Vertical Layering of Quantized Neural Networks for Heterogeneous Inference

Hai Wu, Ruifei He, Haoru Tan, Xiaojuan Qi, Kaibin Huang, *Fellow, IEEE*

Abstract—Although considerable progress has been obtained in neural network quantization for efficient inference, existing methods are not scalable to heterogeneous devices as one dedicated model needs to be trained, transmitted, and stored for one specific hardware setting, incurring considerable costs in model training and maintenance. In this paper, we study a new vertical-layered representation of neural network weights for encapsulating all quantized models into a single one. It represents weights as a group of bits (*i.e.,* vertical layers) organized from the most significant bit (also called the basic layer) to less significant bits (*i.e.,* enhance layers). Hence, a neural network with an arbitrary quantization precision can be obtained by adding corresponding enhance layers to the basic layer. With this representation, we can theoretically achieve any precision network for on-demand service while only needing to train and maintain one model. However, we empirically find that models obtained with existing quantization methods suffer severe performance degradation if adapted to vertical-layered weight representation. A significant issue is that once a quantization model is well-trained for a certain precision, it is difficult to reuse the weights and transfer them to another precision since essential information lying in the full precision source model is discarded after dedicated training. To this end, we propose a simple once quantization-aware training (QAT) scheme for obtaining high-performance vertical-layered models. Our design incorporates a cascade downsampling mechanism which allows us to obtain multiple quantized networks from one full precision source model by progressively mapping the higher precision weights to their adjacent lower precision counterparts. The mapping is achieved by discarding the least significant bits of the higher precision model. Then, with networks of different bit-widths from one source model, multi-objective optimization is employed to train the shared source model weights such that they can be updated simultaneously, considering the performance of all networks. By doing this, the shared weights will be optimized to balance the performance of different quantized models, thus making the weights transferable among different bit widths. After the model is trained, to construct a vertical-layered network, the lowest bit-width quantized weights become the basic layer, and every bit dropped along the downsampling process act as an enhance layer. Our design is extensively evaluated on CIFAR-100 and ImageNet datasets. Experiments show that the proposed vertical-layered representation and developed once QAT scheme are effective in embodying multiple quantized networks into a single one and allow one-time training, and it delivers comparable performance as that of quantized models tailored to any specific bit-width. Code will be available.

Index Terms—layered coding, bit-width scalable network, quantization-aware training, multi-objective optimization

1 Introduction

Deep neural networks (DNNs) have achieved revolutionary progress in many intelligent services, including image recognition, video segmentation, machine translation, and autonomous driving. However, the large storage footprint and heavy computation cost hinder the deployment of DNNs on edge devices with computation constraints, and accelerating DNNs has therefore become a significant research problem.

Among existing model acceleration techniques, neural network quantization is considered a promising direction to shrinking model size and making inference efficient by reducing the number of bits involved in the calculation, transmission, and storage [1], [2], [3]. The quantized networks are often obtained via a dedicated quantization-aware training (QAT) process [4], [5], [6], [7], [8], [9], where the quantized parameters are used in the forward pass to mimic the inference process, and the gradient approximated by straight-through-estimator [10] is obtained in the backward pass to update the full precision source model parameters. Although promising results have been attained, existing schemes only allow obtaining one network of a specific precision at a time. To serve heterogeneous devices, many dedicated models have to be designed, trained, and stored for diverse hardware settings. This incurs huge computational costs in model training and requires ample storage space to save all quantized models, making existing schemes not scalable to heterogeneous settings; see Fig. 1(a).

To this end, we propose a new *vertical-layered representation* of neural network weights for encapsulating all quantization

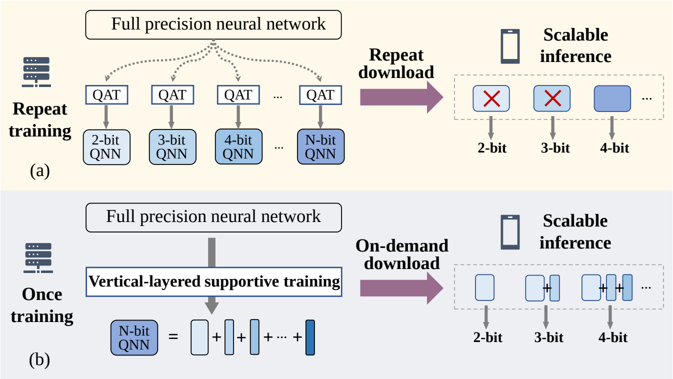


Fig. 1. For heterogeneous deployments: (a) existing quantization frameworks have to conduct quantization-aware training and store one separate model for each specific setting; and (b) our model with vertical-layered weight representation allows one-time training and encapsulates all quantized models into a single one for on-demand deployment.

- *Hai Wu, Ruifei He, Xiaojuan Qi and Kaibin Huang are with the Department of Electrical and Electronic Engineering at The University of Hong Kong, Hong Kong.*

models of different precisions into a single one. It represents weights as a group of bits, which are coined as **vertical layers**, and organizes them from the most significant bit (basic layer) to a series of less significant ones (enhance layers). Each vertical layer represents one quantization resolution and is expected to be compatible with the others for assembling different bit-width quantization networks. Hence a neural network with the desired quantization setting can be obtained by adding a number of enhance layers to the basic layer model. With this representation, we only need to train one vertical-layered model and store one copy of the parameters, which can be used to assemble any precision quantized network to support heterogeneous inference; see Fig. 1(b). Albeit conceptually appealing, obtaining a vertical-layered neural network remains a challenging problem. Our initial investigation shows that the performance of existing methods degrades dramatically if the models are treated as a vertical-layered network directly; see Sec. 3.3.

The major obstacle to developing such vertical-layered networks lies in the diverse weight distribution tailored to different bit-width settings and difficulties in optimizing the model involving multiple groups of discrete parameters simultaneously. As illustrated in Fig. 5(a), to achieve the best performance for all precision levels, the distribution and range of weights vary in different bit-width settings: they concentrate around zero-point and have a narrow range for lower precision networks (*e.g.,* 2-bit) but tend to show a long-tail distribution in high bit-width settings (*e.g.,* 4-bit). This makes network weights optimized to a particular precision level challenging to be reused by another precision level, which causes performance degradation; see Fig. 5(b). Besides, joint training of discrete model parameters using the existing method to make them fit all precision levels suffers from the conflict requirements of networks with different bit widths, which is unfolded in Sec. 3.3. Therefore, a new QAT scheme needs to be carefully re-designed to alleviate interference among vertical-layered representations and tackle the challenges in optimizing multiple quantization networks, especially at extremely low bit-width settings.

To overcome the challenges above, we propose a once QAT scheme for obtaining a desired vertical-layered network without sacrificing model performance too much. First, to support vertical-layered representation and alleviate conflicts among networks of different bit widths, we introduce a cascade downsampling mechanism that directly maps weights of a relatively higher precision level to their adjacent lower precision counterparts by dropping the least significant bits residing in the former ones. Hence, based on one full precision source model, we can have multiple quantization networks of different precisions by progressively reducing its bit widths. Second, given multiple quantized networks from one full precision source model, we optimize the underlying source parameters by minimizing an overall loss of all quantized networks and employ multi-objective optimization (MOO) to balance the contribution of each quantized model to the overall loss. Through such training, the shared source parameters will be updated to balance the performance of all quantized networks and make the weights transferable among networks of different precisions; see Fig. 5(c). Third, as our design naturally encloses multiple networks with diverse capabilities, we study a self-knowledge distillation paradigm where the lower precision network can get supervised by the higher one to strengthen its performance instead of resorting to extra teacher models. Finally, after our source model is trained, to construct a vertical-layered network, the lowest bit-width quantized weights become the basic layer, and every bit dropped in the downsampling process acts as an enhance layer.

We conduct extensive experiments on CIFAR-100 [11] and ImageNet datasets [12] to verify the vertical-layered weight representation and assess the effectiveness of our proposed design. To our best knowledge, without resorting to computationally heavy neural architecture search methods, we are the first to demonstrate a high-performance vertical-layered low-bit-width model which embeds multiple quantized networks into a single one and allows one-time training. At the same time, for all evaluated bit widths, our model can deliver comparable performance as that of models optimized for a specific bit width. Notably, our model upon ResNet [13] backbone can even outperform existing methods that need QAT for each bit width in some settings. Further, we extend the uniform-precision trained model via MOO to the mixed-precision setting by sampling a series of quantized weights from our model via minimizing the weight quantization error given a total network size budget. We obtain performance improvements as the number of bit budgets increases, which manifests the scalability of our method in incorporating many quantized models into a single one.

The remaining of this paper is organized as follows. Related works are summarized in Sec. 2. Then we revisit neural network quantization and introduce the concept of vertical layering in Sec. 3. We elaborate our design for training vertical-layered representation models in Sec. 4 and make a detailed discussion in Sec. 5. We conduct experiments and present concise analysis in Sec. 6. Finally, we draw our conclusion in Sec. 7.

2 Related Work

2.1 Neural Network Quantization

Neural network quantization attempts to approximate the full precision data using a fixed set of code words for saving bits [1], thus slimming the model size and converting the general floating-point arithmetic into low-complexity integer operations. Based on training schemes, existing methods can be categorized into two major streams: the post-training quantization (PTQ) scheme and the quantization-aware training (QAT) technique. Early works adopted PTQ of DNNs [14], [15], [16], [17], [18], where the mapping from full precision weights and activations to fixed integers are optimized after the traditional network training. PTQ requires no (or only a small amount of) training data for optimization and achieves nearly no accuracy drop in *int8* quantization. Yet, it generally suffers significant degradation when shrinking the quantization bit-width further [15]. Another group of works [3], [4], [6], [7], [9], *i.e.,* QAT, incorporates the quantization process along with the network training to optimize the network weights for a more compact quantization with better performance. QAT inserts the weight and activation discretization process in the network forward pass and uses the straight-through estimator [10] to approximate the gradient for updating original full precision parameters, which generally needs dedicated optimization. Two extreme cases, binary neural networks (BNNs) [19], [20], [21], [22] and XOR-Net [23], [24], use only binary or ternary representations of weights or activations to achieve single bit forward inference, which immensely reduces the arithmetic calculation at the cost of significant accuracy degradation.

