Appendix for Retraining-free Model Quantization via One-Shot Weight-Coupling Learning

Chen Tang[1*] Yuan Meng[1*] Jiacheng Jiang[1] Shuzhao Xie[1] Rongwei Lu[1]
Xinzhu Ma[2] Zhi Wang[1†] Wenwu Zhu[1†]
[1]Tsinghua University [2]The Chinese University of Hong Kong

A. Experimental Setups

Dataset. We conduct experiments on three benchmarks, *i.e.* classification on ImageNet, CIFAR-100, and Pets, for classification task. Specifically, we pretrain the model on ImageNet and verify the transfer learning performance on CIFAR and Pets. For ImageNet classification, we use the basic data augmentation methods during training, *i.e.* the input images are randomly cropped to 224×224 with the horizontal flip.
Model. We conduct the experiments on three representative lightweight models including the ResNet18 [3], MobileNetV2 [5], and EfficientNetLite-B0 [6].
Hyper-parameters. The bit-width candidates are set to $\{2, 3, 4, 5, 6\}$ for both weight and activation quantization, we additionally remove 2-bits for Depthwise convolutions since they are already compact. Following the previous practices [1, 7, 9], the first and last layers are fixed to 8 bits. We set $\lambda = 1.5$ for searching.

For training, we adopt the SGD optimizer with cosine learning rate scheduler, the initial learning rate is set to 0.04 and the first 5 epochs are used as warm-up. We train the model for overall 150 epochs. The hyper-parameter ϵ in Eq. (6) is set to 0.25, and Q in Eq. (8) is set to 6, the weight-decay is set to 2.5×10^{-5} [1, 7, 8]. We randomly sample $K = 3$ mixed-precision policies at each training step. Furthermore, to ensure an unbiased large output set $\{\mathbf{O}_l^H\}_{l=0}^{L-1}$ is used for Information Distortion Mitigation (Sec. 3.4), we additionally sample a policy where each layer is the maximum bit-width. The hyper-parameter δ for Information Distortion Mitigation loss is set to $1e-2$. We use the LSQ quantizer [1], we adopt Adam to optimize the scaling factors of these quantizers, the learning rate is set to $1e-5$ without weight-decay.

B. Implementation Details

B.1. Bit-width Policy Sampling

According to Sec. 3.1, we approximate the expectation term over the whole search space by Monte-Carlo sampling, the forward propagation is defined as follows:

$$\arg\min_{\mathbf{W}} \quad \mathbb{E}_{\mathcal{S}\sim\mathcal{A}}\left[\mathcal{L}(f(\mathbf{x};\mathcal{S},\hat{\mathbf{W}}^{(\mathcal{S})}),\mathbf{y})\right] \approx \arg\min_{\mathbf{W}} \frac{1}{K}\sum_{\mathcal{S}_k\sim\mathrm{U}(\mathcal{A})}^{K}\left[\mathcal{L}(f(\mathbf{x};\mathcal{S}_k,\hat{\mathbf{W}}^{(\mathcal{S}_k)}),\mathbf{y})\right], \tag{A}$$

where $\mathcal{A}$ is the search space, $\mathcal{S}$ is the mixed-precision quantization policy (*i.e.,* the bit-width of each layer), that is, the joint gradients are defined as follows:

$$\nabla_{\mathbf{W}_t}\mathbb{E}_{\mathcal{S}\sim\mathcal{A}}\left[\mathcal{L}(f(\mathbf{x};\mathcal{S},\hat{\mathbf{W}}_t^{(\mathcal{S})}),\mathbf{y})\right] = \frac{1}{K}\sum_{\mathcal{S}_k\sim\mathrm{U}(\mathcal{A})}^{K}\left[\nabla_{\mathbf{W}_t}\left(\mathcal{L}(f(\mathbf{x};\mathcal{S}_k,\hat{\mathbf{W}}_t^{(\mathcal{S}_k)}),\mathbf{y})\right)\right]. \tag{B}$$

We adopt random sampling from a uniform distribution of all possible mixed-precision quantization policies, thus $\hat{\mathbf{W}}_t^{(\mathcal{S}_k)}$ is the parameters of uniformly sampled policy $\mathcal{S}_k$.

