One Weight Bitwidth to Rule Them All

Ting-Wu Chin[1], Pierce I-Jen Chuang[2], Vikas Chandra[2], and Diana Marculescu[1,3]

[1] Eletrical and Computer Engineering, Carnegie Mellon University
{tingwuc,dianam}@cmu.edu
[2] Facebook Inc.
{pichuang,vchandra}@fb.com
[3] Eletrical and Computer Engineering, The University of Texas at Austin
{dianam}@utexas.edu

Abstract. Weight quantization for deep ConvNets has shown promising results for applications such as image classification and semantic segmentation and is especially important for applications where memory storage is limited. However, when aiming for quantization without accuracy degradation, different tasks may end up with different bitwidths. This creates complexity for software and hardware support and the complexity accumulates when one considers mixed-precision quantization, in which case each layer's weights use a different bitwidth. Our key insight is that optimizing for the least bitwidth subject to no accuracy degradation is not necessarily an optimal strategy. This is because one cannot decide optimality between two bitwidths if one has smaller model size while the other has better accuracy. In this work, we take the first step to understand if some weight bitwidth is better than others by aligning all to the same model size using a width-multiplier. Under this setting, somewhat surprisingly, we show that using a single bitwidth for the whole network can achieve better accuracy compared to mixed-precision quantization targeting zero accuracy degradation when both have the same model size. In particular, our results suggest that when the number of channels becomes a target hyperparameter, a single weight bitwidth throughout the network shows superior results for model compression.

Keywords: Model Compression, Deep Learning Architectures, Quantization, ConvNets, Image Classification

1 Introduction

Recent success of ConvNets in computer vision applications such as image classification and semantic segmentation has fueled many important applications in storage-constrained devices, *e.g.*, virtual reality headsets, drones, and IoT devices. As a result, improving the parameter-efficiency (the top-1 accuracy to the parameter counts ratio) of ConvNets while maintaining their attractive features (*e.g.*, accuracy for a task) has gained tremendous research momentum recently.

Among the efforts of improving ConvNets' efficiency, weight quantization was shown to be an effective technique [39,38,10,5]. The majority of research efforts in

quantization has targeted quantization algorithms for finding the lowest possible weight bitwidth without compromising the figure-of-merit (*i.e.*, accuracy). Mixed-precision quantization methods, which allow different bitwidths to be selected for different layers in the network, have recently been proposed to further compress deep ConvNets [29,31,6]. Nevertheless, having different bitwidths for different layers greatly increases the neural network implementation complexity from both hardware and software perspectives. For example, hardware and software implementations optimized for executing an 8 bits convolution are sub-optimal for executing a 4 bits convolution, and vice versa.

To minimize the efforts of hardware and software support, it is natural to wonder: "Is some weight bitwidth better than others?" However, this is an ill-posed problem as one cannot decide optimality between two bitwidths if one has smaller model size while the other has better accuracy. This work takes a first step towards understanding if some bitwidth is better than other bitwidths under a given model size constraint. Given the multi-objective nature of the problem, we need to align different bitwidths to the same model size to further decide the optimality for the bitwidth selection. To realize model size alignment for different bitwidths, we use the width-multiplier[4] [11] as a tool to compare the performance of different weight bitwidths under *the same model size*.

With this setting, we find that there exists some weight bitwidth that consistently outperforms others across different model sizes when both are considered under a given model size constraint. This suggests that one can decide the optimal bitwidth for small model sizes to save computing cost and the result generalizes to large model sizes[5]. Additionally, we show that the optimal bitwidth of a convolutional layer negatively correlates to the convolutional kernel fan-in. As an example, depth-wise convolutional layers turn to have optimal bitwidth values that are higher than that of all-to-all convolutions. We further provide a theoretical reasoning for this phenomenon. These findings suggest that architectures such as VGG and ResNets are more parameter-efficient when they are wide and use binarized weights. On the other hand, networks such as MobileNets [11] might require different weight bitwidths for all-to-all convolutions and depth-wise convolutions. Somewhat surprisingly, we find that on ImageNet, under a given model size constraint, a single bitwidth for both ResNet-50 and MobileNetV2 can outperform mixed-precision quantization using reinforcement learning [29] that targets minimum total bitwidth without accuracy degradation. This suggests that searching for the minimum bitwidth configuration that does not introduce accuracy degradation without considering other hyperparameters affecting model size is a sub-optimal strategy. Our results suggest that when the number of channels becomes one of the hyperparameters under consideration, a single weight bitwidth throughout the network shows great potential for model compression.

[4] Width-multiplier grows or shrinks the number of channels across the layers with identical proportion for a certain network, *e.g.*, grow the number of channels for all the layers by $2\times$.

[5] Note that we use width-multiplier to scale model across different sizes.

In summary, we systematically analyze the model size and accuracy trade-off considering both weight bitwidths and the number of channels for various modern networks architectures (variants of ResNet, VGG, and MobileNet) and datasets (CIFAR and ImageNet) and have the following contributions:

- We empirically show that when allowing the network width to vary, lower weight bitwidths outperform higher ones in a Pareto sense (accuracy *vs.* model size) for networks with standard convolutions. This suggests that for such ConvNets, further research on wide binary weight networks is likely to identify better network configurations which will require further hardware/software platform support.
- We empirically show that the optimal bitwidth of a convolutional layer negatively correlates to the convolutional kernel fan-in and provide theoretical reasoning for such a phenomenon. This suggests that one could potential categorize ConvNets based on the convolutional kernel fan-in when designing the corresponding bitwidth support from both software and hardware.
- We empirically show that one can achieve a more accurate model (under a given model size) by using a single bitwidth when compared to mixed-precision quantization that uses deep reinforcement learning to search for layer-wise weight precision values. Moreover, the results are validated on a large-scale dataset, *i.e.*, ImageNet.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 discusses the methodology used to discover our findings. Section 4 discusses our experiments for all our findings. In particular, Section 4.2 shows that some bitwidth can outperform others consistently across model sizes when both are compared under the same model size constraint using width-multipliers. Section 4.3 discusses how fan-in channel count per convolutional kernel affects the resilience of quantization for convolution layers, which further affects the optimal bitwidth for a convolution layer. Section 4.4 scales up our experiments to ImageNet and demonstrates that a single weight bitwidth manages to outperform mixed-precision quantization given the same model size. Section 5 concludes the paper.