Existing schemes can also be categorized into uniform-precision quantization and mixed-precision quantization based on the uniformity of quantization precision. The uniform-precision quantization requires different layers within one network to share the same bit-width settings [3], [4], [14]. The mixed-precision quantization scheme allows the usage of different bit widths for different layers within one network [25], which aims to find a suitable bit allocation scheme for each layer of the network model to minimize the accuracy degradation within the limited bit operation/storage budget. Compared with uniform-precision quantization, mixed-precision quantization can achieve better deployment flexibility and good trade-offs in performance and efficiency. Therefore, it has received extensive attention from the academic community [25], [26], [27]. Since mixed-precision quantization often requires complex hardware adaptation, practitioners often choose the hardware-friendly uniform-precision quantization technology in practice.

However, the above works focus on training one network with a certain precision at one time, which is not scalable for heterogeneous application scenarios. In this paper, we focus on developing a vertical-layered representation of neural network weights to embed multiple neural networks into one model, offering one-time training to serve on-demand deployment. To that end, we develop new QAT schemes. Besides, although our primary analysis is conducted on uniform-precision quantization, we further show the flexibility of our method by extending it to the mixed-precision scenario.

2.2 Dynamic Neural Networks

Dynamic neural networks adjust architecture or parameters to satisfy different computation requirements while maximizing model performance. Architecture-oriented dynamic networks often change the computation graph by selectively activating a subset of model components [28]. Adaptive neural network [29] introduces early-exit branches at different depth levels such that the model can generate outputs at intermediate layers. Further, some other works [30], [31], [32] add gating modules or skip controllers to drop layers or channels during inference. Unlike the above approaches realizing dynamic depth inference, the slimmable network [33] can execute at different layer-wise widths, *i.e.,* the number of active channels. Recently, neural architecture search has become another paradigm for achieving dynamic networks, which generally trains a superfluously parameterized network (*i.e.,* supernet), and a desirable sub-network can be retrieved through sampling and transformation [34], [35], [36]. Based on the once-for-all network [37], which firstly enlarges the architecture search space to obtain promising elastic models to suit varied deployment scenarios, the work in [38] incorporates the mixed precision selection and pruning for parameters in the searching stages to achieve further flexibility. These techniques provide a satisfactory network architecture under given resource constraints, which is orthogonal with our proposed vertical-layered network representation.

The work most related to ours is the Adabits [39], which achieves adaptive bit-width inference based on QAT. However, it only achieves satisfactory accuracy in high bit-width settings (4-bit and above). It is difficult for the model to obtain a low-bit-width model suited for vertical-layered representation, which requires no access to the full precision source parameter for scalable deployment. Meanwhile, existing works demonstrate that a 4-bit quantization network is already sufficient to achieve the performance of a full precision network baseline model [7] [9], which undermines the applicability of this work. In contrast, we focus on more challenging low-bit-width scenarios and demonstrate that our developed vertical-layered model achieves favourable performance at an extremely low-bit-width setting (*i.e.,* 2-bit) while maintaining dynamic flexibility.

2.3 Multi-Objective Optimization

Multi-objective optimization (MOO) involves simultaneously optimizing a set of possibly conflicting objectives, which is popularly adopted in multi-task learning [40], [41]. One widespread practice is performing a weighted sum of all objective functions to transform the original problem into a single-objective problem. The prior knowledge of the combination weights is critical for achieving the optimum, *i.e.,* the Pareto optimal solution [42]. Then, the gradient-based optimization methods are developed to iteratively update the combination weights by solving a convex combination problem, which provides a descent direction to arrive at the Pareto front by analyzing the multi-objective Karush-Kuhn-Tucker (KKT) conditions [43], [44], [45]. Recent work [46] demonstrates that the different MOO optimizers act as regularizers in the original MOO problem, and the unitary scalarization is capable of achieving the optimal solutions. Since our vertical-layered model embodies multiple networks that can yield multiple outputs, the training process must be carefully coordinated to balance different objectives. In this work, we formulate the training of a vertical-layered supportive network as a MOO problem and investigate different MOO optimizers to obtain a high-performance vertical-layered quantization model.

2.4 Knowledge Distillation

Knowledge distillation transfers the knowledge learned by large models (*i.e.,* teacher) to a compact one (*i.e.,* student) via a weighted combination of original loss using ground-truth labels and distillation loss calculated with the soft output of a teacher model [47]. There is a broad consensus that knowledge distillation can help improve the performance of low-bit-width quantized networks by transferring the knowledge of a full precision network to quantized ones [48], [49], [50]. QKD [49] proposes a quantization-aware knowledge distillation scheme by co-training a low-bit-width student and a larger teacher model. SPEQ [50] introduces a novel self-training paradigm where the high-precision teacher and the low-bit-width student from the same full precision source model are trained collaboratively. In our training process, where different bit-width networks are enclosed, we design a self-knowledge transfer that utilizes a relatively higher precision network's output as soft labels for the cooperative training of the lower precision ones.

3 Quantization and Vertical-Layered Representation

In the following, we first revisit the quantization of neural networks in Sec. 3.1. Then in Sec. 3.2, we elaborate on the proposed vertical-layered representation and demonstrate how to decompose the weight parameters into multiple vertical layers. Furthermore, in Sec. 3.3, we provide some baseline methods and present the challenge of obtaining vertical-layered models.

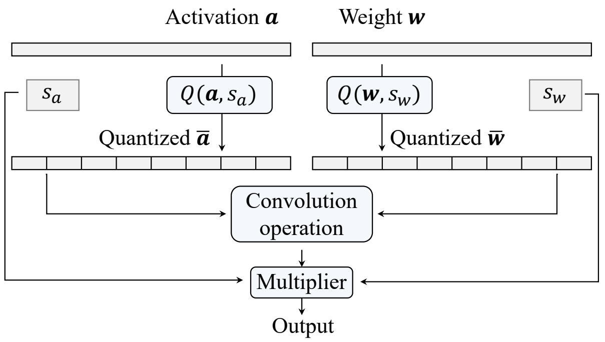


Fig. 2. The data flow of a quantized convolution operator. The quantization of weight can be done offline, and activation quantization is performed online. The quantized values are represented in low bit-width to simplify the convolution multiplication.

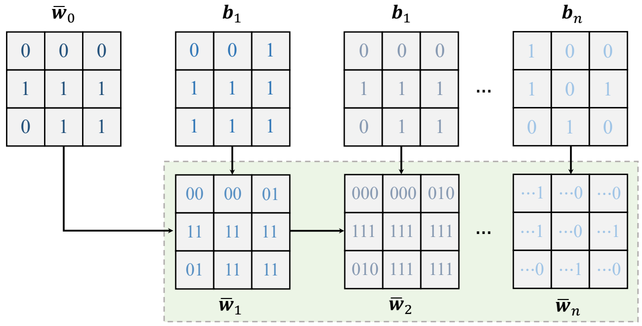


Fig. 3. Vertical-layered representation of a 2D 3×3 convolution kernel.

3.1 Quantization

Without loss of generality, we consider the uniform quantization process [1]. Given a real-valued tensor $\boldsymbol{v}$, which can be weight parameters $\boldsymbol{w}$ or input activation $\boldsymbol{a}$, the quantization process $Q(\cdot,\cdot)$ and de-quantization process $\hat{Q}(\cdot,\cdot)$ can be formulated as:

$$\begin{aligned} \bar{\boldsymbol{v}} &= Q(\boldsymbol{v}, s) = \mathrm{clip}(\lfloor \frac{\boldsymbol{v}}{s} \rceil, -Q_N, Q_P), \\ \hat{\boldsymbol{v}} &= \hat{Q}(\bar{\boldsymbol{v}}, s) = \bar{\boldsymbol{v}} \times s; \end{aligned} \tag{1}$$

where $\bar{\boldsymbol{v}}$ and $\hat{\boldsymbol{v}}$ denote the quantized tensor and de-quantized tensor respectively, $\lfloor \cdot \rceil$ indicates the rounding operation that maps a continuous number to its nearest integer number, and $\mathrm{clip}(\boldsymbol{x}, r_1, r_2)$ returns a tensor with entries below r_1 set to r_1 and entries above r_2 set to r_2. We use $s \in \mathbb{R}_+$ to denote the quantization step size, which determines the smallest resolution of a quantization level. Another quantization parameter, $[-Q_N, Q_P]$, represents the quantization range that constrains the smallest quantized integer to be $-Q_N$ and the largest integer to be Q_P. Given the number of bits K, for signed data, $Q_N = 2^{K-1}$ and $Q_P = 2^{K-1} - 1$; for the unsigned case, $Q_N = 0$ and $Q_P = 2^K - 1$.

The data flow involved in a quantized convolution operator during the inference stage is illustrated in Fig. 2. The real-valued weight $\boldsymbol{w}$ and activation $\boldsymbol{a}$ are quantized into low bit-width representations $\bar{\boldsymbol{w}}$ and $\bar{\boldsymbol{a}}$ respectively according to Eq. (1). Separate step sizes s_a and s_w are used for weight and activation quantization respectively. Then, quantized weights $\bar{\boldsymbol{w}}$ and activations $\bar{\boldsymbol{a}}$ are used to perform the low-precision convolution operation, *i.e.,*

$$\boldsymbol{y} = (\bar{\boldsymbol{w}} \otimes \bar{\boldsymbol{a}}) \times s_w \times s_a; \tag{2}$$

where $\otimes$ denotes the simplified convolution calculation. Finally, the final output $\boldsymbol{y}$ is obtained by multiplying these two step sizes on the computed results.

In QAT, the forward phase is calculated using the quantized/de-quantized tensor, while the backward phase aims to update the full precision source parameters. Since the discrete rounding and clip operation between the full precision $\boldsymbol{v}$ and the quantized $\bar{\boldsymbol{v}}$ in Eq. (1) truncate the gradient propagation path, the gradient estimator (*e.g.,* straight-through-estimator [10]) must be adopted for enabling gradient back-propagation from $\bar{\boldsymbol{v}}$ to $\boldsymbol{v}$. The step size s could be set as fixed values as in many existing works [4], [6], [7], [51]. The work in [9] demonstrates that treating step sizes as learnable parameters and jointly optimizing them with the model parameters will improve model performance. In this work, we adopt such a learnable quantization step size for good flexibility and high efficiency.

