Differentiable Joint Pruning and Quantization for Hardware Efficiency

Ying Wang$^{1[0000-0001-6866-2076]}$, Yadong Lu$^{2[0000-0001-8754-135X]\star}$, and Tijmen Blankevoort^{1}

^{1} Qualcomm AI Research**
{yinwan, tijmen}@qti.qualcomm.com
^{2} University of California, Irvine
yadongl1@uci.edu

Abstract. We present a differentiable joint pruning and quantization (DJPQ) scheme. We frame neural network compression as a joint gradient-based optimization problem, trading off between model pruning and quantization automatically for hardware efficiency. DJPQ incorporates variational information bottleneck based structured pruning and mixed-bit precision quantization into a single differentiable loss function. In contrast to previous works which consider pruning and quantization separately, our method enables users to find the optimal trade-off between both in a single training procedure. To utilize the method for more efficient hardware inference, we extend DJPQ to integrate structured pruning with power-of-two bit-restricted quantization. We show that DJPQ significantly reduces the number of Bit-Operations (BOPs) for several networks while maintaining the top-1 accuracy of original floating-point models (e.g., 53x BOPs reduction in ResNet18 on ImageNet, 43x in MobileNetV2). Compared to the conventional two-stage approach, which optimizes pruning and quantization independently, our scheme outperforms in terms of both accuracy and BOPs. Even when considering bit-restricted quantization, DJPQ achieves larger compression ratios and better accuracy than the two-stage approach.

Keywords: Joint optimization, model compression, mixed precision, bit-restriction, variational information bottleneck, quantization

1 Introduction

There has been an increasing interest in the deep learning community to neural network model compression, driven by the need to deploy large deep neural networks (DNNs) onto resource-constrained mobile or edge devices. So far, pruning and quantization are two of the most successful techniques in compressing a DNN for efficient inference [9][13][12][15]. Combining the two is often done in practice. However, almost no research has been published in combining the two

* Work done during internship at Qualcomm AI Research
** Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

in a principled manner, even though finding the optimal trade-off between both methods is non-trivial. There are two challenges with the currently proposed approaches:

First, most of the previous works apply a two-stage compression strategy: pruning and quantization are applied independently to the model [9][20][35]. We argue that the two-stage compression scheme is inefficient since it does not consider the trade-off between sparsity and quantization resolution. For example, if a model is pruned significantly, it is likely that the quantization bit-width has to be large since there is little redundancy left. Further, the heavily pruned model is expected to be more sensitive to input or quantization noise. Since different layers have different sensitivity to pruning and quantization, it is challenging to optimize these holistically, and manually chosen heuristics are probably not optimal.

Second, the joint optimization of pruning and quantization should take into account practical hardware constraints. Although unstructured pruning is more likely to result in higher sparsity, structured pruning is often preferable as it can be more easily exploited on general-purpose devices. Also, typical quantization methods use a fixed bit-width for all layers of a neural network. However, since several hardware platforms can efficiently support mixed-bit precision, joint optimization could automatically support learning the bit-width. The caveat is that only power-of-two bit-widths are often efficiently implemented and supported in typical digital hardware; other bit-widths are rounded up to the nearest power-of-two-two values, resulting in inefficiencies in computation and storage [14]. Arbitrary bit-widths used in some literature [29][30] are more difficult to exploit for power or computational savings unless dedicated silicon or FPGAs are employed.

To address the above challenges, we propose a differentiable joint pruning and quantization (DJPQ) scheme. It first combines the variational information bottleneck [4] approach to structured pruning and mixed-bit precision quantization into a single differentiable loss function. Thus, model training or fine-tuning can be done only once for end-to-end model compression. Accordingly, we show that on a practical surrogate measure Bit-Operations (BOPs) we employ, the complexity of the models is significantly reduced without degradation of accuracy (e.g., 53x BOPs reduction in ResNet18 on ImageNet, 43x in MobileNetV2).

The contributions of this work are:

– We propose a differentiable joint optimization scheme that balances pruning and quantization holistically.
– We show state-of-the-art BOPs reduction ratio on a diverse set of neural networks (i.e., VGG, ResNet, and MobileNet families), outperforming the two-stage approach with independent pruning and quantization.
– The joint scheme is fully end-to-end and requires training only once, reducing the efforts for iterative training and finetuning.
– We extend the DJPQ scheme to the bit restricted case for improved hardware efficiency with little extra overhead. The proposed scheme can learn mixed-precision for a power-of-two bit restriction.

2 Related work

Both quantization and compression are often considered separately to optimize neural networks, and only a few papers remark on the combination of the two. We can thus relate our work to three lines of papers: mixed-precision quantization, structured pruning, and joint optimization of the two.

Quantization approaches can be summarized into two categories: fixed-bit and mixed-precision quantization. Most of the existing works fall into the first category, which set the same bit-width for all layers beforehand. Many works in this category suffer from lower compression ratio. For works using fixed 4 or 8-bit quantization that are hardware friendly such as [19][8][1], either the compression ratio or performance is not as competitive as the mixed-precision opponents [29]. In [17] 2-bit quantization is utilized for weights, but the scheme suffers from heavy performance loss even with unquantized activations. For mixed-precision quantization, a second-order quantization method is proposed in [6] and [5], which automatically selects quantization bits based on the largest eigenvalue of the Hessian of each layer and the trace respectively. In [30][34] reinforcement-based schemes are proposed to learn the bit-width. Their methods determine the quantization policy by taking the hardware accelerator's feedback into the design loop. In [20], the bit-width is chosen to be proportional to the total variance of that layer, which is heuristic and likely too coarsely estimated. A differentiable quantization (DQ) scheme proposed in [29] can learn the bit-width of both weights and activations. The bit-width is estimated by defining a continuous relaxation of it and using straight-through estimator for the gradients. We employ a similar technique in this paper. For a general overview of quantization, please refer to [31][15].

