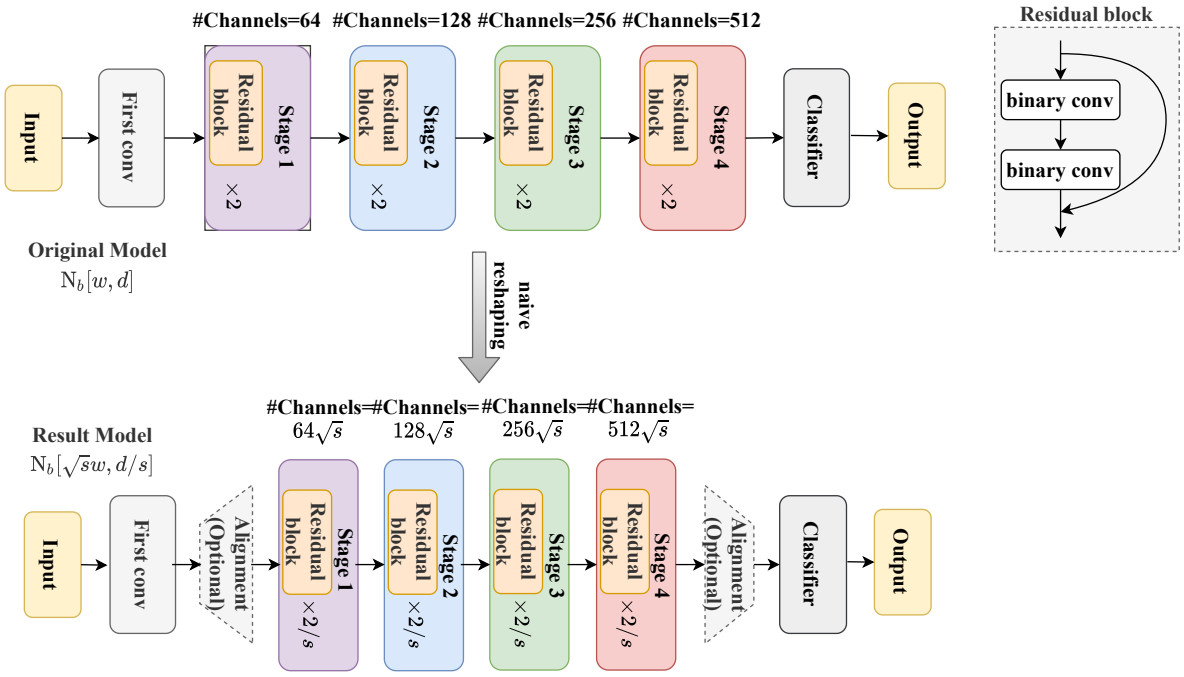

Figure 2: Illustration of the naive reshaping

.

4.1 Naïve Reshaping

Our purpose is to obtain baseline backbones suitable for BNNs by reshaping the depth and width of the existing full-precision network backbones. That is, given a full-precision network $\mathrm{N}_r[w_0, d_0]$, corresponding to a BNN $\mathrm{N}_b[w_0, d_0]$, and a computational budget $\psi_b(w_0, d_0)$, by reshaping the depth and width to obtain $\mathrm{N}_b[w_n, d_n]$ that satisfies:

$$\begin{cases} \max\limits_{w,d}(G_b^t) = G_b^t(w_n, d_n) \geq G_b^t(w_0, d_0) \\ \psi_b(w_n, d_n) \leq \psi_b(w_0, d_0) \end{cases} \tag{22}$$

Theoretically, it is feasible to add/remove any convolutional layer anywhere in a network to adjust the depth, and the number of channels in each layer can be finetuned separately to adjust the width, so it is hard to find $[w_n, d_n]$ exactly. For simplicity and to make the network backbones comparable before and after adjusted, similar to model scaling (Tan and Le 2019), we restrict that all layers must be scaled uniformly with a constant ratio (in practice, the number of layers and channels can only be taken as positive integers, so there is no guarantee that the adjustment is absolutely uniform). For more intuitiveness, the adjustment process of the naive reshaping is illustrated in Figure 2, using ResNet-18 as an example. Since the first and last layers in a BNN are usually kept as real-valued layers, modifying their channels would cause a large impact on the computational cost, so the alignment layer can be considered to keep their channels constant (the alignment layer is optional and can be implemented in a low-cost way using binary convolution, group convolution, etc.).