3.2 Vertical Layering of Network Weights

Existing quantization methods consider designing one dedicated model for a specific precision, which is not scalable to practical heterogeneous settings. In the following, we will introduce our proposed simple vertical-layered representation of neural network weights, which allows encompassing multiple quantized networks into a single one.

As shown in Fig. 3, the vertical-layered representation divides the weight parameters into different groups of bits. The quantized weight parameters, $\bar{\boldsymbol{w}}$, are partitioned into a low bit-width tensor $\bar{\boldsymbol{w}}_0$ and a series of independent and hierarchical binary tensors $\{\boldsymbol{b}_i\}_{i=1}^n$, each one is of the same shape as original $\boldsymbol{w}$. The parameter set $\bar{\boldsymbol{w}}_0$ consisting of the most significant bit of weight is termed as the *basic layer*. And the sequential groups of less significant bits $\{\boldsymbol{b}_i\}_{i=1}^n$ are called *enhance layers*. One can directly use the basic layer $\bar{\boldsymbol{w}}_0$ to construct a quantized neural network running at the lowest precision, and also can build up a higher bit-width quantized network by adding additional enhance layers to $\bar{\boldsymbol{w}}_0$ sequentially. With vertical-layered representation, heterogeneous edge devices can access any desired bit-width quantization network in an on-demand manner (see Fig. 1(b)).

To support the vertical-layered representation of quantized neural networks, the following rule is constructed to build the relationship between quantized weights of different bit widths,

$$\begin{aligned} \bar{\boldsymbol{w}}_i &= 2 \times \bar{\boldsymbol{w}}_{i-1} + \boldsymbol{b}_i, \\ s_i &= {}^1\!/_2 \times s_{i-1}; \end{aligned} \tag{3}$$

where $\bar{\boldsymbol{w}}_{i-1}$ indicates the adjacent lower bit-width quantized weights of the quantized weights $\bar{\boldsymbol{w}}_i$. The high precision $\bar{\boldsymbol{w}}_i$ is constructed by integrating the i-th enhance layer $\boldsymbol{b}_i$ with the lower bit-width $\bar{\boldsymbol{w}}_{i-1}$. Thus, the subscript $i \in \{1, ..., n\}$ indicates that the quantized weight is assembled by i enhance layers. This rule ensures that the high-precision quantized weights shall comprise the adjacent lower-precision quantized ones, which guarantees they are compatible with assembling any bit-width quantized

[1] It is necessary to distinguish between *uniform quantization* and *uniform-precision quantization*, where uniform quantization refers to the type of quantization in which the quantized levels are uniformly spaced, while uniform-precision quantization denotes that all layers within a network share the same bit-width settings. The non-uniform quantization generally requires a specific hardware design and is not favored in general heterogeneous applications.

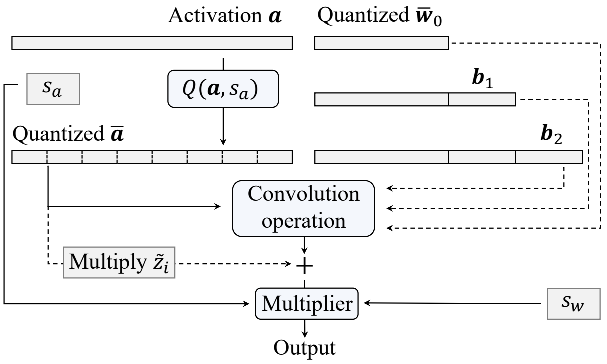


Fig. 4. Data flow for scalable inference. The quantized weights with any precision can be used forwarding convolution operation. And the input activation is quantized into corresponding bit width to match the computation precision.

networks. The recursive rule in Eq. (3) could be rewritten as the following form,

$$\bar{\boldsymbol{w}}_i = 2^i \times (\bar{\boldsymbol{w}}_0 + \sum_{j=1}^{i} 2^{-j} \cdot \boldsymbol{b}_j). \tag{4}$$

For a better illustration, we show the data flow in the convolution block under the vertical-layered representation setting in Fig. 4. The desired bit-width $\bar{\boldsymbol{w}}_i$ is obtained by incrementally adding a number of enhance layers to the basic layer. And the activation tensor $\boldsymbol{a}$ is quantized by $Q(\boldsymbol{a}, s_a)$ on the fly to achieve the corresponding precision to match the computation requirements. Finally, the corresponding step size s_i and s_a are used to scale the convolution results as the final output.

3.3 Baseline Method and Research Challenges

This subsection discusses the unresolved technical challenges faced by the vertical-layered representation. We introduce two intuitive baseline approaches for obtaining a vertical-layered neural network: the *ascending layer training* and the *multi-layer co-training*.

Ascending layer training. In this approach, the basic layer $\bar{\boldsymbol{w}}_0$ and each enhance layer in $\{\boldsymbol{b}_i\}_{i=1}^{n}$ are learned sequentially. Firstly, the basic layer $\bar{\boldsymbol{w}}_0$ is generated through dedicated QAT as elaborated in Sec. 3.1. Then, for training the binary enhance layer $\boldsymbol{b}_i$, we introduce another full precision source parameter $\boldsymbol{b}_i^r$ and set $\boldsymbol{b}_i = \text{sign}(\boldsymbol{b}_i^r)$, where the sign function outputs entry-wise $1/0$ according to the sign of each entry in $\boldsymbol{b}_i^r$. By substituting $\boldsymbol{b}_i$ into Eq. (3), we have

$$\bar{\boldsymbol{w}}_i = 2 \times \bar{\boldsymbol{w}}_{i-1} + \text{sign}(\boldsymbol{b}_i^r).$$

During training, the straight-through-estimator [10] is used to approximate the backward gradient from the binary $\boldsymbol{b}_i$ to source parameter $\boldsymbol{b}_i^r$. The training of all enhance layers contains n stages, and the i-th stage will learn the i-th enhance layer $\boldsymbol{b}_i$.

Multi-layer co-training. The second baseline is to optimize all vertical layers simultaneously. Following the settings in the first baseline, we also introduce the full precision tensor $\boldsymbol{b}_i^r$ to learn the binary enhance layer $\boldsymbol{b}_i$, where $\boldsymbol{b}_i = \text{sign}(\boldsymbol{b}_i^r)$ and the gradient is also estimated via the straight-through-estimator [10]. With Eq. (4), we can decompose any bit-width tensor $\bar{\boldsymbol{w}}_i$ into the combination of the basic layer and corresponding vertical layers,

$$\bar{\boldsymbol{w}}_i = 2^i \times (\bar{\boldsymbol{w}}_0 + \sum_{j=1}^{i} 2^{-j} \cdot \text{sign}(\boldsymbol{b}_j^r)).$$

During training, networks with different bit-width parameters perform forward and backward passes concurrently; hence all the source parameters of the vertical layers are optimized simultaneously.

Challenge analysis. We empirically evaluate the ascending layer training and multi-layer co-training methods and compare them with the oracle results (obtained by bit-width tailored QAT in [9]). The results are shown in Table 1. For the ascending layer training scheme, there is a nearly 2% top-1 accuracy degradation in the 3-bit and 4-bit networks in comparison with the oracle results. To investigate the reason, we visualize the weight histogram of the oracle network with different bit widths in Fig. 5(a). Obviously, the weight distributions of networks of different bit widths vary a lot, *i.e.,* the weights of a 4-bit quantized network show a more long-tailed distribution while the weights of the 2-bit one are more concentrated around zero. However, in the ascending layer scheme, the weight distribution of a higher bit-width network highly depends on the weight of the lower bit-width networks as the more significant bits are frozen when the model progressively grows by adding more ascending layers to support the vertical-layered representation. As for the second baseline, we find that the training process easily diverges because it involves concurrently optimizing multiple binary parameter sets. Therefore, we propose a new joint QAT scheme to achieve high-performance vertical layering supportive models.

TABLE 1
Baseline approaches investigation. Experiments are conducted using ResNet-18 [13] architecture on ImageNet dataset [12]. The basic layer is composed of a 2-bit quantized network. W/A indicates the weight and activation are quantized into W-bit and A-bit respectively. Top-1 accuracy is reported.

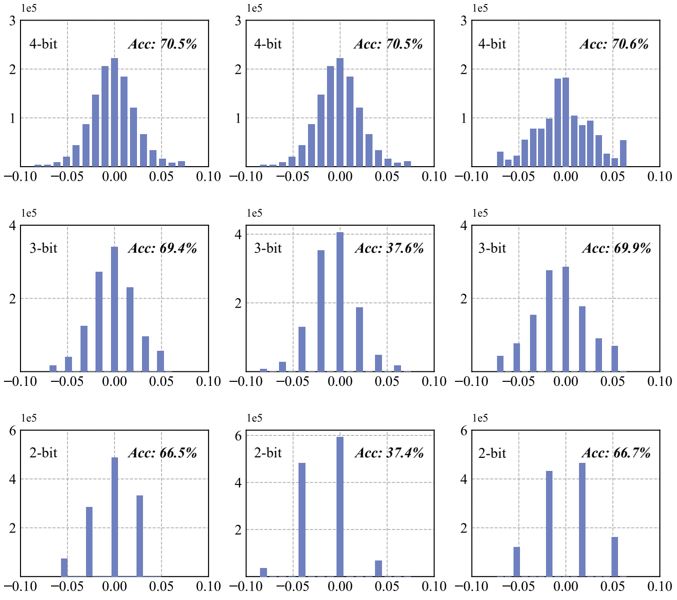
Method	Top-1 Acc (%) @ W/A		
	2 / 2	3 / 3	4 / 4
LSQ [9]	66.5	69.4	70.5
Ascending layer training	66.5	67.9	68.4
Multi-layer co-training		NaN	

4 Our Method

This section provides the methodology for enabling high-performance vertical-layered weight representation within a single model. Our method includes the downsampling quantization scheme (Sec. 4.1) to allow vertical-layered representation and the joint training paradigm (Sec. 4.2) to optimize multiple networks simultaneously.