Structured pruning approaches can also be summarized into two categories: the one with fixed pruning ratio and with learned pruning ratio for each layer. For the first category, [13] proposed an iterative two-step channel pruning scheme, which first selects pruned channels with Lasso regression given a target pruning ratio, and then finetunes weights. In [18] and [11], the pruned channels are determined with L_1 and L_2 norm, respectively for each layer under the same pruning ratio. The approach proposed in [22] is based on first-order Taylor expansion for the loss function. To explore inter-channel dependency, many works explore second-order Taylor expansion, such as [26][24]. For the second category, pruning ratio for each layer is jointly optimized across all the layers. Sparsity learned during training with L_0 [21] regularization has been explored by several works. In [12] the pruning ratio for each layer is learned through reinforcement learning. The pruning approach in [4] relates redundancy of channels to the variational information bottleneck (VIB) [27], adding gates to the network and employing a suitable regularization term to train for sparsity. As mentioned in [27], the VIB-Net scheme has a certain advantage over Bayesian-type of compression [20], as the loss function does not require additional parameters to describe the priors. We also found this method to work better in practice. In our DJPQ scheme, we utilize this VIBNet pruning while optimizing for quantization jointly to achieve a flexible trade-off between the two. For an overview of structured pruning methods, please refer to [16].

To jointly optimize pruning and quantization, two research works have been proposed [35][28]. Unfortunately, neither of them support activation quantization. Thus, the resulting networks cannot be deployed efficiently on edge hardware. Recently, an Alternating Direction Method of Multipliers (ADMM)-based approach was proposed in [33]. It formulates the automated compression problem to a constrained optimization problem and solves it iteratively using ADMM. One key limitation is the lack of support for hardware friendliness. First, it is based on unstructured pruning and thus no performance benefit in typical hardware. Second, they use a non-uniform quantization which has the same issue. In addition, the iterative nature of ADMM imposes excessive training time for end-to-end optimization while our DPJQ method does not, since we train in only a single training pass.

3 Differentiable joint pruning and quantization

We propose a novel differentiable joint pruning and quantization (DJPQ) scheme for compressing DNNs. Due to practical hardware limitations, only uniform quantization and structured pruning are considered. We will first discuss the quantization and pruning method, and then introduce the evaluation metric, and finally go over joint optimization details.

3.1 Quantization with learnable mapping

It has been observed that weights and activations of pre-trained neural networks typically have bell-shaped distributions with long tails [9]. Consequently, uniform quantization is sub-optimal for such distributions. We use a non-linear function to map any weight input x to $\tilde{x}$. Let q_s and q_m be the minimum and maximum value to be mapped, where $0 < q_s < q_m$. Let $t > 0$ be the exponent controlling the shape of mapping (c.f., Appendix A). q_m and t are learnable parameters, and q_s is fixed to a small value. The non-linear mapping is defined as

$$\tilde{x} = \mathrm{sign}(x) \cdot \begin{cases} 0, & |x| < q_s \\ (|x| - q_s)^t, & q_s \le |x| \le q_m \\ (q_m - q_s)^t, & |x| > q_m \end{cases} \tag{1}$$

For activation quantization, we do not use non-linear mapping, i.e., for any input x, $\tilde{x}$ is derived from (1) with t fixed to 1. After mapping, a uniform quantization is applied to $\tilde{x}$. Let d be quantization step-size, and let x_q be the quantized value of x. The quantized version is given by

$$x_q = \mathrm{sign}(x) \cdot \begin{cases} 0, & |x| < q_s \\ d\lfloor \frac{(|x| - q_s)^t}{d} \rceil, & q_s \le |x| \le q_m \\ d\lfloor \frac{(q_m - q_s)^t}{d} \rceil, & |x| > q_m \end{cases} \tag{2}$$

where $\lfloor\cdot\rceil$ is the rounding operation. We note that the quantization grid for weights is symmetric and the bit-width b is given by

$$b = \log_2\lceil \frac{(q_m - q_s)^t}{d} + 1\rceil + 1. \tag{3}$$

For ReLU activations, we use a symmetric unsigned grid since the resulting values are always non-negative. Here, the activation bit-width b is given by

$$b = \log_2\lceil \frac{(q_m - q_s)^t}{d}\rceil. \tag{4}$$

Finally, we use the straight through estimation (STE) method [2] for back-propagating gradients through the quantizers, which [29] has shown to converge fast. The gradients of the quantizer output with respect to d, q_m and t are given by

$$\triangledown_d x_q = \begin{cases} \text{sign}(x)\left(\lfloor\frac{(|x|-q_s)^t}{d}\rceil - \frac{(|x|-q_s)^t}{d}\right), & q_s \le |x| \le q_m \\ \text{sign}(x)\left(\lfloor\frac{(q_m-q_s)^t}{d}\rceil - \frac{(q_m-q_s)^t}{d}\right), & |x| > q_m, \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\triangledown_{q_m} x_q = \begin{cases} 0, & |x| \le q_m \\ \text{sign}(x) t (q_m - q_s)^{t-1}, & \text{otherwise} \end{cases} \tag{6}$$

$$\triangledown_t x_q = \begin{cases} \text{sign}(x)(|x| - q_s)^t \log(|x| - q_s), & q_s \le |x| \le q_m \\ \text{sign}(x)(q_m - q_s)^t \log(q_m - q_s), & |x| > q_m \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

3.2 Structured pruning via VIBNet gates

Our structured pruning scheme is based on the variational information bottleneck (VIBNet) [4] approach. Specifically, we add multiplicative Gaussian gates to all channels in a layer and learn a variational posterior of the weight distribution aiming at minimizing the mutual information between current layer and next layer's outputs, while maximizing the mutual information between current layer and network outputs. The learned distributions are then used to determine which channels need to be pruned. Assuming the network has L layers, and that there are c_l output channels in the l-th layer, $l = 1, \cdots, L$. Let $\boldsymbol{h}_l \in \mathcal{R}^{c_l}$ be the output of l-th layer after activation. Let $\boldsymbol{y} \in \mathcal{Y}^{c_L}$ be the target output or data labels. Let $I(\boldsymbol{x};\boldsymbol{y})$ denote the mutual information between $\boldsymbol{x}$ and $\boldsymbol{y}$. The VIB loss function $\mathcal{L}_{\text{VIB}}$ is defined as

$$\mathcal{L}_{\text{VIB}} = \gamma \sum_{l=1}^{L} I\big(\boldsymbol{h}_l; \boldsymbol{h}_{l-1}\big) - I\big(\boldsymbol{h}_l; \boldsymbol{y}\big), \tag{8}$$

where $\gamma > 0$ is a scaling factor. It then follows that

$$\boldsymbol{h}_l = \boldsymbol{z}_l \odot \boldsymbol{f}_l\big(\boldsymbol{h}_{l-1}\big), \tag{9}$$