By restricting the reshaping strategy, given a computational budget, from Table 1, and taking the classification task as an example, Eq. (22) can be simplified as:

$$\max_{s} \ Accuracy(N_b[\sqrt{s}w_0, d_0/s]), \ s > 0 \tag{23}$$

and meanwhile, the value of s with the following constraints:

1) According to section 3.3, a BNN prefers a shallower and wider backbone compared to the full-precision network, i.e., generally $s > 1$;

2) Scaling the depth from d to d/s , the number of the basic blocks (e.g., the residual blocks of ResNet (He et al. 2016) and the convolutional layers between pooling layers of VGGNet (Simonyan and Zisserman 2015)) should be a positive integer.

Therefore, the range of s is limited. Specifically, we have conducted experiments on ResNet-20, ResNet-18 (He et al. 2016), VGG-Small (Zhang et al. 2018), and VGG-11 (Simonyan and Zisserman 2015). For each backbone, the available values of s are listed in Table 2, where the specific reshaping of VGG-Small is shown in section 5 due to its specificity of backbone.

Model	s
ResNet-20	{1.5, 3}
ResNet-18	{2}
VGG-Small	{1}
VGG-11	{2}

Table 2: The range of s in various networks

4.2 Reshaping with Pruning

The naive reshaping is simple in that the backbone would be determined by determining s. And the range of s is very

Model	Scaling		OPs	Parameters	Activations
before	d	w	dw^2r^2	dw^2	dwr^2
after	d/s	$\sqrt{s}w$	dw^2r^2	dw^2	$dwr^2/\sqrt{s}$

Table 3: Network complexity before and after network reshaping

limited, which results in a limited design space for the network. To further improve the backbone and obtain a strong baseline, pruning techniques are incorporated with the naive reshaping. There are many excellent pruning methods, but we mainly focus on the network width/depth reshaping. For simplicity, the clustering pruning (Chang et al. 2022) is employed as an architecture search method straightforwardly. Specifically, the reshaping with pruning is presented in Figure 1. First, the original backbone $\mathrm{N}_b[w, d]$ is reshaped to $\mathrm{N}_b[sw, d/s]$ via the naive reshaping, at which the computational cost is higher than that of the original one; second, pruning methods usually require pre-trained models, and only a few epochs of pre-training are needed for saving the training cost (see section 5 for a specific discussion on pre-training); and then, the computational cost is restricted below that of the original backbone by clustering pruning (Chang et al. 2022) while minimizing the accuracy drops, and the backbone is tuned to $\mathrm{N}_b[\sqrt{s}w*, d/s]$ finally. It is worth noting that the computational cost of the result model is comparable to that of the backbone $\mathrm{N}_b[\sqrt{s}w, d/s]$ obtained using the naive reshaping in section 4.1, i.e., $\psi_b(\sqrt{s}w*, d/s) \approx \psi_b(\sqrt{s}w, d/s) \leq \psi_b(w, d)$, but the pruning to the width scaling is non-uniform and thus distinguished using $w*$.

4.3 Efficiency Analysis

In this work, the backbone reshaping is carried out under the computational complexity constraints of the original model, and the computational cost of the reshaped network does not exceed that of the original one, which means that the proposed method can obtain better accuracy without introducing additional computational cost. In addition, from Table 1, by reshaping the network from $\mathrm{N}_b[w, d]$ to $N_b[\sqrt{s}w, d/s]$, the complexity before and after network reshaping would be as in Table 3. According to section 3.3, usually $s > 1$, the reshaped network has fewer activations. (Liu et al. 2021; Dollár, Singh, and Girshick 2021) show that the model's running time is more correlated with the activations than with the OPs, i.e., the reshaped network has a better speedup. **In short, the reshaped network via DWR is faster and more accurate compared to the original one.**

5 Experiments

5.1 Datasets and Implementation Details

Datasets: To evaluate the performance of DWR, extensive experiments are conducted on two widely used datasets, including CIFAR-10 (Torralba, Fergus, and Freeman 2008) and ImageNet ILSVRC12 (Deng et al. 2009). CIFAR-10 consists of 60,000 32x32 images divided into 10 categories, of which 50,000 are the training set and the remaining 10,000 are the test set. ImageNet is a more challenging and diverse dataset that is divided into 1000 categories and contains about 1.2 million training images and 50,000 test images.