4.1 Downsampling Quantization

As mentioned in Sec. 3.3, it is challenging to directly learn all discrete vertical layers since the difficulty in optimizing multiple sets of discrete parameters simultaneously. Instead, we build a direct mapping between two adjacent precision quantized weights



(a). LSQ trained (b). Directly downsampled (c). Our result

Fig. 5. Weight histograms of different bit-width quantized weights in ResNet-18 trained on ImageNet dataset based on different approaches, Top-1 validation accuracy is noted. Column (a). Different bit-width quantized networks are trained individually using LSQ [9] to achieve the best representation capability under specific precision criteria. Column (b). The 4-bit network is obtained through QAT, and the lower bit-width weights are directly downsampled from the highest precision network with fine-tuning. Column (c). Different precision networks are jointly trained once based on our proposed training framework.

avoiding the direct optimization of separate enhance layers. The detailed illustration is shown in Fig. 6. First, we quantize the full precision weights $\boldsymbol{w}$ into the highest desired bit width $\bar{\boldsymbol{w}}_n$ through Eq. (1). We denote the step size s_w for quantization and set it as the step size of the highest precision quantized network, *i.e.,* $s_n = s_w$. Then, we introduce the recursive downsampling operation $D(\cdot)$ to generate the lower bit-width quantized weights $\bar{\boldsymbol{w}}_i$,

$$\bar{\boldsymbol{w}}_i = D(\bar{\boldsymbol{w}}_{i+1}) = \lfloor \frac{\bar{\boldsymbol{w}}_{i+1}}{2} \rfloor, \quad i \in \{n-1, \cdots, 0\}; \tag{5}$$

where we use the floor function $\lfloor \cdot \rfloor$ instead of the rounding function $\lfloor \cdot \rceil$ for obtaining the lower bit-width quantized value to satisfy the requirements of vertical-layered representation and avoid the data overflow [2]. Thus, the i-th binary enhance layer could be obtained by $\boldsymbol{b}_i = \bar{\boldsymbol{w}}_i - 2 \times \bar{\boldsymbol{w}}_{i-1}$. The downsampling operation satisfies the requirement of a vertical-layered representation since the mapping between $\bar{\boldsymbol{w}}_i$ and $\bar{\boldsymbol{w}}_{i-1}$ is exactly based on finding and discarding the least significant bit $\boldsymbol{b}_i$ of $\bar{\boldsymbol{w}}_i$. The downsampling operation can be applied recursively until the quantized weight achieves the lowest bit width, *i.e.,* the basic layer $\bar{\boldsymbol{w}}_0$. Therefore, the quantized weight integrated by i enhance layers can be obtained by applying downsampling on $\bar{\boldsymbol{w}}_n$ with $n - i$ times, which is further used to compose a neural network running at the specific precision as illustrated in Fig. 4. The enhance layers $\{\boldsymbol{b}_n, \boldsymbol{b}_{n-1}, \cdots, \boldsymbol{b}_1\}$ can be obtained by collecting the dropped bits along the sequential downsampling process. However, the direct employment of downsampling operation for obtaining the desired bit-width quantized weight will significantly deteriorate performance, see Fig. 5. We propose the following positive compensation scheme to address this issue.

Positive compensation. By comparing the low bit-width parameter distribution histogram, we find two main issues: *1).* most quantized weights become inactivated (*i.e.,* zero); *2).* the portion of parameters with positive value has been significantly reduced. The reason stems from the biased weight quantization introduced by direct downsampling. Specifically, the distributions of the parameters through traditional training can be approximated well by a zero-mean Bell-shaped distribution [18]; and the quantized weight through dedicated QAT mimics such distribution, see Fig. 5(a). While in Fig. 5(b), the direct use of downsampling quantization causes significant disturbance to the weight distribution: it makes most weights inactivate by grouping a large portion of weights and mapping them to zero value. This thus aggravates the asymmetry of parameter distributions, *i.e.,* the generated positive weights account for only a tiny proportion compared with the one from the oracle results (Fig. 5(a)). The above problems become more severe as the bit width decreases, especially for the 2-bit case. Consequently, the model's capability is harmed, making the performance significantly degraded.

The above analysis motivates us to revisit a common practice in biased quantization of neural networks [52], [53] to compensate for the unbalanced weight distribution. Specifically, for each bit-width quantized weight $\boldsymbol{w}_i$, we introduce a preset offset parameter for conducting the convolution operation. As illustrated in Fig. 4, the process is expressed as:

$$\begin{aligned} \boldsymbol{y} &= (\bar{\boldsymbol{w}}_i + z_i) \otimes \bar{\boldsymbol{a}} \times s_i \times s_a; \\ &= (\underbrace{\bar{\boldsymbol{w}}_i \otimes \bar{\boldsymbol{a}}}_{\text{convolution}} + \underbrace{\tilde{z}_i \times \bar{\boldsymbol{a}}}_{\text{overhead}}) \times s_i \times s_a; \end{aligned} \tag{6}$$

where $\tilde{z}_i$ is the effective value to make the scalar-tensor convolution operation yielding the same result through multiplication. In the vertical-layered representations, the compensation alleviates the weight asymmetry after downsampling, which can be seen as shifting the de-quantized weight in the positive direction a certain distance to match the original weight distribution; see Fig. 5(c). Besides, this process will not incur significant computation overheads as the additional term is a low-cost scalar-tensor multiplication, which can be further treated as a bias term.

4.2 Model Training

Equipped with the downsampling scheme for the vertical-layered representation, we only need to learn the full precision weights $\boldsymbol{w}$ and the source step size s_w to meet deployment requirements of different bit widths in the vertical-layered mode. In this subsection, we design an effective joint training strategy with a self-knowledge distillation paradigm to learn a high-performance vertical-layered model.

4.2.1 Joint Training Framework

To alleviate the conflicts among quantized networks with different bit widths, the joint training process uses the loss functions of all bit-width networks to guide the learning process. The overall training framework is illustrated in Fig. 6, where $Q(\cdot, \cdot)$ denotes the quantization operation using Eq. (1), and $D(\cdot)$ represents the proposed downsampling operation as described in Sec. 4.1. In the

[2] The rounding operation may not be able to reduce the bit width and thus violates the principles of the vertical-layered model. For example, given a 3-bit value $\bar{w}_i = 7$, the floor quantized value $\lfloor \frac{\bar{w}_i}{2} \rfloor = 3$ is in 2-bit, while the rounding quantized value $\lfloor \frac{\bar{w}_i}{2} \rceil = 4$ is still in 3-bit.

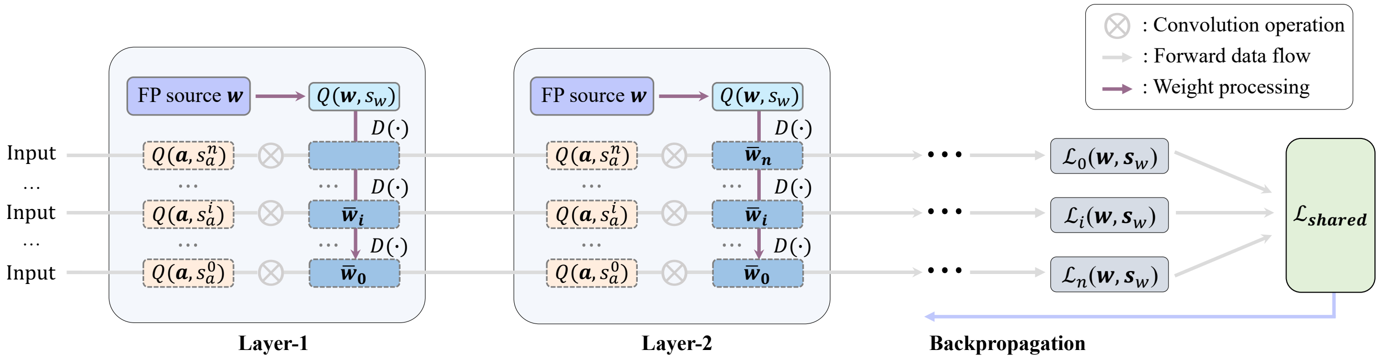

⊗ : Convolution operation
: Forward data flow
: Weight processing
FP source $\boldsymbol{w}$
$Q(\boldsymbol{w}, s_w)$
$D(\cdot)$
Input
…
Input
…
Input
$Q(\boldsymbol{a}, s_a^n)$
$Q(\boldsymbol{a}, s_a^i)$
$Q(\boldsymbol{a}, s_a^0)$
$\bar{\boldsymbol{w}}_n$
$\bar{\boldsymbol{w}}_i$
$\bar{\boldsymbol{w}}_0$
$\mathcal{L}_0(\boldsymbol{w}, \boldsymbol{s}_w)$
$\mathcal{L}_i(\boldsymbol{w}, \boldsymbol{s}_w)$
$\mathcal{L}_n(\boldsymbol{w}, \boldsymbol{s}_w)$
$\mathcal{L}_{shared}$
Layer-1
Layer-2
Backpropagation

Fig. 6. Structure of the proposed training framework. The loss set $\{\mathcal{L}_n, \cdots, \mathcal{L}_0\}$ is calculated by feeding the training data into networks with different bit-width parameters $\{\bar{\boldsymbol{w}}_n, \cdots, \bar{\boldsymbol{w}}_0\}$. These losses are combined and gradients are backpropagated to update the source parameters and the source step size.

forward path, we calculate the cross-entropy loses $\{\mathcal{L}_n, \cdots, \mathcal{L}_0\}$ by feeding the training data into networks and producing the outputs using parameters of different bit widths $\{\bar{\boldsymbol{w}}_n, \cdots, \bar{\boldsymbol{w}}_0\}$. These losses would be combined to a global loss $\mathcal{L}_{\text{shared}}$, which will be discussed in the following Sec. 4.2.2. In the backward pass, we use the straight-through estimator [10] to approximate the gradient of the non-differentiable operation, including the quantization operation $Q(\cdot, \cdot)$ and downsampling operation $D(\cdot)$. Specifically, for a specific full precision source weight w, the gradient of $\mathcal{L}_i$ with respect to it is approximated via

$$\frac{\partial \mathcal{L}_i}{\partial w} \approx \frac{\partial \mathcal{L}_i}{\partial \bar{w}_i} \times \frac{1}{s_i}. \tag{7}$$