2 Related Work

Several techniques for improving the efficiency of ConvNets have been recently proposed. For instance, pruning removes the redundant connections of a trained neural network [41,33,28,16,7,3,34], neural architecture search (NAS) tunes the number of channels, size of kernels, and depth of a network [27,26,2,25], and convolution operations can be made more efficient via depth-wise convolutions [11], group convolutions [12,37], and shift-based convolutions [9,30]. In addition to the aforementioned techniques, network quantization introduces an opportunity for hardware-software co-design to achieve better efficiency for ConvNets.

There are in general two directions for weight quantization in prior literature, post-training quantization [20,18,36,23] and quantization-aware training [21,40,13,14,35,10,4]. The former assumes training data is not available when

quantization is applied. While being fast and training-data-free, its performance is worse compared to quantization-aware training. In contrast, our work falls under the category of quantization-aware training.

In quantization-aware training, [21] introduces binary neural networks, which lead to significant efficiency gain by replacing multiplications with XNOR operations at the expense of significant accuracy degradation. Later, [40] propose ternary quantization and [39,13] bridge the gap between floating-point and binarized neural networks by introducing fixed-point quantization. Building upon prior art, the vast majority of existing work focuses on reducing the accuracy degradation by improving the training strategy [38,32,17,5] and developing better quantization schemes [14,29,35]. However, prior art has studied quantization by fixing the network architecture, which may lead to a sub-optimal bitwidth selection in terms of parameter-efficiency (the top-1 accuracy to the parameter counts ratio).

Related to our work, [19] have also considered the impact of channel count in quantization. In contrast, our work has the following novel features. First, we find that in ConvNets with standard convolutions, *a lower bitwidth outperforms higher ones under a given model size constraint*. Second, we find that the Pareto optimal bitwidth negatively correlates to the convolutional kernel fan-in and we provide theoretical insights for it. Last, we show that a single weight bitwidth can outperform *mixed-precision* quantization on ImageNet for ResNet50 and MobileNetV2.

3 Methodology

In this work, we are interested in comparing different bitwidths under a given model size. To do so, we make use of the width-multiplier to scale the models. To be precise in the following discussion, we define an ordering relation across bitwidths as follows:

Definition 1 (bitwidth ordering). *We say bitwidth A is better than bitwidth B for a network family $\mathcal{F}$, if,*

$$Acc(N(A, s)) > Acc(N(B, s)) \quad \forall s,$$

where $Acc(\cdot)$ evaluates the validation accuracy of a network, $N(A, s)$ produces a network in $\mathcal{F}$ that has bitwidth A and model size of s by using width-multiplier.

With Definition 1, we can now compare weight bitwidths for their parameter-efficiency.

3.1 Quantization

This work focuses on weight quantization and we use a straight-through estimator [1] to conduct quantization-aware training. Specifically, for bitwidth values

larger than 2 bit ($b > 2$), we use the following quantization function for weights during the forward pass:

$$Q(\boldsymbol{W}_{i,:}) = \lfloor \frac{clamp(\boldsymbol{W}_{i,:}, -a_i, a_i)}{r_i} \rceil \times r_i, \quad r_i = \frac{a_i}{2^{b-1}-1} \tag{1}$$

where

$$clamp(w, min, max) = \begin{cases} w, & \text{if } min \leq w \leq max \\ min, & \text{if } w < min \\ max & \text{if } w > max \end{cases}$$

and $\lfloor \cdot \rceil$ denotes the round-to-nearest-neighbor function, $\boldsymbol{W} \in \mathbb{R}^{C_{out} \times d}$, $d = C_{in} K_w K_h$ denotes the real-value weights for the i^{th} output filter of a convolutional layer that has C_{in} channels and $K_w \times K_h$ kernel size. $\boldsymbol{a} \in \mathbb{R}^{C_{out}}$ denotes the vector of clipping factors which are selected to minimize $\|Q(\boldsymbol{W}_{i,:}) - \boldsymbol{W}_{i,:}\|_2^2$ by assuming $\boldsymbol{W}_{i,:} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. More details about the determination of a_i is in Appendix A.

For special cases such as 2 bits and 1 bit, we use schemes proposed in prior literature. Specifically, let us first define:

$$|\bar{\boldsymbol{W}}_{i,:}| = \frac{1}{d} \sum_{j=1}^{d} |\boldsymbol{W}_{i,j}|. \tag{2}$$

For 2 bit, we follow trained ternary networks [40] and define the quantization function as follows:

$$\begin{aligned} Q(\boldsymbol{W}_{i,:}) &= (sign(\boldsymbol{W}_{i,:}) \odot \boldsymbol{M}_{i,j}) \times \left(|\bar{\boldsymbol{W}}_{i,:}|\right) \\ \boldsymbol{M}_{i,j} &= \begin{cases} 0, & \boldsymbol{W}_{i,j} < 0.7|\bar{\boldsymbol{W}}_{i,:}|. \\ 1, & otherwise. \end{cases} \end{aligned} \tag{3}$$

For 1 bit, we follow DoReFaNets [39] and define the quantization function as follows:

$$Q(\boldsymbol{W}_{i,:}) = sign(\boldsymbol{W}_{i,:}) \times \left(|\bar{\boldsymbol{W}}_{i,:}|\right). \tag{4}$$

For the backward pass for all the bitwidths, we use a straight-through estimator as in prior literature to make the training differentiable. That is,

$$\frac{\partial Q(\boldsymbol{W}_{i,:})}{\partial \boldsymbol{W}_{i,:}} = \boldsymbol{I}. \tag{5}$$

In the sequel, we quantize the *first and last layers to 8 bits*. They are fixed throughout the experiments. We note that it is a common practice to leave the first and the last layer *un-quantized* [39], however, we find that using 8 bits can achieve comparable results to the floating-point baselines.

As for activation, we use the technique proposed in [13] and use 4 bits for CIFAR-100 and 8 bits for ImageNet experiments. The activation bitwidths are chosen such that the quantized network has comparable accuracy to the floating-point baselines.

3.2 Model size

The size of the model (C_{size}) is defined as:

$$\mathcal{C}_{size} = \sum_{i=1}^{O} b(i)C_{in}(i)K_w(i)K_h(i) \tag{6}$$

where O denotes the total number of filters, $b(i)$ is the bitwidth for filter i, $C_{in}(i)$ denotes the number of channels for filter i, and $K_w(i)$ and $K_h(i)$ are the kernel height and width for filter i.