where $\boldsymbol{z}_l = \{z_{l,1}, \cdots, z_{l,c_l}\}$ is a vector of gates for the l-th layer. The $\odot$ represents the element-wise multiplication operator. $\boldsymbol{f}_l$ represents l-th layer mapping function, i.e., concatenation of a linear or convolutional transformation, batch normalization and some nonlinear activation. Let us assume that $\boldsymbol{z}_l$ follows a Gaussian distribution with mean $\boldsymbol{\mu}_l$ and variance $\boldsymbol{\sigma}_l^2$, which can be re-parameterized as

$$\boldsymbol{z}_l = \boldsymbol{\mu}_l + \boldsymbol{\epsilon}_l \odot \boldsymbol{\sigma}_l \tag{10}$$

where $\boldsymbol{\epsilon}_l \sim \mathcal{N}(0, I)$. It follows that the prior distribution of $\boldsymbol{h}_l$ conditioned on $\boldsymbol{h}_{l-1}$ is

$$p(\boldsymbol{h}_l|\boldsymbol{h}_{l-1}) \sim \mathcal{N}\big(\boldsymbol{\mu}_l \odot \boldsymbol{f}_l(\boldsymbol{h}_{l-1}), \mathrm{diag}[\boldsymbol{\sigma}_l^2 \odot \boldsymbol{f}_l(\boldsymbol{h}_{l-1})^2]\big) \tag{11}$$

We further assume that the prior distribution of $\boldsymbol{h}_l$ is also Gaussian

$$q(\boldsymbol{h}_l) \sim \mathcal{N}(0, \mathrm{diag}[\boldsymbol{\xi}_l]) \tag{12}$$

where $\boldsymbol{\xi}_l$ is a vector of variances chosen to minimize an upper bound of $\mathcal{L}_{\mathrm{VIB}}$ (c.f., (8)), as given in [4]

$$\tilde{\mathcal{L}}_{\mathrm{VIB}} \triangleq \mathrm{CE}(\boldsymbol{y}, \boldsymbol{h}_L) + \gamma \sum_{l=1}^{L} \sum_{i=1}^{c_l} \log\left(1 + \frac{\mu_{l,i}^2}{\sigma_{l,i}^2}\right). \tag{13}$$

In (13), $\mathrm{CE}(\boldsymbol{y}, \boldsymbol{h}_L)$ is the cross-entropy loss. The optimal $\boldsymbol{\xi}_l$ is then derived as

$$\boldsymbol{\xi}_l^* = (\boldsymbol{\mu}_l^2 + \boldsymbol{\sigma}_l^2)\mathbf{E}_{\boldsymbol{h}_{i-1} \sim p(\boldsymbol{h}_{i-1})}[\boldsymbol{f}_i(\boldsymbol{h}_{i-1})^2].$$

Next, we give the condition for pruning a channel. Let $\boldsymbol{\alpha}_l$ be defined as $\boldsymbol{\alpha}_l = \boldsymbol{\mu}_l^2/\boldsymbol{\sigma}_l^2$. If $\alpha_{li} < \alpha_{th}$, where α_{th} is a small pruning threshold, the i-th channel in the l-th layer is pruned. The pruning parameters $\{\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l\}$ are learned during training to minimize the loss function.

3.3 Evaluation metrics

Actual hardware measurement is generally a poor indicator for the usefulness of these methods. There are many possible target-devices, e.g. GPUs, TPUs, several phone-chips, IoT devices, FPGAs, and dedicated silicon, each with their own quirks and trade-offs. Efficiency of networks also greatly depends on specific kernel-implementations, and currently many devices do not have multiple bit-width kernels implemented to even compare on in a live setting. Therefore, we choose a more general metric to measure the compression performance, namely Bit-Operations (BOPs) count, which assumes computations can be done optimally in an ideal world. The BOPs metric has been used by several papers such as [19] and [3]. Our method can be easily modified to optimize for specific hardware by weighting the contributions of different settings.

We also report results on layerwise pruning ratio and mac counts, for purposes of comparison of the pruning methods. Let p_l be the pruning ratio of l-th layer output channels. We define P_l to be the layerwise pruning ratio, which is the ratio of the number of weights between the uncompressed and compressed models. It can be derived as

$$P_l = 1 - (1 - p_{l-1})(1 - p_l). \tag{14}$$

The l-th layer Multiply-And-Accumulate (MAC) operations is defined as follows. Let l-th layer output feature map have width, height and number of channels of $m_{w,l}$, $m_{h,l}$ and c_l, respectively. Let k_w and k_h be the kernel width and height. The MAC count is computed as

$$\text{MACs}_l \triangleq (1 - p_{l-1})c_{l-1} \cdot (1 - p_l)c_l \cdot m_{w,l} \cdot m_{h,l} \cdot k_w \cdot k_h. \tag{15}$$

The BOP count in the l-th layer BOPs_l is defined as

$$\text{BOPs}_l \triangleq \text{MACs}_l \cdot b_{w,l} \cdot b_{a,l-1}, \tag{16}$$

where $b_{w,l}$ and $b_{a,l}$ denote l-th layer weight and activation bit-width.

The BOP compression ratio is defined as the ratio between total BOPs of the uncompressed and compressed models. MAC compression ratio is similarly defined. Without loss of generality, we use the MAC compression ratio to measure only the pruning effect, and use the BOP compression ratio to measure the overall effect from pruning and quantization. As seen from (16), BOP count is a function of both channel pruning ratio p_l and bit-width $b_{w,l}$ and $b_{a,l}$, hence BOP compression ratio is a suitable metric to measure a DNN's overall compression.