For the corresponding step size $\{s_0, s_1, ..., s_n\}$, we can calculate the gradient of $\mathcal{L}_i$ with respect to s_i through $\bar{w}_i$ as,

$$\left.\frac{\partial \mathcal{L}_i}{\partial s_i}\right|_{\bar{w}_i} \approx \begin{cases} \dfrac{\partial \mathcal{L}_i}{\partial \bar{w}_i} \times \dfrac{1}{s_i} \times (-w/s_i + \bar{w}_i), & -Q_N < \bar{w}_i < Q_P; \\ \dfrac{\partial \mathcal{L}_i}{\partial \bar{w}_i} \times \dfrac{1}{s_i} \times (-Q_n^N), & \bar{w}_i < -Q_N; \\ \dfrac{\partial \mathcal{L}_i}{\partial \bar{w}_i} \times \dfrac{1}{s_i} \times (Q_p^N), & \bar{w}_i > Q_P. \end{cases} \tag{8}$$

Since the step sizes are coupled by the source quantization step size s_w, the gradient of $\mathcal{L}_i$ with respect to the source parameter s_w through specific $\bar{w}_i$ can be successively backpropagated for optimization. Then, the gradients from different losses are combined and further used to update source model weights $\boldsymbol{w}$ and the step size s_w. In the following section, we will elaborate on combining losses/gradients from different losses into a multi-objective optimization (MOO) framework to achieve the balance among optimizing multiple objectives.

4.2.2 Multi-Objective Optimization

It is critical to find a good trade-off between different objectives $\{\mathcal{L}_0, \mathcal{L}_1, \cdots, \mathcal{L}_n\}$ to ensure all intermediate quantized models could achieve competitive performance. To achieve this, we define the following multi-objective optimization (MOO) problem,

$$\mathcal{L}_{\text{shared}} = \sum_{i=0}^{n} \alpha_i \mathcal{L}_i(\boldsymbol{w}, \boldsymbol{s}_w); \tag{9}$$

where α_i is the scale factor of the corresponding loss function $\mathcal{L}_i(\boldsymbol{w}, \boldsymbol{s}_w)$, $\boldsymbol{w}$ and $\boldsymbol{s}_w$ denotes the set of all full precision weights and the source step size. We investigate two well-known baseline optimizers for the MOO problem: the unitary scalarization [46] and the multiple gradient descent [45].

Unitary scalarization. Unitary scalarization (US) [46] is a recently proposed multi-task learning framework, which suggests that using the summation of losses with appropriate scale factors is sufficient to obtain comparable results as the specialized multi-objective optimizers. It is suggested that the interference among multiple losses can be treated as a form of regularization [46]. Without introducing any priors on different objectives, we empirically set $\alpha_i = {}^{1}/_{(n+1)}$ to calculate the combined loss in the following experiments, see Sec. 6.

Multiple gradient descent. The multiple gradient descent (MGD) algorithm demonstrates that a smaller norm of the convex combination of gradients indicates minor interferences among different objectives by analyzing the KKT conditions [45]. It thus dynamically updates the combination factors α_i by minimizing the overall gradient norm as:

$$\underset{\alpha_0, \cdots, \alpha_n}{\arg\min} \left\| \sum_{i=0}^{n} \alpha_i \frac{\partial \mathcal{L}_i(\boldsymbol{w}, \boldsymbol{s}_w)}{\partial(\boldsymbol{w}, \boldsymbol{s}_w)} \right\|_2^2, \quad \text{s.t. } \alpha_i \geq 0 \text{ and } \sum_{i=0}^{n} \alpha_i = 1; \tag{10}$$

where the gradient of $\boldsymbol{w}$ and $\boldsymbol{s}_{\boldsymbol{w}}$ are calculated via Eq. (7) and Eq. (8) respectively. The work in [54] provides an effective update rule for α_i based on the Frank-Wolfe algorithm.

Besides the shared parameters above, our framework has some bit-width-specific parameters, including the batch normalization parameters (*i.e.,* mean, variance, and affine parameters) and all activation quantization step sizes. The mean and variance are updated in a moving average manner for each bit width separately. The affine parameters and step sizes for activations are optimized using the loss of the corresponding bit-width network loss $\mathcal{L}_i$, which we do not include in the MOO framework.

4.2.3 Self-Knowledge Distillation

It is a broad consensus that knowledge distillation is beneficial to improve the quantized models' performance [49], [55]. Here, we adopt the self-knowledge distillation to further improve model performance, where the outputs from a high bit-width network (teacher) are directly used to supervise the lower bit-width network (student). The overall loss function is formulated as

$$\mathcal{L}_i(\boldsymbol{w}, \boldsymbol{s}_w) = \mathcal{L}_{\text{CE}}^i + \mathcal{L}_{\text{KD}}(\mathbf{z}_{i+1}, \mathbf{z}_i); \quad i < n. \tag{11}$$

$\mathcal{L}^i_{\text{CE}}$ is the original cross-entropy training loss, $\mathcal{L}_{\text{KD}}$ is the knowledge distillation function, $\mathbf{z}_{i+1}$ and $\mathbf{z}_i$ denote the probabilistic softmax output from the teacher and the student respectively. We apply the cosine similarity function for constructing the distillation $\mathcal{L}_{\text{KD}}$, which has been demonstrated to be effective when the teacher's prediction is ambiguous [50].

5 Discussion

Mixed-precision quantization. The mixed-precision quantization network allows different layers residing in one network to adopt different bit-width computations for flexibility, especially when there are constraints for a model size or the total number of operations [25]. As our vertical-layered model is trained in a uniform-precision manner, we wonder if extending the obtained model in a mixed-precision setting is possible. Since the quantization error is a commonly adopted criterion for optimizing the bits allocation among layers within one network, we apply a similar criterion that minimizes the overall downsampling error constrained on a total number of bits budget, *i.e.,*

$$\begin{aligned} \underset{i_m\in\{0,\cdots,n\}}{\arg\min} \quad & \sum_{m=1}^{M} \|\bar{\boldsymbol{w}}_{i_m} - \bar{\boldsymbol{w}}_{n_m}\|_2^2, \\ \text{s.t.} \quad & \sum_{m=1}^{M} N_m(|\bar{\boldsymbol{w}}_{i_m}| - |\bar{\boldsymbol{w}}_{n_m}|) \le B; \end{aligned} \tag{12}$$

where $\bar{\boldsymbol{w}}_{i_m}$ is the quantized weight tensor integrated with i enhance layers in the network's m-th layer, $|\bar{\boldsymbol{w}}_{i_m}|$ is the bit width assigned to every entry in the m-th layer, N_m is the corresponding number of weight parameters and B is the total bits budget. The detailed implementation and analysis will be covered in Sec. 6.4.

Comparison with NAS-based approaches. Neural architecture search [25], [26], [27], [35], [38], [55] is a widely adopted technique in the mixed-precision quantization, which is orthogonal to our effort. For instance, by incorporating NAS to search for a more suitable bit width assignment strategy, the obtained model can attain better performance, which is not the focus of this paper. Besides, the model yielded from pure NAS-based approaches cannot offer graceful on-demand model upgrade/downgrade without accessing the superfluous source parameters. For instance, the NAS-based quantization method represented by [55] requires considerable computing resources and time to search the network structure for the specified bit-width, and the searched model also needs further careful fine-tuning to achieve satisfactory performance. In contrast, our method does not require a complex fine-tuning process and heavy searching process.

6 Experiments

6.1 Experimental Setup

We conduct extensive experiments on CIFAR-100 [11] and ImageNet ILSVRC2012 [12]. Experimental setups are shown as follows.

Comparison setup. Unlike our work, which mainly focuses on training vertical-layered networks, prior QAT works aim at optimizing the model for a specific bit width. We incorporate multiple state-of-the-art QAT approaches for comparisons, including PACT [7], LQ-Nets [6], DSQ [8], QIL [56], LQW [57]. Besides, we implement the quantized networks tailored to specific bit-width via QAT using LSQ [9], which serves as the oracle results and sets an upper bound performance of our model.

TABLE 2
Results of different quantization approaches on CIFAR-100 are presented. The network architecture used here is the ResNet-18. The network's weights and the corresponding layers' input are quantized into W/A-bit separately. The results marked with * are from [58].

Quantization scheme	Top-1 Acc (%) @ W/A 2 / 2	3 / 3	4 / 4	Vertical-layered
LQ-Nets* [6]	70.8	72.9	-	×
LQW* [57]	65.1	73.0	-	×
DiffQ* [58]	66.6	76.7	77.5	×
LSQ [9]	**76.9**	77.7	77.8	×
Ours	76.3	**77.8**	**78.3**	√

Training framework. The overall vertical-layered training framework is shown in Fig. 6. The full precision weights $\boldsymbol{w}$ and the source step sizes $\boldsymbol{s}_w$ are learned under the MOO framework introduced in Sec. 4.2.2. Different bit-width networks hold separate sets of quantization step sizes for the quantization of activations, which are learned through the specific loss $\mathcal{L}_i$ as detailed in [9]. By default, the batch normalization layers of networks with different bit widths are separate. Following the common practice [3], [7], [9], [23], the first and last layers of a DNN model are not quantized, and the weight parameters of these two layers are shared among different bit-width networks. We set the basic layer of our model as a 2-bit quantized network instead of 1-bit because the binary neural networks usually require dedicated network structures, which are incompatible with higher bit width quantization structures [19], [23]. The highest bit-width in our model is set to be 4-bit since a 4-bit quantized network can achieve a full precision network's performance [7].