4 Experiments

We conduct all our experiments on image classification datasets including CIFAR-100 [15] and ImageNet. All experiments are trained from scratch to ensure different weight bitwidths are trained equally long. While we do not start from a pre-trained model, we note that our baseline fixed-point models (*i.e.*, 4 bits for CIFAR and 8 bits for ImageNet) have accuracy comparable to their floating-point counterparts. For all the experiments on CIFAR, we run the experiments three times and report the mean and standard deviation.

4.1 Training hyper-parameters

For CIFAR, we use a learning rate of 0.05, cosine learning rate decay, linear learning rate warmup (from 0 to 0.05) with 5 epochs, batch size of 128, total training epoch of 300, weight decay of $5e^{-4}$, SGD optimizer with Nesterov acceleration and 0.9 momentum.

For ImageNet, we have identical hyper-parameters as CIFAR except for the following hyper-parameters batch size of 256, 120 total epochs for MobileNetV2 and 90 for ResNets, weight decay $4e^{-5}$, and 0.1 label smoothing.

4.2 bitwidth comparisons

In this subsection, we are primarily interested in the following question:

When taking network width into account, does one bitwidth consistently outperform others across model sizes?

To our best knowledge, this is an open question and we take a first step to answer this question empirically. If the answer is affirmative, it may be helpful to focus the software/hardware support on the better bitwidth when it comes to parameter-efficiency. We consider three kinds of commonly adopted ConvNets, namely, ResNets with basic block [8], VGG [24], and MobileNetV2 [22]. These networks differ in the convolution operations, connections, and filter counts. For ResNets, we explored networks from 20 to 56 layers in six layer increments. For VGG, we investigate the case of eleven layers. Additionally, we also study MobileNetV2, which is a mobile-friendly network. We note that we modify the

stride count in of the original MobileNetV2 to match the number of strides of ResNet for CIFAR. The architectures that we introduce for the controlled experiments are discussed in detail in Appendix B.

For CIFAR-100, we only study weight bitwidths below 4 since it achieves performance comparable to its floating-point counterpart. Specifically, we consider 4 bits, 2 bits, and 1 bit weights. To compare different weight bitwidths using Definition 1, we use the width-multiplier to align the model size among them. For example, one can make a 1-bit ConvNet twice as wide to match the model size of a 4-bit ConvNet [6]. For each of the networks we study, we sweep the width-multiplier to consider points at multiple model sizes. Specifically, for ResNets, we investigate seven depths, four model sizes for each depth, and three bitwidths, which results in $7 \times 4 \times 3 \times 3$ experiments. For both VGG11 and MobileNetV2, we consider eight model sizes and three bitwidths, which results in $2 \times 8 \times 3 \times 3$ experiments.

As shown in Fig. 1, across the three types of networks we study, there exists some bitwidth that is better than others. That is, the answer to the question we raised earlier in this subsection is affirmative. For ResNets and VGG, this value is 1 bit. In contrast, for MobileNetV2, it is 4 bits. The results for ResNets and VGG are particularly interesting since lower weight bitwidths are better than higher ones. In other words, binary weights in these cases can achieve the best accuracy and model size trade-off. On the other hand, MobileNetV2 exhibits a different trend where higher bitwidths are better than lower bitwidths up to 4 bits[7].

4.3 ConvNet architectures and quantization

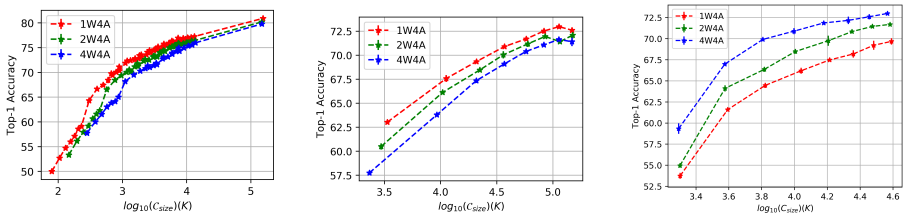
While there exists an ordering among different bitwidths as shown in Fig. 1, it is not clear what determines the optimal weight bitwidth. To further uncover the relationship between ConvNet's architectural parameters and its optimal weight bitwidth, we ask the following questions.

What architectural components determine the MobileNetV2 optimal weight bitwidth of 4 bits as opposed to 1 bit?

As it can be observed in Fig. 1, MobileNetV2 is a special case where the higher bitwidth is better than lower ones. When comparing MobileNetV2 to the other two networks, there are many differences, including how convolutions are connected, how many convolutional layers are there, how many filters in each of them, and how many channels for each convolution. To narrow down which of these aspects result in the reversed trend compared to the trend exhibits in ResNets and VGG, we first consider the inverted residual blocks, *i.e.*, the basic component in MobileNetV2. To do so, we replace all basic blocks (two

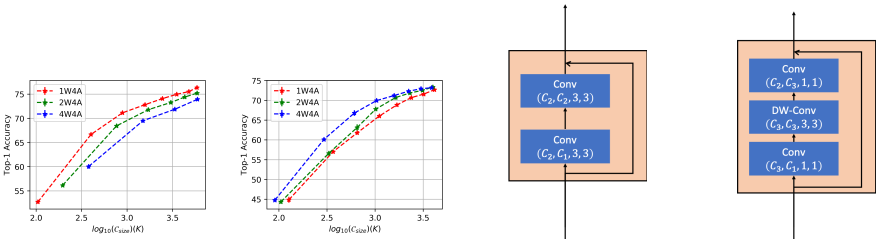
[6] Increase the width of a layer increases the number of output filters for that layer as well as the number of channels for the subsequent layer. Thus, number of parameters and number of operations grow approximately quadratically with the width-multiplier.

[7] However, not higher than 4 bits since the 4-bit model has accuracy comparable to the floating-point model.



(a) ResNets (20 to 56 layers in increments of 6) (b) VGG11 (c) MobileNetV2

Fig. 1: Some bitwidth is consistently better than other bitwidths across model sizes. $\mathcal{C}_{size}$ denotes model size. xWyA denotes x-bit weight quantization and y-bit activation quantization. The experiments are done on the CIFAR-100 dataset. For each network, we sweep the width-multiplier to cover points at multiple model sizes. For each dot, we plot the mean and standard deviation of three random seeds. The standard deviation might not be visible due to little variances.