Reshaping Method	Result Model	Bit-width (W/A)	Accuracy (%)
FP	$\mathrm{N}_r[w, d]$	32/32	90.8
Original	$\mathrm{N}_b[w, d]$	1/1	85.2
Original+1	$\mathrm{N}_b[w, d] + 1$	1/1	85.2
Naïve reshaping	$\mathrm{N}_b[\sqrt{1.5}w, d/1.5]$	1/1	85.4
	$\mathrm{N}_b[\sqrt{3}w, d/3]$	1/1	85.8
Reshaping with pruning (DWR)	$\mathrm{N}_b[\sqrt{1.5}w*, d/1.5]$	1/1	85.5
	$\mathrm{N}_b[\sqrt{3}w*, d/3]$	1/1	**86.0**

Table 4: Comparison of naïve reshaping and reshaping with pruning

Reshaping Method	Pre-training Epochs	Bit-width (W/A)	Accuracy (%)
FP	0	32/32	90.8
Original	0	1/1	87.5
DWR	5	1/1	88.0
DWR	10	1/1	88.2
DWR	15	1/1	88.3
DWR	400 (well-trained)	1/1	**88.4**

Table 5: Impact of the pre-training epochs ((Liu et al. 2020b) as the optimization method)

Implementation Details: We implement DWR with PyTorch. Consistent with existing binarization methods (Liu et al. 2020a; Qin et al. 2020; Lin et al. 2020; Martínez et al. 2020), all layers are binarized except for the first and last layers and the downsampling layer. Since the binarization function sign is not differentiable, we use ApproxSign (Liu et al. 2020a) for the gradient approximation. For a fair comparison, the same training scheme is used for the original and DWR backbones. Specifically, on CIFAR-10, VGG-Small, VGG-11, ResNet-20, and ResNet-18 are used as the original backbones, respectively, trained using SGD algorithm with weight decay 1e-4, momentum 0.9, and batch size 128; for ImageNet, ResNet-18 with the additional shortcut is used as the original backbone (Liu et al. 2020a), trained using Adam optimizer with weight decay 1e-5, momentum 0.9, and batch size 512. All models are trained from scratch without using any pre-trained models. For all others, we mostly follow the settings of their original papers if not otherwise specified.

5.2 Ablation study

Section 3.3 argues from a theoretical perspective that BNNs may require relatively shallow and wide backbones compared to full-precision networks. We construct DWR backbones incorporating pruning techniques. To study the effect of pruning and to verify the analysis of section 3.3 from an empirical perspective, we test the performance of reshaping with/without pruning separately. Besides, to evaluate the training cost of pruning, the impact of the number of the pre-training epochs is analyzed. All experiments are conducted on ResNet-20 with the additional shortcut as the original

backbone (Liu et al. 2020a) and trained from scratch.

Reshaping with/without Pruning: Table 4 shows the impact of various backbone reshaping methods on model performance, where "Original+1" indicates that one binary convolutional layer is inserted between the last convolutional layer and the fully connected layer of ResNet-20. As can be seen from the table, the accuracy gains of increasing the depth of the original backbone by one layer are negligible, whereas there are already more significant gains (up to 0.6%) from just using the naive reshaping to make the network shallower and wider, which verifies the analysis in section 3.3 empirically. It is worth noting that scaling down the depth of ResNet-20 by a factor of 3 is more accurate than that of by a factor of 1.5, which further indicates that binarization has a significant impact on the gain relationship between the width and depth of the network. Moreover, compared with naive reshaping, by incorporating the pruning technique, DWR achieves the best accuracy of 86.0%. From Tables 2 and 4, $s = 3$ for ResNet-20 in the subsequent experiments.

Impact of the pre-training epochs: As described in section 4.2, pruning usually requires pre-training the models, and to evaluate the training cost of pruning, ResNet-20 with the optimization method of ReActNet (Liu et al. 2020b) is used as the original backbone. First, an intermediate model is obtained using the naive reshaping, and then pruning is performed after pre-training the intermediate model for various epochs to obtain the final backbones. Table 5 shows the impact of the pre-training epochs on the accuracy of the final models. It can be seen that the accuracy of the DWR backbone is better than that of the original one even with very few epochs of pre-training (e.g., 5 epochs). Moreover, when $\#\text{epochs} \geq 15$, the accuracy is comparable to that of the final model obtained by complete training. Therefore, the construction of DWR backbones can be performed with just a few epochs of pre-training, which has a negligible impact on the overall training cost of the model.