Image pre-processing. For the CIFAR dataset, we follow a standard procedure in [60] where training images are zero-padded with 4 pixels on every side horizontally and vertically, and then a random 32×32 patch is cropped, which is followed by a random horizontal flip with a probability of 0.5. For the ImageNet dataset, images in the training set are randomly resized and cropped to 224×224 with a random horizontal flipping probability of 0.5. The images in the validation set are resized to 256×256 and center copped to 224×224 for evaluation.

Hyper-parameters. All networks are trained using stochastic gradient descent with a momentum of 0.9 using the cosine learning rate decay schedule without restart [61]. The softmax cross-entropy loss is applied to guide the learning process. We first train the full precision networks from scratch and then conduct the quantization-aware training. For the CIFAR dataset, we train 300 epochs with an initial learning rate of 0.1, batch size 128, and weight decay $1e-3$; the initial learning rate is scaled by 0.1 for training quantized networks. For ImageNet, the initial learning rate is 0.1, and batch size is 256 for ResNet-18 and ResNet-34; for ResNet-50, the initial learning rate is 0.05 with batch size 128. The weight decay is $1e-4$ for full precision network training. For quantized networks, all initial learning rates are set to be 0.01, and the same weight decay and batch size as full precision training are applied, except for a weight decay of $5e-5$ in ResNet-18. In all

TABLE 3
Comparison of different quantization approaches on ImageNet dataset. Top-1 and Top-5 accuracy (%) are reported separately. The network's weight and the corresponding layers' input are quantized into W/A-bit separately. The full precision (FP) models' Top-1 and Top-5 accuracy are attached in each model. Results marked with † are from the original paper. Results marked with § are from [59].

Network model	Quantization scheme	Top-1 Acc (%) @ W/A			Top-5 Acc (%) @ W/A			Vertical-layered
		2 / 2	3 / 3	4 / 4	2 / 2	3 / 3	4 / 4	
ResNet-18 *FP: 70.5 / 89.6*	PACT† [7]	64.4	68.1	69.2	85.6	88.2	89.0	×
	LQ-Nets† [6]	64.9	68.2	69.3	85.9	87.9	88.8	×
	DSQ† [8]	65.2	68.7	69.6	-	-	-	×
	QIL† [56]	65.7	69.2	70.4	-	-	-	×
	LSQ [9]	66.5	69.4	70.5	87.0	88.9	**89.5**	×
	Ours (*w/o* self-KD)	66.7	69.9	**70.6**	**87.3**	**89.1**	**89.5**	√
	Ours (*w.* self-KD)	**67.1**	**70.1**	**70.6**	87.2	**89.1**	**89.5**	√
ResNet-34 *FP: 74.1 / 92.8*	LQ-Nets† [6]	69.8	71.9	-	89.1	90.2	-	×
	DSQ† [8]	70.0	72.5	72.8	-	-	-	×
	QIL† [56]	70.6	73.1	73.7	-	-	-	×
	LSQ [9]	71.1	**73.6**	**74.1**	**90.0**	**91.5**	**91.6**	×
	Ours (*w/o* self-KD)	70.7	73.3	73.9	89.8	91.3	**91.6**	√
	Ours (*w.* self-KD)	**71.2**	73.5	74.0	89.8	91.3	91.5	√
ResNet-50 *FP: 76.9 / 93.4*	PACT† [7]	72.2	75.3	76.5	-	-	-	×
	LQ-Nets† [6]	71.5	74.2	75.1	90.3	91.6	92.4	×
	LSQ [9]	73.2	75.8	**76.8**	**91.3**	92.6	93.0	×
	Ours (*w/o* self-KD)	72.4	75.9	**76.8**	90.9	92.7	93.0	√
	Ours (*w.* self-KD)	**73.2**	**76.1**	**76.8**	91.1	**92.8**	**93.1**	√
MobileNetV2 *FP: 71.9 / 91.3*	PACT† [7]	-	-	61.4	-	-	83.7	×
	LSQ§ [9]	46.7	-	66.3	-	-	-	×
	Ours (*w/o* self-KD)	47.4	64.3	67.1	72.5	85.5	**87.8**	√
	Ours (*w.* self-KD)	**50.6**	**65.0**	**67.5**	**75.3**	**86.0**	**87.8**	√
VGG-16 BN *FP: 73.4 / 91.5*	LSQ† [9]	**71.4**	**73.4**	**74.0**	**90.4**	**91.5**	**92.0**	×
	Ours (*w/o* self-KD)	70.9	73.0	73.4	90.1	91.3	91.6	√
	Ours (*w.* self-KD)	71.0	73.3	73.4	89.9	91.2	91.4	√

experimented settings, we simply use a preset compensation value as $z_i = 2^{-1}(1 - 2^{i-2})$ to minimize the downsampling error, which is demonstrated to be sufficient for obtaining comparable accuracy as the bit-width tailored QAT results. The oracle bit-width tailored quantized neural networks are trained with the same fixed gradient scaling factors for fair comparisons.

Initialization. We initialize the network weights and all batch normalization parameters using well-trained full precision networks. The full precision MobileNetV2 [62] and VGG-16 [63] are from the Pytorch library [64]. For the quantization step size of weights, we initialize them using averaged absolute values of full precision weights, *i.e.*, $s_w = {}^{2 \times mean\{|\boldsymbol{w}|\}}/_{\sqrt{2^4-1}}$. For the quantization step size of activation, we use the first iteration's input activation to initialize every step size by $s_a^i = {}^{\max\{|\boldsymbol{a}|\}}/_{(2^{i+1}-1)}$.

6.2 Main results and Analysis

6.2.1 Experiments on CIFAR-100

We experiment with ResNet-18 [13] using different quantization methods on the CIFAR-100 dataset. The Top-1 validation accuracy is shown in Table 2. The network weights and the corresponding input are quantized into W-bit and A-bit, respectively, denoted as W/A. Our method significantly surpasses many baselines. For example, it exceeds LQ-Net [6] by 5.5% under the 2-bit setting, and outperforms DiffQ [58] by 1.1% and 0.8% under the 3-bit and 4-bit setting respectively. Moreover, our method shows comparable performance even in comparison with the bit-width tailored QAT using LSQ [9]. Compared with all baselines, only our method has the vertical-layered characteristic, that is, once training and scalable inference for various bit-width, while does not require separate quantization and QAT training tailored to specific bit-width as the baseline does. We can conclude that: *1).* our vertical-layered representation enables one-time training for scalable inference; *2).* our proposed joint training design can support vertical-layered settings well without sacrificing model performance.

6.2.2 Experiments on ImageNet

Main results. For the large-scale ImageNet dataset, we perform experiments on the ResNet family [13] and the MobileNetV2 [62]. Table 3 shows the Top-1 and Top-5 validation accuracy with state-of-the-art quantization methods. Among all compared methods,

TABLE 4
Results of using different batch normalization layers on ImageNet are compared. We summarize the number (M) of BN parameters under different BN settings and list their proportion (‰) in the total number of model parameters. Top-1 and Top-5 accuracy (%) are reported separately.

Model	BN Setting	# of BN Parameters	Top-1 Acc (%) @ W / A			Top-5 Acc (%) @ W / A		
			2 / 2	3 / 3	4 / 4	2 / 2	3 / 3	4 / 4
ResNet-18	Shared BN	0.19M (1.6‰)	33.5	67.4	49.3	58.0	87.8	74.1
	+ Individual Statistics	0.38M (3.3‰)	66.6	68.8	69.2	86.9	88.6	88.8
	++ Individual Weights	0.58M (4.9‰)	**66.7**	**69.9**	**70.6**	**87.3**	**89.1**	**89.5**
ResNet-34	Shared BN	0.34M (1.5‰)	56.2	72.2	41.1	79.7	90.8	66.6
	+ Individual Statistics	0.68M (3.0‰)	70.7	72.7	73.0	89.8	91.0	91.3
	++ Individual Weights	1.02M (4.5‰)	**70.7**	**73.3**	**73.9**	**89.8**	**91.3**	**91.6**

ours is the only one that can achieve once training for scalable inference without tailored QAT. The joint training framework (Sec. 4) achieves satisfactory performance in almost all evaluated bit-width settings, and self-KD further improves the model's Top-1 accuracy by $0.1\% - 0.5\%$. Our method shows comparable or even better performance in comparison with the bit-width-specific oracle LSQ method, especially for the lightweight model cases (ResNet-18 and MobileNetV2). For example, our joint training outperforms the oracle LSQ [9] by $0.6\%/0.9\%$ for 2-bit/3-bit ResNet-18 and $0.4\%/0.9\%$ for 2-bit/4-bit MobileNetV2 in terms of the Top-1 accuracy. In the case of a large model (ResNet-50), compared with the compared baselines, our method can effectively narrow the performance gap between 2-bit/3-bit and full precision networks, and even achieves 4-bit inference with only 0.1% performance sacrifice compared with the full precision one.

TABLE 5
Results of different scalable bit-width inference approaches on ImageNet are compared. The network architecture used here is the ResNet-50. Networks' weight and activation are quantized into W/A-bit respectively. Results marked with † are from the original paper.

Scheme	Weight size (bits)	Top-1 Acc (%) @ W/A		
		2 / 2	3 / 3	4 / 4
Ascending layer	159.6M	73.2	73.6	73.9
QAT + fine-tuning	159.6M	60.1	73.4	76.8
Adabits† [39]	816.1M	**73.2**	75.8	76.1
Ours (*w/o* self-KD)	159.6M	72.4	75.9	**76.8**
Ours (*w.* self-KD)	159.6M	**73.2**	**76.1**	**76.8**

One model for scalable bit-width inference. In this part, we compare different methods for obtaining a model that can support scalable bit-width inference. We implement the ascending layer approach described in Sec. 3.3, fine-tuning based on the highest bit-width network obtained via QAT, and our training framework. Here, we also compare with the related work called Adabits [39], which can be deployed for scalable bit-width inference based on a shared full precision model and is also optimized in a collaborative training manner. All experiments are conducted with ResNet-50 [13] on ImageNet [12], and results are reported in Table 5. Compared with naive approaches (ascending layer and QAT + fine-tuning), our method and Adabits significantly improve the performance of all assessed bit widths, which suggests that joint training is beneficial for scalable bit-width deployment. With only one-fifth of the weight parameters required by Adabit, our method achieves the highest accuracy in 3-bit and 4-bit settings. However, Adabits requires accessing the full precision parameters or storing all bit-width quantized models for scalable inference, which incurs large storage space requirements.