3.4 Joint optimization of pruning and quantization

As described in Section 3.3, the BOP count is a function of channel pruning ratio p_l and bit-widths $b_{w,l}$ and $b_{a,l}$. As such, incorporating the BOP count in the loss function allows for joint optimization of pruning and quantization parameters. When combing the two methods for pruning and quantization, we need to define a loss function that combines both in a sensible fashion. So we define the DJPQ loss function as

$$\mathcal{L}_{\text{DJPQ}} \triangleq \tilde{\mathcal{L}}_{\text{VIB}} + \beta \sum_{l=1}^{L} \text{BOPs}_l \tag{17}$$

where $\tilde{\mathcal{L}}_{\text{VIB}}$ is the upper bound to the VIB loss given by (13), β is a scalar, and BOPs_l is defined in (16). To compute BOPs_l, $b_{w,l}$ and $b_{a,l}$ are given by (3) and (4), respectively, and $b_{a,0}$ denotes DNN's input bit-width. For hard pruning, p_l can be computed as

$$p_l = \frac{\sum_{i=1}^{c_l} \mathbf{1}\{\alpha_{l,i} < \alpha_{th}\}}{c_l} \tag{18}$$

Algorithm 1 Differentiable joint pruning and quantization

```
Input: A neural network with weight $\boldsymbol{w}_l$, $l \in [1, L]$; training data
Output: $b_{w,l}$, $b_{a,l}$, $P_l$, $l \in [1, L]$
Parameters: $\boldsymbol{w}_l, (\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l), (q_{m,wl}, d_{w,l}, t_{w,l}, q_{m,al}, d_{a,l})$, $l \in [1, L]$
Forward pass:
Anneal strength $\gamma$ and $\beta$ after each epoch
for $l \in \{1, \cdots, L\}$ do
    Draw samples $\boldsymbol{z}_l \sim \mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l^2)$ and multiply to the channel output
    Compute $p_l$ according to (19)
    Compute the layerwise pruning ratio $P_l$ according to (14)
    Compute bit-width $b_{w,l}$ and $b_{a,l}$ according to (3) and (4)
    if bit-restricted then
        Adjust $b_{w,l}$, $d_{w,l}$, $b_{a,l}$ and $d_{a,l}$ according to Algorithm 2
    Quantize $\boldsymbol{w}_l$ and $\boldsymbol{h}_l$
Compute $\mathcal{L}_{\text{DJPQ}}$ according to (17)

Backward pass:
for $l \in \{1, \cdots, L\}$ do
    Compute the gradients of $(d_{w,l}, d_{a,l})$, $(q_{m,wl}, q_{m,al})$, and $t_{w,l}$ according to (5), (6)
    and (7), respectively
    Compute the gradients of $\boldsymbol{\mu}_l$ and $\boldsymbol{\sigma}_l$
    Compute the gradients of $\boldsymbol{w}_l$
    Update $\boldsymbol{w}_l, (\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l), (q_{m,wl}, d_{w,l}, t_{w,l}, q_{m,al}, d_{a,l})$ with corresponding learning rate
```

where $\mathbf{1}\{\cdot\}$ is the indicator function. For soft pruning during training, the indicator function in (18) is relaxed by a sigmoid function $\sigma(\cdot)$, i.e.,

$$p_l = \frac{\sum_{i=1}^{c_l} \sigma(\frac{\alpha_{l,i} - \alpha_{th}}{\tau})}{c_l} \tag{19}$$

where τ is a temperature parameter. Note that soft pruning makes p_l, hence the BOPs$_l$, differentiable with respect to $\boldsymbol{\mu}_l$ and $\boldsymbol{\sigma}_l$.

The parameters of joint optimization of pruning and quantization include $\boldsymbol{w}_l$, $(\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l)$ and $(q_{m,wl}, d_{w,l}, t_{w,l}, q_{m,al}, d_{a,l})$ for $l = 1, \cdots, L$. In our experimentation we have found that a proper scaling of the learning rates for different pruning and quantization parameters plays an important role in achieving a good performance. Our experiments also showed that using the ADAM optimizer to automatically adapt the learning rates does not give as good a performance as using the SGD optimizer, provided that suitable scaling factors are set manually. Please refer to Appendix B.3 for detailed setting of the scaling factors and other hyper-parameters.

3.5 Power-of-two quantization

To further align this method with what is feasible in common hardware, in this section we extend DJPQ to the scenario where quantization bit-width b is restricted to power-of-two values, i.e., $b \in \{2, 4, 8, 16, 32\}$. As reflected in Alg. 1, this extension involves an added step where bit-widths are rounded to the nearest power-of-two representable value, and the quantizer stepsizes are updated accordingly without changing q_m, c.f. Alg. 2 for details.

Algorithm 2 Adjust bit-width to power of two integers

```
**Input**: stepsize $d_l$, max. range $q_{m,l}$, exponent $t_l$
**Output**: Adjusted bit-width $b'_l$ and stepsize $d'_l$
Compute $b_l$ according to (3) or (4)
Compute $s_l$: $s_l = \log_2 b_l$
Adjust $s_l$ to $s'_l$: $s'_l = \lfloor s_l \rceil$
Adjust $b_l$ to $b'_l$ with $s'_l$: $b'_l = 2^{s'_l}$
Adjust $d_l$ to $d'_l$ with $b'_l$: $d'_l = \frac{(q_m - q_s)^t}{2^{b'-1}-1}$ or $d'_l = \frac{(q_m - q_s)^t}{2^{b'}}$
```

This extension only involves a small overhead to the training phase. More specifically, while there is no change in the backward pass, in the forward pass the quantizer bit-width and stepsize are adjusted. The adjustment of the stepsize would result in larger variance in gradients, however, the scheme is still able to converge quickly in practice.

4 Experiments

We run experiments on different models for image classification, including VGG7 [17] on CIFAR10, ResNet18 [10] and MobileNetV2 [25] on ImageNet. Performance is measured by top-1 accuracy and the BOP count. The proposed DJPQ scheme is always applied to pretrained models. We compare with several other schemes from literature including LSQ [7], TWN [17], RQ [19], WAGE [32] and DQ [29]. We also conduct experiments for the two-stage compression approach, to see how much gain is achieved by co-optimizing pruning and quantization. Furthermore, we conduct experiments of DJPQ with power-of-two quantization, denoted as DJPQ-restrict. We also modify the DQ scheme to do restricted mixed-bit quantization to compare with DJPQ-restrict.

4.1 Comparison of DJPQ with quantization only schemes

CIFAR10 results For the VGG7 model on CIFAR10 classification, DJPQ performance along with its baseline floating-point model are provided in Table 1. Fig. 1 shows the weight and activation bit-widths for each layer. The pruning ratio P_l for each layer is given in Fig. 2. Compared to the uncompressed model, we see that DJPQ reduces the amount of BOPs by 210x with less than a 1.5% accuracy drop. DJPQ is able to achieve a larger compression ratio compared to the other schemes. Comparing e.g. to DQ, which also learns mixed precision, DJPQ has a very similar BOPs reduction.