5.3 Comparison with SOTA methods

We further perform comparison experiments. The original backbones are compared with our DWR backbones on CIFAR-10 and ImageNet, with/without using additional optimization methods, respectively, to comprehensively evaluate the performance of the DWR backbones (optimization methods including BinaryNet (Hubara et al. 2016), XNOR-Net (Rastegari et al. 2016), Bi-Real Net (Liu et al. 2020a), IR-Net (Qin et al. 2020), BBG (Shen et al. 2020), RBNN (Lin et al. 2020), Real-to-Bin (Martínez et al. 2020), ReActNet (Liu et al. 2020b), SD-BNN (Xue et al. 2022), BinaryDuo (Kim et al. 2020), DGRL (Ye, Wang, and Zhang 2021), XNOR-Net++ (Bulat and Tzimiropoulos 2019), Equal Bits (Li, Pintea, and van Gemert 2022), ANE (Zhang et al. 2021), and BNN-DL (Ding et al. 2019)).

CIFAR-10: On the CIFAR-10 dataset, we conduct experiments with VGG-11, ResNet-20, and ResNet-18 as the original backbones, respectively. As can be seen from Table 6, our DWR backbones outperform the original ones either with or without using additional optimization methods. In particular, the DWR backbone of VGG-11 outperforms the original one by up to 1.3% in terms of absolute accuracy. In addition, VGG-Small is investigated as well. Intuitively, compared with the backbones of VGG-11, ResNet-20, and ResNet-18, the VGG-Small backbone is shallower and wider, and it may not be suitable to make it further shallower and wider, so different from the other backbones, the reshaping method and results are shown in Table 7. The "Original-1" and "Original+1" represent removing/appending the last convolutional layer of the original backbone, respectively, and then reshaping with pruning is used to ensure the computational cost under the constraints. From table 7, it can be seen that further reducing the depth of the shallow enough network can harm the accuracy of the model, whereas deepening it appropriately can improve its accuracy, which also confirms the analysis in section 3.3 from another perspective.

ImageNet: We further investigate the performance of DWR on ImageNet, Table 8 shows the results. As can be

Original Model	Optimization Method	Reshaping Method	Bit-width (W/A)	Accuracy (%)
VGG-11	FP	Original	32/32	89.3
	None	Original	1/1	83.6
		DWR	1/1	**84.9**
ResNet-20	FP	Original	32/32	90.8
	None	Original	1/1	85.2
		DWR	1/1	**86.0**
	IR-Net	Original	1/1	86.5
	SD-BNN	Original	1/1	86.9
	ReActNet	Original	1/1	87.5
		DWR	1/1	**88.4**
ResNet-18	FP	Original	32/32	93.0
	None	Original	1/1	90.2
		DWR	1/1	**91.3**
	BNN-DL	Original	1/1	90.5
	ANE	Original	1/1	91.0
	IR-Net	Original	1/1	91.5
	RBNN	Original	1/1	92.2
	SD-BNN	Original	1/1	92.5
		DWR	1/1	**92.6**

Table 6: Comparison between original and DWR backbones on CIFAR-10

Optimization Method	Reshaping Method	Bit-width (W/A)	Accuracy (%)
FP	Original	32/32	91.7
None	Original	1/1	90.2
	Original-1	1/1	90.1
	Original+1 (DWR)	1/1	**90.6**
XNOR-Net	Original	1/1	89.8
BinaryNet	Original	1/1	89.9
BNN-DL	Original	1/1	90.0
IR-Net	Original	1/1	90.4
BinaryDuo	Original	1/1	90.4
SD-BNN	Original	1/1	90.8
	DWR	1/1	**91.3**