6.3 Ablation Study

In this subsection, we conduct the following ablation experiments to assess the effectiveness of different components of our model:

- Sec. 6.3.1 explores effective ways to construct batch normalization layers in vertical-layered models.
- Sec. 6.3.2 shows the effect of the positive compensation scheme used in the downsampling operation.
- Sec. 6.3.3 investigates different multi-objective optimizers for MOO training.
- Sec. 6.3.4 studies the effect of the self-knowledge distillation paradigm in MOO training.
- Sec. 6.3.5 shows the effect of using different first and last layers in vertical-layered models.

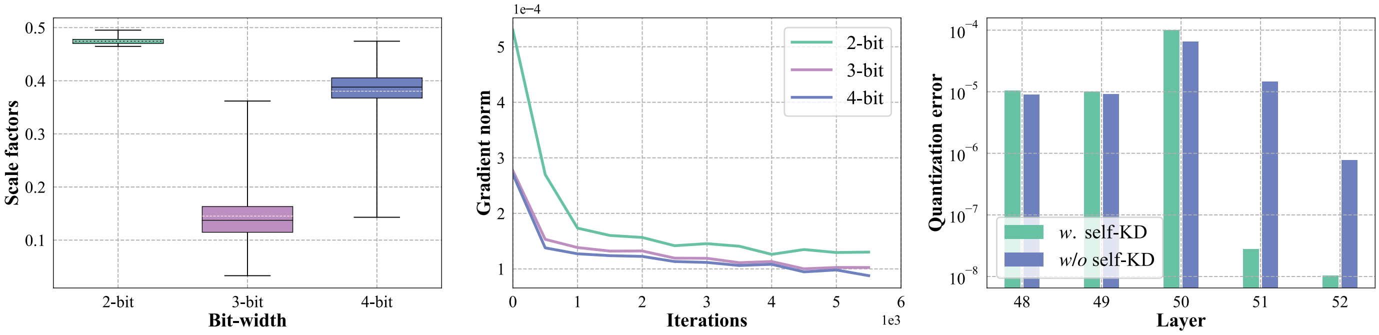
6.3.1 Effective Ways to Construct Batch Normalization Layer

We study three ways to construct batch normalization (BN) layers in the vertical-layered model training:
1). Shared BN: all batch normalization parameters, including the BN statistics (running mean and variance) and the affine parameters, are shared among networks of different bit widths.
2). Individual statistics: individual statistics with shared BN affine parameters for different bit-width precision quantized networks are applied.
3). Individual BN: BN layers are independent across different bit-width networks. This scheme has the highest flexibility and is the default setting in our main experiments.

We experiment with ResNet-18 and ResNet-34 on ImageNet and report the comparisons in Table 4. Experimental results show that the first choice (shared BN) does not converge well: the 2-bit ResNet-18 only achieves 33% top-1 accuracy, which is significantly lower than any other choice. Compared with the first setting, the individual statistics setting is more flexible and achieves better performance with negligible costs for increasing the number of parameters. The individual BN, the default setting in our experiments, achieves the highest performance among all these

TABLE 6
Results of different MOO optimizers used in the proposed training framework are compared. Top-1 and Top-5 validation accuracy (%) on ImageNet are reported separately. W/A indicates that the well-trained model is running on specific precision.

Model	Training approach	Top-1 Acc (%) @ W / A			Top-5 Acc (%) @ W / A		
		2 / 2	3 / 3	4 / 4	2 / 2	3 / 3	4 / 4
ResNet-18	Unitary Scalarization (US) [46]	66.7	**69.9**	**70.6**	**87.3**	**89.1**	**89.5**
	Multi-Gradient Descent (MGD) [45]	**66.8**	69.3	69.8	**87.3**	88.7	89.0
ResNet-34	Unitary Scalarization (US) [46]	70.7	**73.3**	**73.9**	89.8	**91.3**	**91.6**
	Multi-Gradient Descent (MGD) [45]	**70.9**	72.8	73.0	**89.9**	91.1	91.2



(a)

(b)

(c)

Fig. 7. Ablation study: (a). the statistics of scale factors calculated by MGD during ResNet-18 training on the ImageNet dataset; (b). the average gradient norm during MGD-based ResNet-18 training on the ImageNet dataset; (c). the average activation quantization error norm of the last five convolution layers (layer-48 to layer-52) in 2-bit ResNet-50 models trained with/without the self-KD scheme.

choices at the cost of less than 5‰ additional parameter overhead. For example, it exceeds the first one (shared BN) by 33% top-1 accuracy in the 2-bit ResNet-18 experiment and surpasses the second one (individual statistics) by about 1% top-1 accuracy in the 3-bit ResNet-18 experiment.

6.3.2 Effect of the Positive Compensation

In this part, we investigate the effect of introducing positive compensation in the downsampling process. Table 7 shows our experiment results with ResNet-18 and MobilNetV2 on the ImageNet dataset. We can observe that without positive compensation, the 2-bit ResNet-18 gets around a 5% Top-1 accuracy drop compared with the compensated one, and the 3-bit network also suffers from direct downsampling. For MobileNetV2, the QAT without compensation significantly drops all the bit-width network performance, especially in the 2-bit case where the Top-1 accuracy is decreased by 10%. As observed from Table 3, the joint training achieves competitive accuracy as the bit-width tailored QAT with the assistance of positive compensation in all bit-width settings.

6.3.3 Choice of the MOO Optimizer

The proposed vertical-layered training framework should find a balance between the losses of different bit-width networks, which constitutes a multi-objective optimization (MOO) problem. We use the unitary scalarization (US) as the MOO optimizer by default. We further explore whether other MOO optimizers, such as the popular multi-gradient descent (MGD), can make our model achieve better results. The MGD method iteratively updates the combination weight α_i by solving the min-norm point problem detailed in Eq. (10).

TABLE 7
Effect of involving positive compensation in downsampling is presented.

Model	Quantization Scheme	Top-1 Acc (%) @ W/A		
		2 / 2	3 / 3	4 / 4
ResNet-18	*w/o* compensation	61.3	67.4	70.5
	w. compensation	**66.7**	**69.9**	**70.6**
MobileNetV2	*w/o* compensation	40.4	57.5	63.6
	w. compensation	**50.6**	**65.0**	**67.5**

We present the results in Table 6. Although MGD is developed to benefit multi-task learning, it will not necessarily boost network performance compared to the simple US approach or even degrade the performance. In particular, MGD shades the performance significantly in 3-bit and 4-bit cases. We attribute this to the following two reasons: 1). MGD is not a stable choice here. Fig. 7(a) shows that the factors obtained by MGD have significant variances, which are unstable or even change dramatically since the inexact gradient is calculated via the unreliable biased STE [10] gradient estimator; 2). Fig. 7(b) shows that the gradient norm of the 2-bit parameter is of the greatest magnitude, especially in the early training stage. However, MGD puts more emphasis on the 2-bit parameter loss function (see Fig. 7(a)), while ignoring the 3-bit and 4-bit loss and further aggravates the imbalance between loss functions. Beyond these points, it slows down the overall training speed since MGD needs to solve the problem in Eq. (10) once per iteration. Therefore, the US strategy is the best choice regarding performance and speed.

TABLE 8
Effect of replacing full precision first convolutional and final fully-connected layer. The additional number of parameters for the first convolutional layer and final FC are summed. Top-1 and Top-5 accuracy (%) at bit-width W/A are reported separately.

Model	Replaced layers	# of additional parameters	Top-1 Acc (%) @ W / A			Top-5 Acc (%) @ W / A		
			2 / 2	3 / 3	4 / 4	2 / 2	3 / 3	4 / 4
ResNet-18	×	-	66.7	69.9	**70.6**	**87.3**	89.1	**89.5**
	√	1.04 M (9.4%)	**66.9**	**70.0**	70.5	**87.3**	**89.3**	89.4
ResNet-50	×	-	**72.4**	**75.9**	**76.8**	**90.9**	**92.7**	**93.0**
	√	4.12 M (17.6%)	71.0	74.8	75.8	90.3	92.3	92.7

6.3.4 Effect of Self-Knowledge Distillation

We further study the effectiveness of the self-KD paradigm described in Sec. 4.2.3. Table 9 compares the training results with (*w.*) and without (*w/o*) self-KD. As an effortless and lightweight training technique, self-KD positively affects the performance of vertical-layered models, especially for low-bit-width networks. In particular, the self-KD improves the 2-bit ResNet-50 by 0.8% (from 72.4% to 73.2%) in Top-1 accuracy. We analyze the quantization error of activations of the last five convolution layers from well-trained ResNet-50 models running at a 2-bit setting in Fig. 7(c). It is analyzed that deeper layers are more prone to activation quantization, which significantly degrades the network performance [65]. With the self-KD paradigm here, the averaged quantization errors of activation in the last two layers are dramatically reduced compared with the setting without self-KD, hence improving the quantized model performance by feeding more accurate features into the final classifier.

TABLE 9
Experiment results of using different self-KD settings in joint training are presented. We use KL and CS to denote the Kullback-Leibler divergence loss and the cosine similarity loss respectively.

Model	Self-KD	Top-1 Acc (%) @ W/A		
		2 / 2	3 / 3	4 / 4
ResNet-18	*w/o*	66.7	69.9	70.6
	KL	66.7	69.9	70.4
	CS	**67.1**	**70.1**	**70.6**
ResNet-50	*w/o*	72.4	75.9	**76.8**
	KL	72.1	75.8	**76.8**
	CS	**73.2**	**76.1**	**76.8**

Moreover, we also study the selection of the self-KD loss function by comparing the cosine similarity (CS) based self-KD with the Kullback–Leibler-divergence (KL) based one in Table 9. The experimental results show that the selection of distillation loss function is crucial, and the KL loss is almost useless. This is because the classic KL distillation loss may mislead model training since the teacher is sometimes unreliable, while the cosine similarity function serves as a more robust choice for ambiguous knowledge [50].