Table 1. VGG7 results on CIFAR10. If weight and activation bit-width are fixed, they are represented in the format of weight/activation bit-width in the table. If weight and activation bit-width have different values in different layers, they are denoted as 'mixed'. Baseline is the floating point model. 'BOP comp. ratio' denotes the BOP compression ratio defined in Section 3.3. 'DJPQ-restrict' denotes DJPQ with power-of-two bit-restricted quantization as presented in Section 3.5.

	bit-width	Test Acc.	MACs(G)	BOPs(G)	BOP comp. ratio
Baseline	32/32	93.0%	0.613	629	–
TWN [17]	2/32	92.56%	0.613	39.23	16.03
RQ [19]	8/8	93.30%	0.613	39.23	16.03
RQ [19]	4/4	92.04%	0.613	9.81	64.12
WAGE [32]	2/8	93.22%	0.613	9.81	64.12
DQ[1] [29]	mixed	91.59%	0.613	3.03	207.59
DQ-restrict [2] [29]	mixed	91.59%	0.613	3.40	185.00
DJPQ	mixed	**91.54%**	0.367	2.99	**210.37**
DJPQ-restrict	mixed	**91.43%**	0.372	2.92	**215.41**

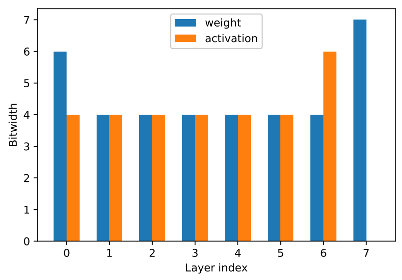


Fig. 1. Bit-width of weights and activations for VGG7 with DJPQ scheme

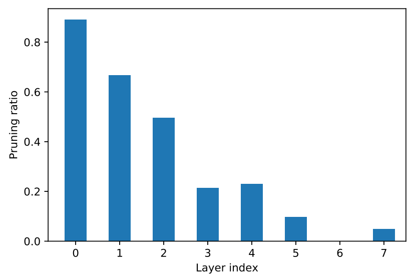


Fig. 2. Pruning ratio for VGG7 with DJPQ scheme

ImageNet results For experiments on the ImageNet dataset, we applied our DJPQ scheme to ResNet18 and MobileNetV2. Table 2 provides a comparison of DJPQ with state-of-art and other works. The learned bit-width and pruning ratio distributions for DJPQ with 69.27% accuracy are shown in Figs. 3 and 4, respectively. From the table we see that DJPQ achieves a 53x BOPs reduction while the top-1 accuracy only drops by 0.47%. The results showed a smooth trade-off achieved by DJPQ between accuracy and BOPs reduction. Compared with other fixed-bit quantization schemes, DJPQ achieves a significantly larger BOPs reduction. Particularly compared with DQ, DJPQ achieves around a 25% reduction in BOP counts (40.71G vs 30.87G BOPs) with a 0.3% accuracy improvement. A more detailed comparison of DJPQ and DQ has been given in Appendix B.1, which includes weight and activation bit-width distribution, along with pruning ratio distribution. Through the comparison, we show that DJPQ can flexibly trade-off pruning and activation to achieve a larger compression ratio.

Table 2. Comparison of ResNet18 compression results on ImageNet. Baseline is the floating point model. 'DJPQ-restrict' denotes DJPQ with bit-restricted quantization.

	Bit-width	Test Acc.	MACs(G)	BOPs(G)	BOP comp. ratio
Baseline	32/32	69.74%	1.81	1853.44	–
LSQ[3] [7]	3/3	70.00%	1.81	136.63	13.56
RQ [19]	8/8	69.97%	1.81	115.84	16.00
DFQ [23]	4/8	69.3%	1.81	57.92	32.00
SR+DR [8]	8/8	68.17%	1.81	115.84	16.00
UNIQ [1]	4/8	67.02%	1.81	57.92	32.00
DFQ [23]	4/4	65.80%	1.81	28.96	64.00
TWN [17]	2/32	61.80%	1.81	115.84	16.00
RQ [19]	4/4	61.52%	1.81	28.96	64.00
DQ[1] [29]	mixed	68.49%	1.81	40.71	45.53
DQ-restrict [2] [29]	mixed	68.49%	1.81	58.68	31.59
VIBNet [4]+fixed quant.	8/8	69.24%	1.36	87.04	21.29
VIBNet [4]+DQ[1] [29]	mixed	68.52%	1.36	39.83	46.53
DJPQ	mixed	**69.27%**	1.39	35.01	**52.94**
DJPQ	mixed	68.80%	1.39	30.87	60.04
DJPQ-restrict	mixed	**69.12%**	1.46	35.45	**52.28**

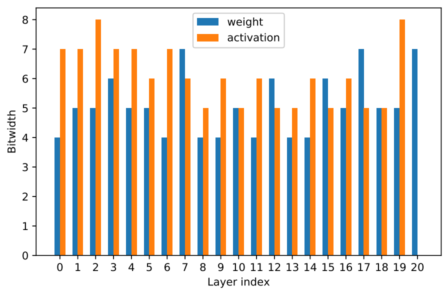


Fig. 3. Bit-width for ResNet18 on ImageNet with DJPQ scheme

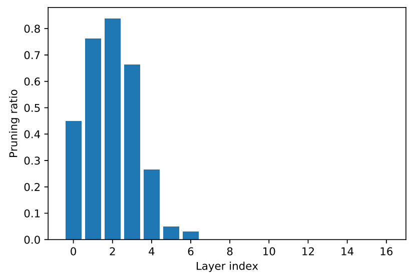


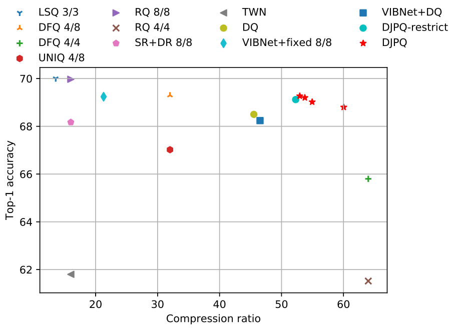
Fig. 4. Pruning ratio for ResNet18 on ImageNet with DJPQ scheme

MobileNetV2 has been shown to be very sensitive to quantization [23]. Despite this, we show in Table 3 that the DJPQ scheme is able to compress MobileNetV2 with a large compression ratio. The optimized bit-width and pruning ratio distributions are provided in Appendix B.2. It is observed that DJPQ achieves a 43x BOPs reduction within 2.4% accuracy drop. Compared with DQ, DJPQ achieves around 25% reduction in BOP counts over DQ on MobileNetV2

(175.24G vs 132.28G BOPs) with 0.5% accuracy improvement. Comparisons of BOPs for different schemes are plotted in Figs. 5 and 6 for ResNet18 and MobileNetV2, respectively. DJPQ provides superior results for ResNet18 and MobileNetV2, and can be easily extended to other architectures.