(a) ResNet26 (b) Inv-ResNet26 (c) Basic block (d) Inverted residual block

Fig. 2: The optimal bitwidth for ResNet26 changes from 1 bit (a) to 4 bit (b) when the building blocks change from basic blocks (c) to inverted residual blocks (d). $\mathcal{C}_{size}$ in (a) and (b) denotes model size. (C_{out}, C_{in}, K, K) in (c) and (d) indicate output channel count, input channel count, kernel width, and kernel height of a convolution.

consecutive convolutions) of ResNet26 with the inverted residual blocks as shown in Fig. 2c and 2d. We refer to this new network as Inv-ResNet26. As shown in Fig. 2a and 2b, the optimal bitwidth shifts from 1 bit to 4 bit once the basic blocks are replaced with inverted residual blocks. Thus, we can infer that the inverted residual block itself or its components are responsible for such a reversed trend.

Since an inverted residual block is composed of a point-wise convolution and a depth-wise separable convolution, we further consider the case of depth-wise separable convolution (DWSConv). To identify whether DWSConv can cause the inverted trend, we use VGG11 as a starting point and gradually replace each of the convolutions with DWSConv. We note that doing so results in architectures that

gradually resemble MobileNetV1 [11]. Specifically, we introduce three variants of VGG11 that have an increasing number of convolutions replaced by DWSConvs. Starting with the second layer, *variant A* has one layer replaced by DWSConv, *variant B* has four layers replaced by DWSConvs, and *variant C* has all of the layers except for the first layer replaced by DWSConvs (the architectures are detailed in Appendix B).

As shown in Fig. 4, as the number of DWSConv increases (from variant A to variant C), the optimal bitwidth shifts from 1 bit to 4 bits, which implies that depth-wise separable convolutions or the layers within it are affecting the optimal bitwidth. To identify which of the layers of the DWSConv (*i.e.*, the depth-wise convolution or the point-wise convolution) has more impact on the optimal bitwidth, we keep the bitwidth of depth-wise convolutions fixed at 4 bits and quantize other layers. As shown in Fig. 4d, the optimal curve shifts from 4 bits being the best back to 1 bit, with a similarly performing 2 bits. Thus, depth-wise convolutions appear to directly affect the optimal bitwidth trends.

Is depth-wise convolution less resilient to quantization or less sensitive to channel increase?

After identifying that depth-wise convolutions have a different characteristic in optimal bitwidth compared to standard all-to-all convolutions, we are interested in understanding the reason behind this. In our setup, the process to obtain a lower bitwidth network that has the same model size as a higher bitwidth network can be broken down into two steps: (1) quantize a network to lower bitwidth and (2) grow the network with width-multiplier to compensate for the reduced model size. As a result, the fact that depth-wise convolution has higher weight bitwidth better than lower weight bitwidth might potentially be due to the large accuracy degradation introduced by quantization or the small accuracy improvements from the use of more channels.

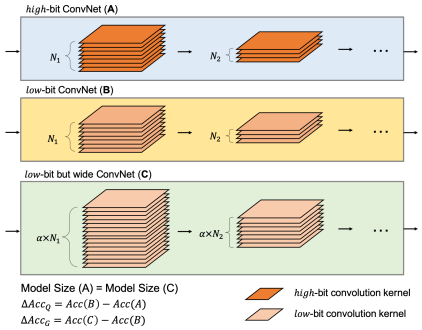
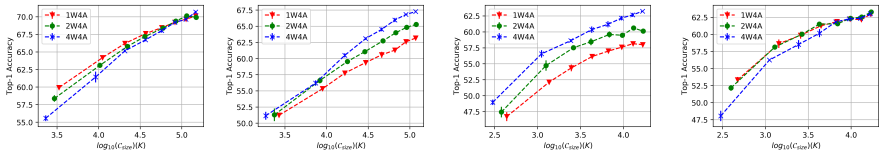


Fig. 3: Visualization of our accuracy decomposition, which is used for analyzing depth-wise convolutions.

To further diagnose the cause, we decompose the accuracy difference between a lower bitwidth but wider network and a higher bitwidth but narrower network into accuracy differences incurred in the aforementioned two steps as shown in Fig. 3. Specifically, let ΔAcc_Q denote the accuracy difference incurred by quantizing a network and let ΔAcc_G denote the accuracy difference incurred by increasing the channel count of the quantized network.

We analyze ΔAcc_G and ΔAcc_Q for networks with and without quantizing depth-wise convolutions, *i.e.*, Fig. 4c and Fig. 4d. In other words, we would like to understand how depth-wise convolutions affect ΔAcc_G and ΔAcc_Q. On



(a) Variant A (b) Variant B (c) Variant C (d) Variant C without quantizing depth-wise convolutions

Fig. 4: The optimal bitwidth for VGG shifts from 1 bit to 4 bit as more convolutions are replaced with depth-wise separable convolutions (DWSConv), *i.e.*, from (a) to (c). Variant A, B, and C have 30%, 60%, and 90% of the convolution layers replaced with DWSConv, respectively. As shown in (d), the optimal bitwidth changes back to 1 bit if we only quantize point-wise convolution but not depth-wise convolutions.

one hand, ΔAcc_Q is evaluated by comparing the accuracy of the 4-bit model and the corresponding 1-bit model. On the other hand, ΔAcc_G is measured by comparing the accuracy of the 1-bit model and its 2× grown counterpart. As shown in Table 1, when quantizing depth-wise convolutions, ΔAcc_Q becomes more negative such that $\Delta Acc_Q + \Delta Acc_G < 0$. This implies that the main reason for the optimal bitwidth change is that quantizing depth-wise convolutions introduce more accuracy degradation than it can be recovered by increasing the channel count when going below 4 bits compared to all-to-all convolutions. We note that it is expected that quantizing the depth-wise convolutions would incur smaller ΔAcc_Q compared to their no-quantization baseline because we essentially quantized more layers. However, depth-wise convolutions only account for 2% of the model size but incur on average near 4× more accuracy degradation when quantized.

We would like to point out that Sheng *et al.* [23] also find that quantizing depth-wise separable convolutions incurs large accuracy degradation. However, their results are based on post-training layer-wise quantization. As mentioned in their work [23], the quantization challenges in their setting could be resolved by quantization-aware training, which is the scheme considered in this work. Hence, our observation is novel and interesting.

Why is depth-wise convolution less resilient to quantization?

Having uncovered that depth-wise convolutions introduce large accuracy degradation when weights are quantized below 4 bits, in this section, we investigate depth-wise convolutions from a quantization perspective. When comparing depth-wise convolutions and all-to-all convolutions in the context of quantization, they differ in the number of elements to be quantized, *i.e.*, $C_{in} = 1$ for depth-wise convolutions and $C_{in} >> 1$ for all-to-all convolutions.