Table 7: Effect of reshaping the depth/width of VGG-Small

Optimization Method	Reshaping Method	Bit-width (W/A)	OPs ($\times 10^8$)	Top-1 (%)	Top-5 (%)
FP	Original	32/32	18.3	69.3	89.2
BinaryNet	Original	1/1	1.58	42.2	-
XNOR-Net	Original	1/1	1.60	51.2	73.2
Bi-Real Net	Original	1/1	1.75	56.4	79.5
XNOR-Net++	Original	1/1	-	57.1	79.9
IR-Net	Original	1/1	1.75	58.1	80.0
BBG	Original	1/1	-	59.4	-
Equal Bits	Original	1/1	-	60.4	82.9
BinaryDuo	Original	1/1	-	60.9	82.6
Real-to-Bin	Original†	1/1	1.82	65.4	86.2
DGRL	Original†	1/1	2.08	65.7	-
None	Original	1/1	1.74	55.5	78.3
	DWR	1/1	1.73	**56.9**	**78.9**
ReActNet	Original	1/1	1.81	61.8	83.2
	DWR	1/1	1.81	62.5	83.4
	Original†	1/1	1.81	65.5	-
	DWR†	1/1	1.81	**67.2**	**87.2**

Table 8: Comparison between original and DWR backbones on ImageNet (†indicates training using the 2-step strategy (Martínez et al. 2020; Liu et al. 2020b))

seen from the table, the DWR backbone still has a significant superiority over the original one on the large-scale dataset. Without using additional optimization methods, the DWR backbone improves the absolute Top-1 accuracy by 1.4% over the original backbone; while with the DWR backbone, the existing method achieves a new SOTA (from 65.5% to 67.2%). In particular, the reconstruction of the original backbone by the proposed method is under the constraint of computational cost, and thus the computational cost is comparable between before and after reconstruction (as shown in the OPs column of Table 8); meanwhile, as described in section 4.3, the DWR backbone has fewer activations, so the better model acceleration. In fact, we use 2 Nvidia RTX Titan GPUs with the same training scheme for both the original and DWR backbones without additional optimization methods, the original one takes about 4 days, whereas the DWR backbone only needs 3.5 days, which indicates $\sim$12.5% speedups.

6 Conclusion

In this paper, we start from the causes of performance degradation of BNNs, and analyze that for a given computational budget, the backbones of BNNs may need to be shallower and wider than that of full-precision networks. Based on this, DWR is proposed to first uniformly scale the width and depth of the existing full-precision network backbones, and then further optimize them via the pruning technique. Compared with the original backbones, the DWR backbones have fewer activations, better model acceleration, and higher accuracy, with comparable computational cost. Experiments with various backbones on CIFAR-10 and ImageNet demonstrate the analytical result and the effectiveness of the proposed method. Crucially, DWR provides a novel insight into the backbone design of BNNs.

Acknowledgments

This work was supported in part by the Anhui Provincial Key Research and Development Program under Grant 202004a05020040, in part by the National Key Research and Development Program under Grant 2018YFC0604404, in part by Intelligent Network and New Energy Vehicle Special Project of Intelligent Manufacturing Institute of HFUT under Grant IMIWL2019003, and in part by Fundamental Research Funds for the Central Universities under Grant PA2021GDGP0061.

References

Bengio, Y.; Léonard, N.; and Courville, A. C. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. arXiv:1308.3432.

Bulat, A.; and Tzimiropoulos, G. 2019. XNOR-Net++: Improved binary neural networks. In *BMVC*, 62.

Chang, J.; Lu, Y.; Xue, P.; Xu, Y.; and Wei, Z. 2022. Automatic channel pruning via clustering and swarm intelligence optimization for CNN. *Applied Intelligence*, 1–21.

Delalleau, O.; and Bengio, Y. 2011. Shallow vs. Deep Sum-Product Networks. In *NIPS*, 666–674.

Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. In *CVPR*, 248–255.

Ding, R.; Chin, T.; Liu, Z.; and Marculescu, D. 2019. Regularizing Activation Distribution for Training Binarized Deep Networks. In *CVPR*, 11408–11417.

Dollár, P.; Singh, M.; and Girshick, R. B. 2021. Fast and Accurate Model Scaling. In *CVPR*, 924–932.

Eldan, R.; and Shamir, O. 2016. The Power of Depth for Feedforward Neural Networks. In *COLT*, volume 49, 907–940.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

Hu, J.; Shen, L.; Albanie, S.; Sun, G.; and Wu, E. 2020. Squeeze-and-Excitation Networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(8): 2011–2023.

Huang, Y.; Cheng, Y.; Bapna, A.; Firat, O.; Chen, D.; Chen, M. X.; Lee, H.; Ngiam, J.; Le, Q. V.; Wu, Y.; and Chen, Z. 2019. GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism. In *NIPS*, 103–112.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized Neural Networks. In *NIPS*, 4107–4115.