Table 3. Comparison of MobileNetV2 compression results on ImageNet. Baseline is the uncompressed floating point model.

	Bit-width	Test Acc.	MACs(G)	BOPs(G)	BOP comp. ratio
Baseline	32/32	71.72%	5.55	5682.32	–
SR+DR [8]	8/8	61.30%	5.55	355.14	16.00
DFQ [23]	8/8	70.43%	5.55	355.14	16.00
RQ [19]	8/8	71.43%	5.55	355.14	16.00
RQ [19]	6/6	68.02%	5.55	199.80	28.44
UNIQ [1]	4/8	66.00%	5.55	177.57	32.00
DQ[1] [29]	mixed	68.81%	4.81	175.24	32.43
DJPQ	mixed	**69.30%**	4.76	132.28	**42.96**



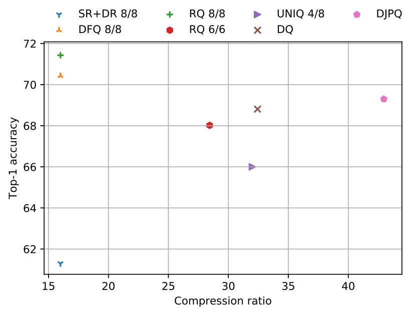


Fig. 5. Comparison of BOPs reduction for ResNet18 on ImageNet

Fig. 6. Comparison of BOPs reduction for MobileNetV2 on ImageNet

4.2 Comparison of DJPQ with two-stage optimization

In this section we provide a comparison of DJPQ and the two-stage approach - pruning first, then quantizing, and show that DJPQ outperforms both in BOPs

[1] For a fair comparison, we replaced the memory regularization term in DQ [29] with BOPs regularization.

[2] DQ-restrict [29] refers to the scheme where bit-width learned by DQ are upper rounded to the nearest power-of-two integers.

[3] LSQ [23] does not quantize the first and last layers.

reduction and accuracy. To ensure a fair comparison, pruning and quantization are optimized independently in the two-stage approach. We have done extensive experiments to find a pruning ratio that results in high accuracy. The two-stage results in Table 2 are derived by first pruning a model with VIBNet gates to a 1.33x MAC compression ratio. The accuracy of the pruned model is 69.54%. Then the pruned model is quantized with both fixed-bit and mixed-precision quantization. For fixed 8-bit quantization, the two-stage scheme obtains a 21.29x BOP reduction with 69.24% accuracy. For the VIBNet+DQ approach, we achieve a 46.53x BOP reduction with 68.52% accuracy.

By comparing the resulting MAC counts of the compressed model of DJPQ with the two-stage approach, we see a very close MAC compression ratio from pruning between the two schemes (1.24x vs 1.33x). They also have comparable BOP counts, however, DJPQ achieves 0.75% higher accuracy (69.27% vs 68.52%) than the two-stage scheme. The results provide good evidence that even under similar pruning ratio and BOP reduction, DJPQ is able to achieve a higher accuracy. This is likely due to the pruned channel distribution of DJPQ being dynamically adapted and optimized jointly with quantization resulting in a higher accuracy.

4.3 Comparison of DJPQ with others under bit restriction

We further run experiments of the DJPQ scheme for power-of-two bit-restricted quantization. For VGG7 on CIFAR10, the DJPQ result with bit-restricted quantization is provided in Table 1 named 'DJPQ-restrict'. With DJPQ compression, VGG7 is compressed by 215x with a 91.43% accuracy. The degradation of accuracy compared to DJPQ without bit restriction is negligible, showing that our scheme also works well for power-of-two restricted bit-widths. For ResNet18 on ImageNet, Table 2 shows DJPQ-restrict performance. DJPQ-restrict is able to compress ResNet18 by 53x with only a 0.5% accuracy drop. We also provide a comparison of DJPQ and DQ both with bit restrictions, denoted as 'DJPQ-restrict' and 'DQ-restrict', respectively. Compared with DQ, DQ-restrict has a much smaller compression ratio. It shows that the performance of compression would be largely degraded if naively rounding the trained quantization bits. DJPQ-restrict has significant compression ratio gain over DQ-restrict results. Comparing DJPQ-restrict to other schemes with fixed 2/4/8-bit quantization, it is clear that DJPQ-restrict has the highest compression ratio.

4.4 Analysis of learned distributions

For ResNet18, the learned bit-width and pruning ratio distributions for DJPQ with 69.27% accuracy are shown in Figs. 3 and 4, respectively. It is observed that pruning occurs more frequently at earlier layers. The pruning ratio can be very large, indicating heavy over-parameterization in that layer. Layers $\{7,12,17\}$ are residual connections. From Fig. 3 we see that all the three residual connections require larger bits than their corresponding regular branches. Regarding to the distribution of t in the nonlinear mapping, it is observed that for layers with

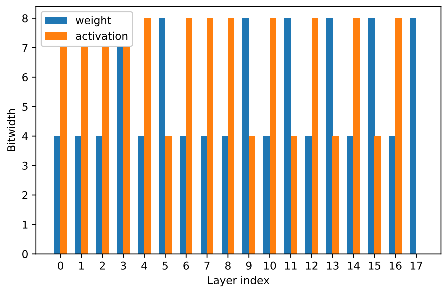


Fig. 7. bit-width for ResNet18 on ImageNet. Bits are restricted to power of 2.

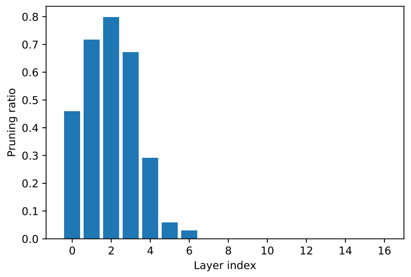


Fig. 8. Pruning ratio for ResNet18 on ImageNet. Bits are restricted to power of 2.

heavy pruning, t is generally smaller and $t < 1$; for layers with no pruning, t is close to 1. This gives a good reflection of interaction between pruning and quantization. For MobileNetV2, we found that point-wise convolution layers require larger bit-width than depth-wise ones, indicating that point-wise convolutional layers are more sensitive than depth-wise ones.