Table 1: Quantizing depth-wise convolution introduces large accuracy degradation across model sizes. $\Delta Acc_Q = Acc_{1bit} - Acc_{4bit}$ denotes the accuracy introduced by quantization and $\Delta Acc_G = Acc_{1bit,2\times} - Acc_{1bit}$ denotes the accuracy improvement by increasing channel counts. The ConvNet is VGG variant C with and without quantizing the depth-wise convolutions from 4 bits to 1 bit.

Width-multiplier	1.00×		1.25×		1.50×		1.75×		2.00×		Average	
Variant C	ΔAcc_Q	ΔAcc_G	ΔAcc_Q	ΔAcc_G	ΔAcc_Q	ΔAcc_G	ΔAcc_Q	ΔAcc_G	ΔAcc_Q	ΔAcc_G	ΔAcc_Q	ΔAcc_G
w/o Quantizing DWConv	-1.54	+2.61	-2.76	+2.80	-1.77	+1.74	-1.82	+1.64	-1.58	+1.55	-1.89	+2.07
Quantizing DWConv	-8.60	+4.39	-7.60	+3.41	-7.74	+3.19	-8.61	+4.09	-7.49	+2.25	-8.01	+3.47

Why does the number of elements matter? In quantization-aware training, one needs to estimate some statistics of the vector to be quantized (*i.e.*, $\boldsymbol{a}$ in Equation 1 and $|\bar{\boldsymbol{w}}|$ in Equations 3,4) based on the elements in the vector. The number of elements affect the robustness of the estimate that further decides the quantized weights. More formally, we provide the following proposition.

Proposition 1. *Let $\boldsymbol{w} \in \mathbb{R}^d$ be the weight vector to be quantized where $\boldsymbol{w}_i$ is characterized by normal distribution $\mathcal{N}(0, \sigma^2)$ $\forall$ i without assuming samples are drawn independently and $d = C_{in}K_wK_h$. If the average correlation of the weights is denoted by ρ, the variance of $|\bar{\boldsymbol{w}}|$ can be written as follows:*

$$\mathrm{Var}(|\bar{\boldsymbol{w}}|) = \frac{\sigma^2}{d} + \frac{(d-1)\rho\sigma^2}{d} - \frac{2\sigma^2}{\pi}. \tag{7}$$

The proof is in Appendix C. This proposition states that, as the number of elements (d) increases, the variance of the estimate can be reduced (due to the first term in equation (7)). The second term depends on the correlation between weights. Since the weights might not be independent during training, the variance is also affected by their correlations.

We empirically validate Proposition 1 by looking into the sample variance of $|\bar{\boldsymbol{w}}|$ across the course of training[8] for different d values by increasing (K_w, K_h) or C_{in}. Specifically, we consider the 0.5× VGG variant C and change the number of elements of the depth-wise convolutions. Let $d = (C_{in} \times K_w \times K_h)$ for a convolutional layer, we consider the original depth-wise convolution, *i.e.*, $d = 1\times3\times3$ and increased channels with $d = 4 \times 3 \times 3$ and $d = 16\times3\times3$, and increased kernel size with $d = 1\times6\times6$ and $d = 1\times12\times12$.

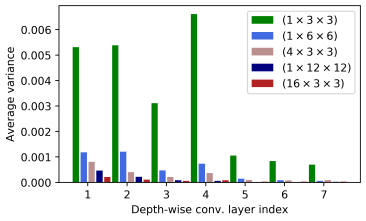


Fig. 5: The average estimate $\mathrm{Var}(|\bar{\boldsymbol{w}}|)$ for each depth-wise convolution under different $d = (C_{in} \times K_w \times K_h)$ values.

[8] We treat the calculated $|\bar{\boldsymbol{w}}|$ at each training step as a sample and calculate the sample variance across training steps.

The numbers are selected such that increasing the channel count results in the same d compared to increasing the kernel sizes. We note that when the channel count (C_{in}) is increased, it is no longer a depth-wise convolution, but rather a group convolution.

In Fig. 5, we analyze layer-level sample variance by averaging the kernel-level sample variance in the same layer. First, we observe that results align with Proposition 1. That is, one can reduce the variance of the estimate by increasing the number of elements along both the channel (C_{in}) and kernel size dimensions (K_w, K_h). Second, we find that increasing the number of channels (C_{in}) is more effective in reducing the variance than increasing kernel size (K_w, K_h), which could be due to the weight correlation, *i.e.*, intra-channel weights have larger correlation than inter-channel weights.

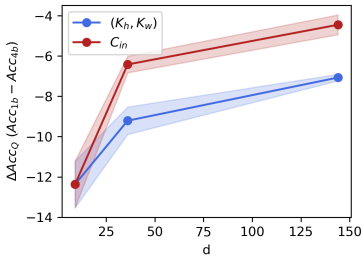


Fig. 6: d negatively correlates with the variance and positively correlates with the accuracy difference induced by quantization $\Delta Acc_Q = Acc_{1bit} - Acc_{4bit}$.

Nonetheless, while lower variance suggests a more stable value during training, it might not necessarily imply lower quantization error for the quantized models. Thus, we conduct an accuracy sensitivity analysis with respect to quantization for different d values. More specifically, we want to understand how d affects the accuracy difference between lower bitwidth (1 bit) and higher bitwidth (4 bits) models (ΔAcc_Q). As shown in Fig. 6, we empirically find that d positively correlates with ΔAcc_Q, *i.e.*, the larger the d, the smaller the accuracy degradation is. On the other hand, when comparing channel counts and kernel sizes, we observe that increasing the number of channels is more effective than increasing the kernel size in reducing accuracy degradation caused by quantization. This analysis sheds light on the two different trends observed in Fig. 1.

4.4 Remarks and scaling up to ImageNet

We have two intriguing findings so far. First, there exists some bitwidth that is better than others across model sizes when compared under a given model size. Second, the optimal bitwidth is architecture-dependent. More specifically, the optimal weight bitwidth negatively correlates with the fan-in channel counts per convolutional kernel. These findings show promising results for the hardware and software researchers to support only a certain set of bitwidths when it comes to parameter-efficiency. For example, use binary weights for networks with all-to-all convolutions.

Next, we scale up our analysis to the ImageNet dataset. Specifically, we study ResNet50 and MobileNetV2 on the ImageNet dataset. Since we keep the bitwidth of the first and last layer quantized at 8 bits, scaling them in terms of width will grow the number of parameters much more quickly than other

Table 2: bitwidth ordering for MobileNetV2 and ResNet50 with *the model size aligned to the* 0.25× *8 bits models* on ImageNet. Each cell reports the top-1 accuracy of the corresponding model. The trend for the optimal bitwidth is similar to that of CIFAR-100 (4 bit for MobileNetV2 and 1 bit for ResNet).