*: Equal contributions †: Corresponding authors Z. Wang is also with TBSI.

Algorithm 1: Bit-width Policy Sampling and Dynamic Training

```
Input: Full-precision Network $f$; Bit-list; Learning rate $\rho$; Bit-width freezing rate $\mathcal{K}$; Number of sampled policies $K$; Profiling window $M$.
Output: Retraining-free Weight-sharing Model.
1  Initialize the quantizer;
2  for $t$=1,...,$T$ do
3      Get batch of input data $\mathbf{x}$ and label $\mathbf{y}$;
4      if $t$ % $M$ == 0 then
           /* Profile the unstable weights $\hat{\Delta}\mathbf{W}^{\text{unstable}}$ and freeze the corresponding bit-widths according to Eq. (6) and Eq. (7). */
5          $\Omega \leftarrow$ TopKToFreeze($\hat{\Delta}\mathbf{W}^{\text{unstable}}; \mathcal{K}$);
6      end
7      Clear gradients of weights, optimizer.zero_grad();
8      $L = length(layers)$;
9      for $k$=0,...,$K$-1 do
           /* Sample $K$ mixed-precision policies and compute the gradients. */
10         Sample $k$-th layer-wise policy from the bit-width list to construct $\mathcal{S}_k = \{(\text{random.choice(weight\_list)}, \text{random.choice(act\_list)})\}_{l=0}^{L-1}$;
11         Switch $f$'s each layer to bit-width policy $\mathcal{S}_k$;
12         Get the quantized parameters $\hat{\mathbf{W}}^{(\mathcal{S}_k)}$ of $k$-th policy $\mathcal{S}_k$ from the shared latent weights $\mathbf{W}$;
13         Compute outputs under $k$-th policy $\hat{\mathbf{y}} = f(\mathbf{x}; \mathcal{S}_k, \mathbf{W}^{(\mathcal{S}_k)})$ ;
           /* Compute Cross-Entropy loss $\mathcal{L}_{\text{CE}} = \text{CE}(\mathbf{y}, \hat{\mathbf{y}})$, Quantization Error Minimization loss $\mathcal{L}_{\text{QE}}$ (Eq. (C)), Information Distortion Mitigation Loss $\mathcal{L}_{\text{IDM}}$. */
           /* $\kappa$ and $\delta$ are the hyper-parameters for weighting the loss. */
14         Loss $= \mathcal{L}_{\text{CE}} + \kappa\mathcal{L}_{\text{QE}} + \delta\mathcal{L}_{\text{IDM}}$;
15         Compute gradients through BP, $\nabla_{\mathbf{W}^{(\mathcal{S}_k)}} = \text{Loss}.backward()$;
16     end
       /* Use joint gradients to update the whole network. */
17     Aggregate above gradients to get $\mathbf{g}_t$ for updating the latent weights $\mathbf{W}$;
18     Update $f$'s parameters $\mathbf{W}_t = \mathbf{W}_{t-1} - \rho * \mathbf{g}_t$
19 end
```

In summary, at training step t, we first compute the gradients of K ramdomly sampled policies, and we aggregate the above gradient to update the parameters of whole network. During this sampling process, the dynamic training is incorporated. The pseudocode is shown in Alg. 1.

B.2. Optimization

Fairness Regularization. We disable the weight-decay for current smallest bit-width for ensuring proper regularization. Specifically, if a layer samples the current smallest bit-width, we add an additional negative weighted L2 term to the latent weights that fall within the clipping range of that smallest bitwidth, with the same value as the weight-decay term.

Quantization Error Minimization. We also explicitly optimize the distance between full-precision latent weights $\mathbf{W}$ and the quantized weights $\hat{\mathbf{W}}$ to reduce the quantization error as follows:

$$\mathcal{L}_{\text{QE}} = \frac{1}{K}\sum_{k=0}^{K-1}||\hat{\mathbf{W}}^{(\mathcal{S}_k)} - \mathbf{W}||_2 = \frac{1}{K}\sum_{k=0}^{K-1}\sum_{l=0}^{L-1}||Q_{b_{l,\text{w}}^{(k)} \in \mathcal{S}_k}(\mathbf{W}_l; \gamma) - \mathbf{W}_l||_2, \tag{C}$$

where K represents the number of sampled policies in each training iteration, L is the number of layers, and $Q(\cdot;\cdot)$ is the quantizer. More details about the notations please refer to Sec. 3.1.