6.3.5 Effects of Replacing Full Precision First and Final Layers

For the quantization of neural networks, it is a common practice to keep the first convolutional layer and the last fully connected layer in full precision to preserve accuracy [23]. Our vertical-layered models share these parameters among quantized networks of different precision for convenient deployments. One might wonder whether the quantized networks' performance can be enhanced by making the first and last layers tailored to different bit widths. Under this circumstance, when the devices require more enhance layers, the server must also offload the corresponding first and last layers. We experiment with ResNet-18 and ResNet-50 trained on ImageNet, and the result is shown in Table 8. We can observe that replacing these two layers shows little effect on the accuracy improvement at the cost of a large portion of overhead parameters in ResNet-18. Regarding the ResNet-50 model, the results demonstrate that the joint optimization of parameters in these two layers, *i.e.,* using the same layers for different bit-width networks in Eq. (9), assists the quantized model training, especially for the lowest 2-bit quantized one. Hence, we do not recommend training individual first and last layers for scalable bit-width inference.

6.4 Mixed-Precision Network

In this part, we assess whether our vertical layered model supports the mixed-precision setting well. We accommodate our quantization network to the mixed-precision scenario according to the criterion of minimizing the quantization error given a limited number of bits on model weights at the unit of each layer, see Eq. (12). We experiment with ResNet-18 on ImageNet and plot the network accuracy against the respective model size in Fig. 8. Since the BN statistics vary in different precisions, the BN statistics in the sampled network are re-estimated using the training data. We can observe that the Top-1 validation accuracy increases with the enlarging of the overall number of bits in general, manifesting the effectiveness of our model in the mixed-precision setting. We note that the 3-bit uniform precision network performs worse than the mixed-precision quantized one with fewer bits requirement. This phenomenon can be explained by the different sensitivity for quantization over all layers in one network model [26], which highlights the advantages of mixed-precision quantization.

7 Conclusion

We have proposed a vertical-layered representation for quantizing deep neural networks, which allows heterogeneous devices to access the quantized model in an on-demand manner for scalable and efficient inference. To obtain a high-performance model that supports vertical-layered representation, we propose a once quantization-aware training framework with a downsampling operation, and multi-objective optimization techniques are employed

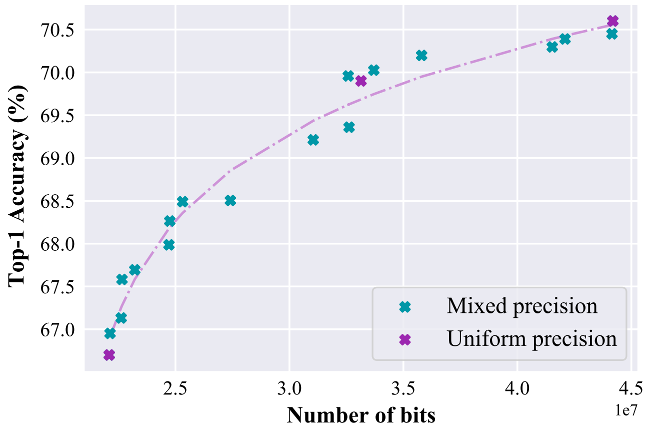


Fig. 8. Top-1 accuracy vs. model size for mixed precision quantized ResNet-18 on ImageNet dataset.

to balance the collaborative training process. Further, we find that the self-knowledge distillation paradigm can boost the training of the low-bit-width networks with the proper selection of a distillation function. Extensive experimental results demonstrate the feasibility of vertical-layered representation of quantized networks, where our model delivers competitive performance in all assessed precision settings. Our vertical-layered model achieves significant storage and transmission cost compression for scalable inference. We hope our study would inspire more research investigations on employing vertical-layered weight representation for quantization in heterogeneous application scenarios.

References

[1] S. Han, H. Mao, and W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding," in *International Conference on Learning Representations*, 2016.

[2] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[3] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[4] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, "Dorefa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint arXiv:1606.06160*, 2016.

[5] E. Park, S. Yoo, and P. Vajda, "Value-aware quantization for training and inference of neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 580–595.

[6] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *European Conference on Computer Vision (ECCV)*, 2018.

[7] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.

[8] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4852–4861.

[9] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," in *International Conference on Learning Representations*, 2020.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[11] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "ImageNet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[14] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Accurate post training quantization with small calibration sets," in *International Conference on Machine Learning*. PMLR, 2021, pp. 4466–4475.

[15] P. Wang, Q. Chen, X. He, and J. Cheng, "Towards accurate post-training network quantization via bit-split and stitching," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9847–9856.

[16] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] Y. Nahshan, B. Chmiel, C. Baskin, E. Zheltonozhskii, R. Banner, A. M. Bronstein, and A. Mendelson, "Loss aware post-training quantization," *Machine Learning*, vol. 110, no. 11, pp. 3245–3262, 2021.

[18] J. Fang, A. Shafiee, H. Abdel-Aziz, D. Thorsley, G. Georgiadis, and J. H. Hassoun, "Post-training piecewise linear quantization for deep neural networks," in *European Conference on Computer Vision*. Springer, 2020, pp. 69–86.

[19] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

[20] Z. Liu, W. Luo, B. Wu, X. Yang, W. Liu, and K.-T. Cheng, "Bi-Real Net: Binarizing deep network towards real-network performance," *International Journal of Computer Vision*, vol. 128, no. 1, pp. 202–219, 2020.

[21] B. Martinez, J. Yang, A. Bulat, and G. Tzimiropoulos, "Training binary neural networks with real-to-binary convolutions," in *International Conference on Learning Representations*, 2020.

[22] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5918–5926.

[23] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XOR-Net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[24] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *International Conference on Learning Representations*, 2017.

[25] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8612–8620.

[26] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "HAWQ: Hessian aware quantization of neural networks with mixed-precision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 293–302.

[27] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. Mahoney *et al.*, "HAWQ-V3: Dyadic neural network quantization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 875–11 886.

[28] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[29] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *International Conference on Machine Learning*. PMLR, 2017, pp. 527–536.

[30] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "SkipNet: Learning dynamic routing in convolutional networks," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 409–424.

[31] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8817–8826.

[32] W. Hua, Y. Zhou, C. M. De Sa, Z. Zhang, and G. E. Suh, "Channel gating neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[33] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," in *International Conference on Learning Representations*, 2019.

[34] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4095–4104.

[35] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," *arXiv preprint arXiv:1904.00420*, 2019.

[36] X. Li, C. Lin, C. Li, M. Sun, W. Wu, J. Yan, and W. Ouyang, "Improving one-shot nas by suppressing the posterior fading," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020.

[37] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations*, 2020.

[38] T. Wang, K. Wang, H. Cai, J. Lin, Z. Liu, H. Wang, Y. Lin, and S. Han, "Apq: Joint search for network architecture, pruning and quantization policy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[39] Q. Jin, L. Yang, and Z. Liao, "AdaBits: Neural network quantization with adaptive bit-widths," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[40] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *Advances in neural information processing systems*, vol. 31, 2018.

[41] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.

[42] C.-L. Hwang and A. S. M. Masud, *Multiple objective decision making—methods and applications: a state-of-the-art survey*. Springer Science & Business Media, 2012, vol. 164.

[43] J. Fliege and B. F. Svaiter, "Steepest descent methods for multicriteria optimization," *Mathematical methods of operations research*, vol. 51, no. 3, pp. 479–494, 2000.

[44] J. Fliege and A. I. F. Vaz, "A method for constrained multiobjective optimization based on sqp techniques," *SIAM Journal on Optimization*, vol. 26, no. 4, pp. 2091–2119, 2016.

[45] J.-A. Désidéri, "Mutiple-gradient descent algorithm for multiobjective optimization," in *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2012)*, 2012.

[46] V. Kurin, A. De Palma, I. Kostrikov, S. Whiteson, and M. P. Kumar, "In defense of the unitary scalarization for deep multi-task learning," *arXiv preprint arXiv:2201.04122*, 2022.

[47] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[48] A. Mishra and D. Marr, "Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy," *arXiv preprint arXiv:1711.05852*, 2017.

[49] J. Kim, Y. Bhalgat, J. Lee, C. Patel, and N. Kwak, "QKD: Quantization-aware knowledge distillation," *arXiv preprint arXiv:1911.12491*, 2019.

[50] Y. Boo, S. Shin, J. Choi, and W. Sung, "Stochastic precision ensemble: self-knowledge distillation for quantized deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 6794–6802.

[51] B. Zhuang, M. Tan, J. Liu, L. Liu, I. Reid, and C. Shen, "Effective training of convolutional neural networks with low-bitwidth weights and activations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[52] M. Nagel, M. v. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1325–1334.

[53] A. Finkelstein, U. Almog, and M. Grobman, "Fighting quantization bias with bias," in *CVPR Workshop*, 2019.

[54] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[55] M. Shen, F. Liang, R. Gong, Y. Li, C. Li, C. Lin, F. Yu, J. Yan, and W. Ouyang, "Once quantization-aware training: High performance extremely low-bit architecture search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5340–5349.

[56] S. Jung, C. Son, S. Lee, J. Son, J.-J. Han, Y. Kwak, S. J. Hwang, and C. Choi, "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4350–4359.

[57] T. Hoang, T.-T. Do, T. V. Nguyen, and N.-M. Cheung, "Direct quantization for training highly accurate low bit-width deep neural networks," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 2020, pp. 2111–2118, main track.

[58] A. Défossez, Y. Adi, and G. Synnaeve, "Differentiable model compression via pseudo quantization noise," *arXiv preprint arXiv:2104.09987*, 2021.

[59] Y. Li, M. Shen, J. Ma, Y. Ren, M. Zhao, Q. Zhang, R. Gong, F. Yu, and J. Yan, "MQBench: Towards reproducible and deployable model quantization benchmark," *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[60] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2016, pp. 87.1–87.12.

[61] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[62] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[63] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[64] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "PyTorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[65] Y. Boo, S. Shin, and W. Sung, "Quantized neural networks: Characterization and holistic optimization," in *2020 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2020, pp. 1–6.