Weight bitwidth for Convs \ DWConvs	MobileNetV2				ResNet50
	8 bits	4 bits	2 bits	1 bit	None
8 bits	52.17	53.89	50.51	48.78	71.11
4 bits	56.84	**59.51**	57.37	55.91	74.65
2 bits	53.89	57.10	55.26	54.04	75.12
1 bit	54.82	58.16	56.90	55.82	**75.44**

layers. As a result, we keep the number of channels for the first and last channel fixed for the ImageNet experiments. As demonstrated in Section 4.2, the bit ordering is consistent across model sizes, we conduct our analysis for ResNet50 and MobileNetV2 by scaling them down with a width-multiplier of 0.25× for computational considerations. The choices of bitwidths are limited to $\{1, 2, 4, 8\}$.

As shown in Table 2, we can observe a trend similar to the CIFAR-100 experiments, *i.e.*, for networks without depth-wise convolutions, the lower weight bitwidths the better, and for networks with depth-wise convolutions, there are sweet spots for depth-wise and other convolutions. Specifically, the final weight bitwidth selected for MobileNetV2 is 4 bits for both depth-wise and standard convolutions. On the other hand, the selected weight bitwidth for ResNet50 is 1 bit. If bit ordering is indeed consistent across model sizes, these results suggest that the optimal bitwidth for MobileNetV2 is 4 bit and it is 1 bit for ResNet50. However, throughout our analysis, we have not considered mixed-precision, which makes it unclear if the so-called optimal bitwidth (4 bit for MobileNetV2 and 1 bit for ResNet-50) is still optimal when compared to mixed-precision quantization.

As a result, we further compare with mixed-precision quantization that uses reinforcement learning to find the layer-wise bitwidth [29]. Specifically, we follow [29] and use a reinforcement learning approach to search for the lowest bitwidths without accuracy degradation (compared to the 8 bits fixed point models). To compare the searched model with other alternatives, we use width-multipliers on top of the searched network match the model size of the 8 bit quantized model. We consider networks of three sizes, *i.e.*, the size of $1\times, 0.5\times$ and $0.25\times$ 8-bit fixed point models. As shown in Table 3, we find that a single bitwidth (selected via Table 2) outperforms both 8 bit quantization and mixed-precision quantization by a significant margin for both networks considered. This results suggest that searching for the bitwidth without accuracy degradation is indeed a sub-optimal strategy and can be improved by incorporating channel counts into the search space and reformulate the optimization problem as maximizing accuracy under storage constraints. Moreover, our results also imply that when the number of channels are allowed to be altered, a single weight bitwidth

Table 3: The optimal bitwidth selected in Table 2 is indeed better than 8 bit when scaled to larger model sizes and more surprisingly, it is better than mixed-precision quantization. All the activations are quantized to 8 bits.

Width-multiplier for 8-bit model		1×		0.5×		0.25×	
Networks	Methods	Top-1 (%)	$\mathcal{C}_{size}$ (10^6)	Top-1 (%)	$\mathcal{C}_{size}$ (10^6)	Top-1 (%)	$\mathcal{C}_{size}$ (10^6)
ResNet50	Floating-point	76.71	816.72	74.71	411.48	71.27	255.4
	8 bits	76.70	204.18	74.86	102.87	71.11	63.85
	Flexible [29]	77.23	204.18	76.04	102.90	74.30	63.60
	Optimal (1 bit)	**77.58**	204.08	**76.70**	102.83	**75.44**	63.13
MobileNetV2	Floating-point	71.78	110.00	63.96	61.76	52.79	47.96
	8 bits	71.73	27.50	64.39	15.44	52.17	11.99
	Flexible [29]	72.13	27.71	65.00	15.54	55.20	12.10
	Optimal (4 bit)	**73.91**	27.56	**68.01**	15.53	**59.51**	12.15

throughout the network shows great potential for model compression, which has the potential of greatly reducing the software and hardware optimization costs for quantized ConvNets.

5 Conclusion

In this work, we provide the first attempt to understand the ordering between different weight bitwidths by allowing the channel counts of the considered networks to vary using the width-multiplier. If there exists such an ordering, it may be helpful to focus on software/hardware support for higher-ranked bitwidth when it comes to parameter-efficiency, which in turn reduces software/hardware optimization costs. To this end, we have three surprising findings: (1) there exists a weight bitwidth that is better than others across model sizes under a given model size constraint, (2) the optimal weight bitwidth of a convolutional layer negatively correlates to the fan-in channel counts per convolutional kernel, and (3) with a single weight bitwidth for the whole network, one can find configurations that outperform layer-wise mixed-precision quantization using reinforcement learning when compared under a given same model size constraint. Our results suggest that when the number of channels are allowed to be altered, a single weight bitwidth throughout the network shows great potential for model compression.

Acknowledgement

This research was supported in part by NSF CCF Grant No. 1815899, NSF CSR Grant No. 1815780, and NSF ACI Grant No. 1445606 at the Pittsburgh Supercomputing Center (PSC).

References

1. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
2. Cai, H., Zhu, L., Han, S.: Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332 (2018)
3. Chin, T.W., Ding, R., Zhang, C., Marculescu, D.: Towards efficient model compression via learned global ranking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
4. Choi, J., Chuang, P.I.J., Wang, Z., Venkataramani, S., Srinivasan, V., Gopalakrishnan, K.: Bridging the accuracy gap for 2-bit quantized neural networks (qnn). arXiv preprint arXiv:1807.06964 (2018)
5. Ding, R., Chin, T.W., Liu, Z., Marculescu, D.: Regularizing activation distribution for training binarized deep networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
6. Dong, Z., Yao, Z., Gholami, A., Mahoney, M., Keutzer, K.: Hawq: Hessian aware quantization of neural networks with mixed-precision. arXiv preprint arXiv:1905.03696 (2019)
7. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=rJl-b3RcF7
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
9. He, Y., Liu, X., Zhong, H., Ma, Y.: Addressnet: Shift-based primitives for efficient convolutional neural networks. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 1213–1222. IEEE (2019)
10. Hou, L., Kwok, J.T.: Loss-aware weight quantization of deep networks. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=BkrSv0lA-
11. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
12. Huang, G., Liu, S., van der Maaten, L., Weinberger, K.Q.: Condensenet: An efficient densenet using learned group convolutions. group **3**(12), 11 (2017)
13. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
14. Jung, S., Son, C., Lee, S., Son, J., Han, J.J., Kwak, Y., Hwang, S.J., Choi, C.: Learning to quantize deep networks by optimizing quantization intervals with task loss. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
15. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep., Citeseer (2009)
16. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. International Conference on Learning Representation (ICLR) (2017)
17. Louizos, C., Reisser, M., Blankevoort, T., Gavves, E., Welling, M.: Relaxed quantization for discretized neural networks. In: International Conference on Learning Representations (2019), https://openreview.net/forum?id=HkxjYoCqKX