Kim, H.; Kim, K.; Kim, J.; and Kim, J. 2020. BinaryDuo: Reducing Gradient Mismatch in Binary Activation Network by Coupling Binary Activations. In *ICLR*.

Kim, H.; Park, J.; Lee, C.; and Kim, J. 2021. Improving Accuracy of Binary Neural Networks Using Unbalanced Activation Distribution. In *CVPR*, 7862–7871.

Li, Y.; Pintea, S. L.; and van Gemert, J. C. 2022. Equal Bits: Enforcing Equally Distributed Binary Network Weights. In *AAAI*.

Lin, M.; Ji, R.; Xu, Z.; Zhang, B.; Wang, Y.; Wu, Y.; Huang, F.; and Lin, C. 2020. Rotated Binary Neural Network. In *NIPS*.

Liu, C.; Ding, W.; Chen, P.; Zhuang, B.; Wang, Y.; Zhao, Y.; Zhang, B.; and Han, Y. 2022. RB-Net: Training Highly Accurate and Efficient Binary Neural Networks with Reshaped Point-wise Convolution and Balanced Activation. *IEEE Transactions on Circuits and Systems for Video Technology*, 1–1.

Liu, G.; Zhang, K.; and Lv, M. 2022. SOKS: Automatic Searching of the Optimal Kernel Shapes for Stripe-Wise Network Pruning. *IEEE Transactions on Neural Networks and Learning Systems*, 1–13.

Liu, L.; Zhang, S.; Kuang, Z.; Zhou, A.; Xue, J.; Wang, X.; Chen, Y.; Yang, W.; Liao, Q.; and Zhang, W. 2021. Group Fisher Pruning for Practical Network Compression. In *ICML*, volume 139, 7021–7032.

Liu, Z.; Luo, W.; Wu, B.; Yang, X.; Liu, W.; and Cheng, K. 2020a. Bi-Real Net: Binarizing Deep Network Towards Real-Network Performance. *Int. J. Comput. Vis.*, 128(1): 202–219.

Liu, Z.; Shen, Z.; Savvides, M.; and Cheng, K. 2020b. ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions. In *ECCV*, 143–159.

Lu, Z.; Pu, H.; Wang, F.; Hu, Z.; and Wang, L. 2017. The Expressive Power of Neural Networks: A View from the Width. In *NIPS*, 6231–6239.

Martínez, B.; Yang, J.; Bulat, A.; and Tzimiropoulos, G. 2020. Training binary neural networks with real-to-binary convolutions. In *ICLR*.

Pascanu, R.; Montufar, G.; and Bengio, Y. 2013. On the number of response regions of deep feed forward networks with piece-wise linear activations. arXiv:1312.6098.

Qin, H.; Gong, R.; Liu, X.; Shen, M.; Wei, Z.; Yu, F.; and Song, J. 2020. Forward and Backward Information Retention for Accurate Binary Neural Networks. In *CVPR*, 2247–2256.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In *ECCV*, 525–542.

Shen, M.; Liu, X.; Gong, R.; and Han, K. 2020. Balanced Binary Neural Networks with Gated Residual. In *ICASSP*, 4197–4201.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Tan, M.; and Le, Q. V. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *ICML*, volume 97, 6105–6114.

Torralba, A.; Fergus, R.; and Freeman, W. T. 2008. 80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11): 1958–1970.

Tung, F.; and Mori, G. 2020. Deep Neural Network Compression by In-Parallel Pruning-Quantization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(3): 568–579.

Xue, P.; Lu, Y.; Chang, J.; Wei, X.; and Wei, Z. 2022. Self-distribution binary neural networks. *Applied Intelligence*, 1–13.

Ye, J.; Wang, J.; and Zhang, S. 2021. Distillation-Guided Residual Learning for Binary Convolutional Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 1–13.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In *BMVC*.

Zhang, D.; Yang, J.; Ye, D.; and Hua, G. 2018. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks. In *ECCV*, 373–390.

Zhang, S.; Ge, F.; Ding, R.; Liu, H.; and Zhou, X. 2021. Learning to Binarize Convolutional Neural Networks with Adaptive Neural Encoder. In *IJCNN*, 1–8.