5 Conclusion

We proposed a differentiable joint pruning and quantization (DJPQ) scheme that optimizes bit-width and pruning ratio simultaneously. The scheme integrates variational information bottleneck, structured pruning and mixed-precision quantization, achieving a flexible trade-off between sparsity and bit precision. We show that DJPQ is able to achieve larger bit-operations (BOPs) reduction over conventional two-stage compression while maintaining the state-of-art performance. Specifically, DJPQ achieves 53x BOPs reduction in ResNet18 and 43x reduction in MobileNetV2 on ImageNet. We further extend DJPQ to support power-of-two bit-restricted quantization with a small overhead. The extended scheme is able to reduce BOPs by 52x on ResNet18 with almost no accuracy loss.

Acknowledgments

We would like to thank Jilei Hou and Joseph Soriaga for consistent support, and thank Jinwon Lee, Kambiz Azarian and Nojun Kwak for their great help in revising this paper and providing valuable feedback.

References

1. Baskin, C., Schwartz, E., Zheltonozhskii, E., Liss, N., Giryes, R., Bronstein, A.M., Mendelson, A.: UNIQ: uniform noise injection for the quantization of neural networks. CoRR **abs/1804.10969** (2018), http://arxiv.org/abs/1804.10969
2. Bengio, Y., Léonard, N., Courville, A.C.: Estimating or propagating gradients through stochastic neurons for conditional computation. CoRR **abs/1308.3432** (2013), http://arxiv.org/abs/1308.3432
3. Bethge, J., Bartz, C., Yang, H., Chen, Y., Meinel, C.: MeliusNet: Can binary neural networks achieve mobilenet-level accuracy? arXiv:2001.05936 (2020)
4. Dai, B., Zhu, C., Guo, B., Wipf, D.: Compressing neural networks using the variational information bottleneck. In: International Conference on Machine Learning. pp. 1135–1144 (2018)
5. Dong, Z., Yao, Z., Cai, Y., Arfeen, D., Gholami, A., Mahoney, M.W., Keutzer, K.: HAWQ-V2: hessian aware trace-weighted quantization of neural networks. arXiv:1911.03852 (2019)
6. Dong, Z., Yao, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: HAWQ: Hessian aware quantization of neural networks with mixed-precision. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 293–302 (2019)
7. Esser, S.K., McKinstry, J.L., Bablani, D., Appuswamy, R., Modha, D.S.: Learned step size quantization. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=rkgO66VKDS
8. Gysel, P., Pimentel, J., Motamedi, M., Ghiasi, S.: Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks. IEEE transactions on neural networks and learning systems **29**(11), 5784–5789 (2018)
9. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding. In: 4th International Conference on Learning Representations (2016)
10. 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)
11. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. arXiv:1808.06866 (2018)
12. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.J., Han, S.: Amc: AutoML for model compression and acceleration on mobile devices. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 784–800 (2018)
13. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1389–1397 (2017)
14. Ignatov, A., Timofte, R., Kulik, A., Yang, S., Wang, K., Baum, F., Wu, M., Xu, L., Van Gool, L.: AI benchmark: All about deep learning on smartphones in 2019. arXiv:1910.06663 (2019)
15. Krishnamoorthi, R.: Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv:1806.08342 (2018)
16. Kuzmin, A., Nagel, M., Pitre, S., Pendyam, S., Blankevoort, T., Welling, M.: Taxonomy and evaluation of structured compression of convolutional neural networks. arXiv:1912.09802 (2019)
17. Li, F., Zhang, B., Liu, B.: Ternary weight networks. arXiv:1605.04711 (2016)
18. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient ConvNets. In: International Conference on Learning Representations (2017), https://openreview.net/pdf?id=rJqFGTslg

19. 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
20. Louizos, C., Ullrich, K., Welling, M.: Bayesian compression for deep learning. In: Advances in Neural Information Processing Systems. pp. 3288–3298 (2017)
21. Louizos, C., Welling, M., Kingma, D.P.: Learning sparse neural networks through L_0 regularization. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=H1Y8hhg0b
22. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. In: International Conference on Learning Representations (2017), https://openreview.net/pdf?id=SJGCiw5gl
23. Nagel, M., Baalen, M.v., Blankevoort, T., Welling, M.: Data-free quantization through weight equalization and bias correction. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1325–1334 (2019)
24. Peng, H., Wu, J., Chen, S., Huang, J.: Collaborative channel pruning for deep networks. In: International Conference on Machine Learning. pp. 5113–5122 (2019)
25. 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)
26. Theis, L., Korshunova, I., Tejani, A., Huszár, F.: Faster gaze prediction with dense networks and fisher pruning. arXiv:1801.05787 (2018)
27. Tishby, N., Pereira, F., Bialek, W.: The information bottleneck method. Proceedings of the 37th Allerton Conference on Communication, Control and Computation **49** (07 2001)
28. Tung, F., Mori, G.: CLIP-Q: Deep network compression learning by in-parallel pruning-quantization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7873–7882 (2018)
29. Uhlich, S., Mauch, L., Yoshiyama, K., Cardinaux, F., García, J.A., Tiedemann, S., Kemp, T., Nakamura, A.: Differentiable quantization of deep neural networks. CoRR **abs/1905.11452** (2019), http://arxiv.org/abs/1905.11452
30. 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)
31. Wu, H., Judd, P., Zhang, X., Isaev, M., Micikevicius, P.: Integer quantization for deep learning inference: Principles and empirical evaluation. arXiv:2004.09602 (2020)
32. Wu, S., Li, G., Chen, F., Shi, L.: Training and inference with integers in deep neural networks. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=HJGXzmspb
33. Yang, H., Gui, S., Zhu, Y., Liu, J.: Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2178–2188 (2020)
34. Yazdanbakhsh, A., Elthakeb, A.T., Pilligundla, P., Mireshghallah, F., Esmaeilzadeh, H.: ReLeQ: An automatic reinforcement learning approach for deep quantization of neural networks. arXiv:1811.01704 (2018)
35. Ye, S., Zhang, T., Zhang, K., Li, J., Xie, J., Liang, Y., Liu, S., Lin, X., Wang, Y.: A unified framework of DNN weight pruning and weight clustering/quantization using ADMM. arXiv:1811.01907 (2018)

A Quantization scheme in DJPQ

Fig. 9 illustrates the proposed quantization scheme. First, a non-linear function is applied to map any weight input x to $\tilde{x}$ (shown in blue curve). Then a uniform quantization is applied to $\tilde{x}$. The quantized value x_q is shown in red.