18. Meller, E., Finkelstein, A., Almog, U., Grobman, M.: Same, same but different: Recovering neural network quantization error through weight factorization. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 4486–4495. PMLR, Long Beach, California, USA (09–15 Jun 2019), http://proceedings.mlr.press/v97/meller19a.html
19. Mishra, A., Nurvitadhi, E., Cook, J.J., Marr, D.: WRPN: Wide reduced-precision networks. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=B1ZvaaeAZ
20. Nagel, M., van Baalen, M., Blankevoort, T., Welling, M.: Data-free quantization through weight equalization and bias correction. arXiv preprint arXiv:1906.04721 (2019)
21. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: European Conference on Computer Vision. pp. 525–542. Springer (2016)
22. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
23. Sheng, T., Feng, C., Zhuo, S., Zhang, X., Shen, L., Aleksic, M.: A quantization-friendly separable convolution for mobilenets. In: 2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2). pp. 14–18. IEEE (2018)
24. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
25. Stamoulis, D., Chin, T.W.R., Prakash, A.K., Fang, H., Sajja, S., Bognar, M., Marculescu, D.: Designing adaptive neural networks for energy-constrained image classification. In: Proceedings of the International Conference on Computer-Aided Design. p. 23. ACM (2018)
26. Stamoulis, D., Ding, R., Wang, D., Lymberopoulos, D., Priyantha, B., Liu, J., Marculescu, D.: Single-path nas: Designing hardware-efficient convnets in less than 4 hours. arXiv preprint arXiv:1904.02877 (2019)
27. Tan, M., Chen, B., Pang, R., Vasudevan, V., Le, Q.V.: Mnasnet: Platform-aware neural architecture search for mobile. arXiv preprint arXiv:1807.11626 (2018)
28. Theis, L., Korshunova, I., Tejani, A., Huszár, F.: Faster gaze prediction with dense networks and fisher pruning. arXiv preprint arXiv:1801.05787 (2018)
29. Wang, K., Liu, Z., Lin, Y., Lin, J., Han, S.: Haq: Hardware-aware automated quantization with mixed precision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8612–8620 (2019)
30. Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., Gholaminejad, A., Gonzalez, J., Keutzer, K.: Shift: A zero flop, zero parameter alternative to spatial convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9127–9135 (2018)
31. Wu, B., Wang, Y., Zhang, P., Tian, Y., Vajda, P., Keutzer, K.: Mixed precision quantization of convnets via differentiable neural architecture search. arXiv preprint arXiv:1812.00090 (2018)
32. Yang, J., Shen, X., Xing, J., Tian, X., Li, H., Deng, B., Huang, J., Hua, X.s.: Quantization networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
33. Ye, J., Lu, X., Lin, Z., Wang, J.Z.: Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. International Conference on Learning Representation (ICLR) (2018)

34. Yu, R., Li, A., Chen, C.F., Lai, J.H., Morariu, V.I., Han, X., Gao, M., Lin, C.Y., Davis, L.S.: Nisp: Pruning networks using neuron importance score propagation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018)
35. Yuan, X., Ren, L., Lu, J., Zhou, J.: Enhanced bayesian compression via deep reinforcement learning. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
36. Zhao, R., Hu, Y., Dotzel, J., De Sa, C., Zhang, Z.: Improving neural network quantization without retraining using outlier channel splitting. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 7543–7552. PMLR, Long Beach, California, USA (09–15 Jun 2019), http://proceedings.mlr.press/v97/zhao19c.html
37. Zhao, R., Hu, Y., Dotzel, J., Sa, C.D., Zhang, Z.: Building efficient deep neural networks with unitary group convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 11303–11312 (2019)
38. Zhou, A., Yao, A., Guo, Y., Xu, L., Chen, Y.: Incremental network quantization: Towards lossless cnns with low-precision weights. In: International Conference on Learning Representations (2017), https://openreview.net/forum?id=HyQJ-mclg
39. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., Zou, Y.: Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160 (2016)
40. Zhu, C., Han, S., Mao, H., Dally, W.J.: Trained ternary quantization. In: International Conference on Learning Representations (2017), https://openreview.net/forum?id=S1_pAu9xl
41. Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., Zhu, J.: Discrimination-aware channel pruning for deep neural networks. In: Advances in Neural Information Processing Systems. pp. 883–894 (2018)

A Clipping Point for Quantization-aware Training

As mentioned earlier, $\boldsymbol{a} \in \mathbb{R}^{C_{out}}$ denotes the vector of clipping factors which is selected to minimize $\|Q(\boldsymbol{W}_{i,:}) - \boldsymbol{W}_{i,:}\|_2^2$ by assuming $\boldsymbol{W}_{i,:} \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. More specifically, we run simulations for weights drawn from a zero-mean Gaussian distribution with several variances and identify the best $a_i^* = \arg\min_{a_i} \|Q_{a_i}(\boldsymbol{W}_{i,:}) - \boldsymbol{W}_{i,:}\|_2^2$ empirically. According to our simulation, we find that one can infer a_i from the sample mean $|\bar{\boldsymbol{W}}_{i,:}|$, which is shown in Fig. 7. As a result, for the different precision values considered, we find $c = \frac{|\bar{\boldsymbol{W}}_{i,:}|}{a_i^*}$ via simulation and use the obtained c to calculate a_i on-the-fly throughout training.

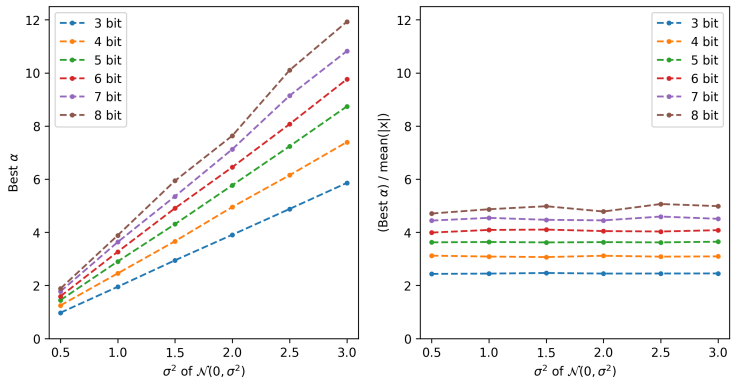


Fig. 7: Finding best a_i for different precision values empirically through simulation using Gaussian with various σ^2.