B.3. Training Stability Estimation

The reasons for using distance between full-precision (FP) and quantized weights for estimating the training stability in Eq. (6) are as follows: (i) The distance depicts the rounding errors–large rounding errors in weight-sharing mean the weights are more vulnerable. For example, suggest an FP weight is 0.3999, and the two neighboring quantized weights of a certain

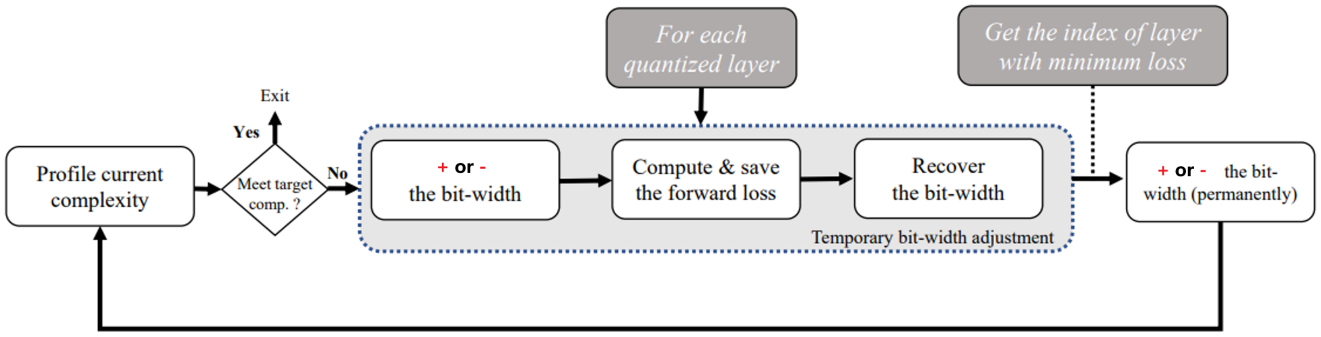
For each quantized layer
Get the index of layer with minimum loss
Exit
Yes
Profile current complexity
Meet target comp. ?
No
+ or - the bit-width
Compute & save the forward loss
Recover the bit-width
Temporary bit-width adjustment
+ or - the bit-width (permanently)

Figure A. Illustration of our bidirectional greedy search method.

bit-width are 0.0 and 0.8 (i.e., quantization bound is 0.4), meaning very small perturbations induced by other bit-widths on full-precision weight could make it large than 0.4 and cause the quantized weight change from 0.0 to next value 0.8 and then go back to 0.0, causing training instability (shown in Fig. 1); (*ii*) recent studies [2, 4] find that the large quantization error will disturb the quantization-aware training.

C. Bidirectional Greedy Search

We show the illustration of our search framework in Fig. A.

References

[1] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned Step Size quantization. In *Proc. of ICLR*, 2020. 1

[2] Tiantian Han, Dong Li, Ji Liu, Lu Tian, and Yi Shan. Improving low-precision network quantization via bin regularization. In *Proc. of ICCV*, 2021. 3

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proc. of CVPR*, 2016. 1

[4] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *Proc. of ICML*, 2022. 3

[5] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proc. of CVPR*, 2018. 1

[6] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proc. of ICML*, 2019. 1

[7] Chen Tang, Kai Ouyang, Zhi Wang, Yifei Zhu, Wen Ji, Yaowei Wang, and Wenwu Zhu. Mixed-precision Neural Network Quantization via Learned Layer-wise Importance. In *Proc. of ECCV*, 2022. 1

[8] Chen Tang, Kai Ouyang, Zenghao Chai, Yunpeng Bai, Yuan Meng, Zhi Wang, and Wenwu Zhu. Seam: Searching transferable mixed-precision quantization policy through large margin regularization. In *Proc. of ACM MM*, 2023. 1

[9] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware Automated Quantization With Mixed Precision. In *Proc. of CVPR*, 2019. 1