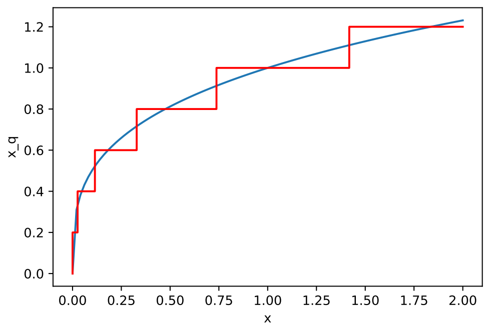


Fig. 9. Illustration of quantization scheme. The blue curve gives the nonlinear mapping function. The red curve corresponds to the quantization value.

B Experimental details

B.1 Comparison of DJPQ with DQ

To show that joint optimization of pruning and quantization outperforms quantization only scheme such as DQ, we plot in Fig. 10 a comparison of weight and activation bit-width for DQ and DJPQ. The results are for ResNet18 on ImageNet. We plot the pruning ratio curve of DJPQ in both figures to better show the pruning effect in the joint optimization scheme. As seen in the figure, there is no big difference between weight bit-width for DQ and DJPQ. However, the difference between activation bit-width for the two schemes is significant. Layer 0 to 6 in DJPQ has much larger activation bit-width than those in DQ, while those layers correspond to high pruning ratios in DJPQ. It provides a clear evidence that pruning and quantization can well tradeoff between each other in DJPQ, resulting in lower redundancy in the compressed model.

B.2 DJPQ results for MobileNetV2

Figs. 11 and 12 show the optimized bit-width and pruning ratio distributions, respectively. It is observed that earlier layers tend to have larger pruning ratios than the following layers. And for many layers in MobileNetV2, DJPQ is able to quantize the layers into to small bit-width.

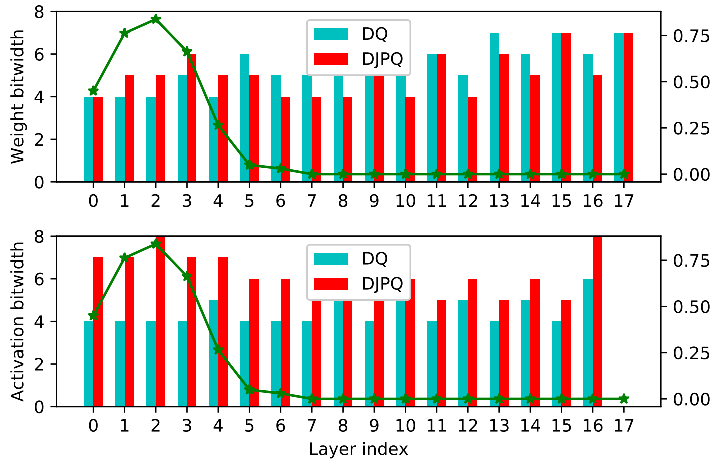


Fig. 10. Comparison of bit-width distributions of DQ and DJPQ for ResNet18 on ImageNet. The top figure plots the weight bit-width for each layer, while the bottom plots the corresponding activation bit-width. The green curve is the pruning ratio of DJPQ in each layer .

B.3 Experimental setup

The input for all experiments are uniformly quantized to 8 bits. For the two-stage scheme including 'VIBNet+fixed quant.' and 'VIBNet+DQ', VIBNet pruning is firstly optimized at learning rate 1e-3 with SGD, with a pruning learning rate scaling of 5. The strength γ is set to 5e-6. In DQ for the pruned model, β is chosen to 1e-11 and the learning rate is 5e-4. The learning rate scaling for quantization is 0.05. All the pruning threshold α_{th} is chosen to 1e-3. The number of epochs is 20 for each of the stage.

For DJPQ experiments on VGG7, the learning rate is set to 1e-3 with an ADAM optimizer. The initial bit-width is 6. The strength γ and β are 1e-6 and 1e-9, respectively. The scaling of learning rate for pruning and quantization is 10 and 0.05, respectively. For DJPQ on ResNet18, we chose a learning rate 1e-3 with SGD optimization. The initial bit-width for weights and activations are 6 and 8. The scaling of learning rate for pruning and quantization are 5 and 0.05, respectively. The strength γ and β are 1e-5 and 1e-10. For DJPQ on MobileNetV2, the learning rate is set to 1e-4 for SGD optimization. The initial bit-width is 8. The strength γ and β are set to 1e-8 and 1e-11. The learning rate scaling for pruning and quantization are selected to be 1 and 0.005, respectively.

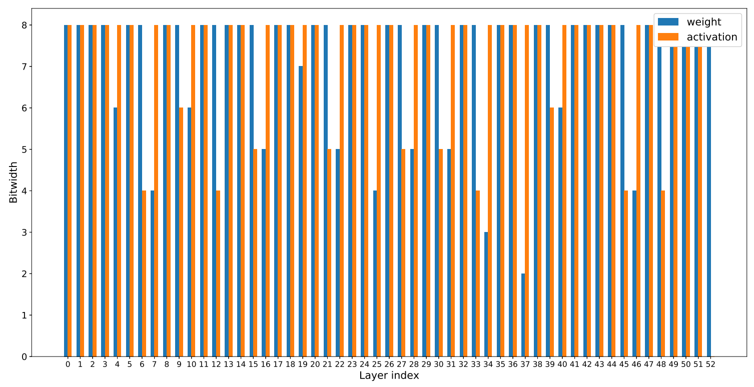


Fig. 11. Bit-width for MobileNetV2 on ImageNet with DJPQ scheme

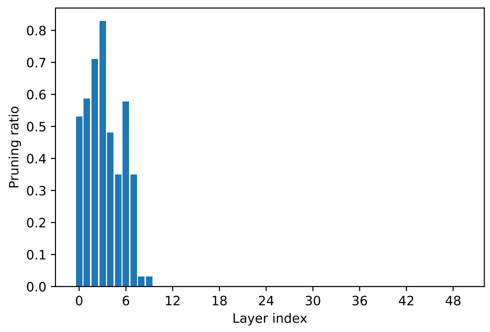


Fig. 12. Pruning ratio for MobileNetV2 on ImageNet with DJPQ scheme