B Network Architectures

For the experiments in Section 4.2, the ResNets used are detailed in Table 4. Specifically, for the points in Fig. 1a, we consider ResNet20 to ResNet56 with width-multipliers of $0.5\times, 1\times, 1.5\times$, and $2\times$ for the 4-bit case. Based on these values, we consider additional width-multipliers $2.4\times$ and $2.8\times$ for the 2-bit case and $2.5\times, 3\times, 3.5\times$, and $3.9\times$ for the 1-bit case. We note that the rightmost points in Fig. 1a is a $10\times$ ResNet26 for the 4 bits case. On the other hand, VGG11 is detailed in Table 6 for which we consider width-multipliers from $0.25\times$ to $2\times$ with a step of 0.25 for the 4 bits case (blue dots in Fig. 1b). The architecture of MobileNetV2 used in the CIFAR-100 experiments follows the original MobileNetV2 (Table 2 in [22]) but we change the stride of all the bottleneck blocks to 1 except for the fifth bottleneck block, which has a stride of 2. As a result, we down-sample the image twice in total, which resembles the ResNet design for the CIFAR experiments [8]. Similar to VGG11, we consider

width-multipliers from 0.25× to 2× with a step of 0.25 for MobileNetV2 for the 4 bits case (blue dots in Fig. 1c).

Table 4: ResNet20 to ResNet56

Layers	20	26	32	38	44	50	56
Stem	Conv2d (16,3,3) Stride 1						
Stage 1	$3\times$ { Conv2d$(16,3,3)$ Stride 1; Conv2d$(16,3,3)$ Stride 1 }	4×	5×	6×	7×	8×	9×
Stage 2	$3\times$ { Conv2d$(32,3,3)$ Stride 2; Conv2d$(32,3,3)$ Stride 1 }	4×	5×	6×	7×	8×	9×
Stage 3	$3\times$ { Conv2d$(64,3,3)$ Stride 2; Conv2d$(64,3,3)$ Stride 1 }	4×	5×	6×	7×	8×	9×

Table 5: Inv-ResNet26

Stem	Conv2d (16,3,3) Stride 1
Stage 1	$4\times$ { Conv2d$(16\times 6,1,1)$ Stride 1; DWConv2d$(16\times 6,3,3)$ Stride 1; Conv2d$(16,1,1)$ Stride 1 }
Stage 2	$4\times$ { Conv2d$(32\times 6,1,1)$ Stride 1; DWConv2d$(32\times 6,3,3)$ Stride 2; Conv2d$(32,1,1)$ Stride 1 }
Stage 3	$4\times$ { Conv2d$(64\times 6,1,1)$ Stride 1; DWConv2d$(64\times 6,3,3)$ Stride 2; Conv2d$(64,1,1)$ Stride 1 }

Table 6: VGGs

VGG11	Variant A	Variant B	Variant C
Conv2d (64,3,3)			
MaxPooling			
Conv2d (128,3,3)	$\begin{cases}\text{Conv2d}(128,1,1)\\ \text{DWConv2d}(128,3,3)\end{cases}$	$\begin{cases}\text{Conv2d}(128,1,1)\\ \text{DWConv2d}(128,3,3)\end{cases}$	$\begin{cases}\text{Conv2d}(128,1,1)\\ \text{DWConv2d}(128,3,3)\end{cases}$
MaxPooling			
Conv2d (256,3,3)	Conv2d (256,3,3)	$\begin{cases}\text{Conv2d}(256,1,1)\\ \text{DWConv2d}(256,3,3)\end{cases}$	$\begin{cases}\text{Conv2d}(256,1,1)\\ \text{DWConv2d}(256,3,3)\end{cases}$
Conv2d (256,3,3)	Conv2d (256,3,3)	$\begin{cases}\text{Conv2d}(256,1,1)\\ \text{DWConv2d}(256,3,3)\end{cases}$	$\begin{cases}\text{Conv2d}(256,1,1)\\ \text{DWConv2d}(256,3,3)\end{cases}$
MaxPooling			
Conv2d (512,3,3)	Conv2d (512,3,3)	$\begin{cases}\text{Conv2d}(512,1,1)\\ \text{DWConv2d}(512,3,3)\end{cases}$	$\begin{cases}\text{Conv2d}(512,1,1)\\ \text{DWConv2d}(512,3,3)\end{cases}$
Conv2d (512,3,3)	Conv2d (512,3,3)	Conv2d (512,3,3)	$\begin{cases}\text{Conv2d}(512,1,1)\\ \text{DWConv2d}(512,3,3)\end{cases}$
MaxPooling			
Conv2d (512,3,3)	Conv2d (512,3,3)	Conv2d (512,3,3)	$\begin{cases}\text{Conv2d}(512,1,1)\\ \text{DWConv2d}(512,3,3)\end{cases}$
Conv2d (512,3,3)	Conv2d (512,3,3)	Conv2d (512,3,3)	$\begin{cases}\text{Conv2d}(512,1,1)\\ \text{DWConv2d}(512,3,3)\end{cases}$
MaxPooling			

C Proof for Proposition 5.1

Based on the definition of variance, we have:

$$\begin{aligned}\mathrm{Var}(\frac{1}{d}\sum_{i=1}^{d}|\boldsymbol{w}_i|) &:= \mathbb{E}\left[\left(\frac{1}{d}\sum_{i=1}^{d}|\boldsymbol{w}_i|\right)^2 - \left(\mathbb{E}\frac{1}{d}\sum_{i=1}^{d}|\boldsymbol{w}_i|\right)^2\right]\\ &= \mathbb{E}\left[\left(\frac{1}{d}\sum_{i=1}^{d}|\boldsymbol{w}_i|\right)^2 - \frac{2\sigma^2}{\pi}\right]\\ &= \frac{1}{d^2}\mathbb{E}\left(\sum_{i=1}^{d}|\boldsymbol{w}_i|\right)^2 - \frac{2\sigma^2}{\pi}\\ &= \frac{\sigma^2}{d} + \frac{d-1}{d}\rho\sigma^2 - \frac{2\sigma^2}{\pi}.\end{aligned}$$