Bit-shrinking: Limiting Instantaneous Sharpness for Improving Post-training Quantization

Chen Lin[1] Bo Peng[1] Zheyang Li[1] Wenming Tan[1] Ye Ren[1] Jun Xiao[2] Shiliang Pu[1]
[1] Hikvision Research Institute [2]Zhe Jiang University
linchen7@hikvision.com pengbo7@hikvision.com lizhengyang@hikvision.com tanwenming@hikvision.com
renye@hikvision.com junxiao@zju.edu.com pushingliang@hikvision.com

Abstract

Post-training quantization (PTQ) is an effective compression method to reduce the model size and computational cost. However, quantizing a model into a low-bit one, e.g., lower than 4, is difficult and often results in non-negligible performance degradation. To address this, we investigate the loss landscapes of quantized networks with various bit-widths. We show that the network with more ragged loss surface, is more easily trapped into bad local minima, which mostly appears in low-bit quantization. A deeper analysis indicates, the ragged surface is caused by the injection of excessive quantization noise. To this end, we detach a sharpness term from the loss which reflects the impact of quantization noise. To smooth the rugged loss surface, we propose to limit the sharpness term small and stable during optimization. Instead of directly optimizing the target bit network, we design a self-adapted shrinking scheduler for the bit-width in continuous domain from high bit-width to the target by limiting the increasing sharpness term within a proper range. It can be viewed as iteratively adding small "instant" quantization noise and adjusting the network to eliminate its impact. Widely experiments including classification and detection tasks demonstrate the effectiveness of the Bit-shrinking strategy in PTQ. On the Vision Transformer models, our INT8 and INT6 models drop within 0.5% and 1.5% Top-1 accuracy, respectively. On the traditional CNN networks, our INT4 quantized models drop within 1.3% and 3.5% Top-1 accuracy on ResNet18 and MobileNetV2 without fine-tuning, which achieves the state-of-the-art performance.

1. Introduction

In recent years, network compression, such as Knowledge distillation [16], Pruning [9, 17, 38], and Quantization [30, 34, 35, 45] are rapidly developing to achieve efficient inference for Deep Neural Networks (DNNs) both at edge devices and in the clouds. Among them, quantization is a promising as well as hardware-friendly approach. It reduces the computation complexity and memory footprint by representing weights and activations with low-bit integers.

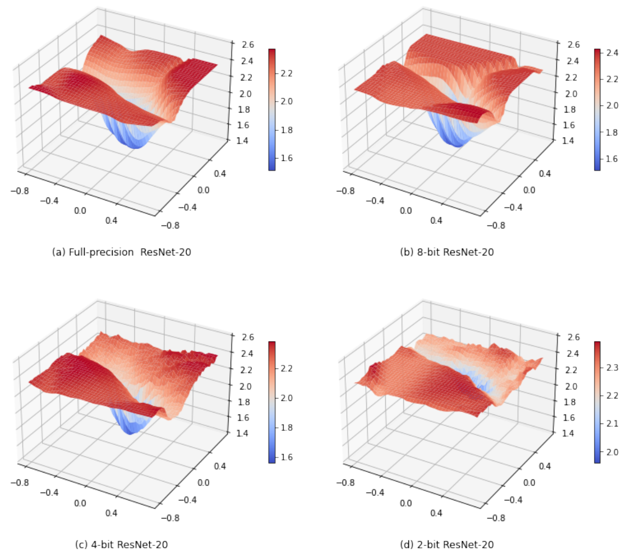

Figure 1. The loss landscapes of the full-precision and various bits ResNet-20 on CIFAR-100. We plot the loss landscapes using the visualization methods in [18]. The landscape of the lower-bit quantized network is more ragged, and is more easily trapped into bad local minima.

Traditional approaches often perform a Quantization-aware Training (QAT) [35–37, 45] process to achieve guaranteed accuracy. However, the long time re-training and the requirements of full training data consume unacceptable computation and storage resources, making it impractical in industry. On the contrary, Post Training Quantization (PTQ) [2, 12, 25, 26, 43] is fast and light. PTQ efficiently turns a pre-trained full precision network to quantized one with only a small calibration set. Although more user-friendly, PTQ suffers performance degradation when pursuing ultra low compression ratio [28, 39] (e.g. lower than 4 bits). This is because only the magnitude information of weights and activations is utilized by simply applying rounding-to-nearest in predominant approaches. The irreparable quantization noise will accumulate layer by layer throughout the network.

To this end, some layer-wise or block-wise reconstruction-based algorithms (e.g. AdaRound [26], Bit-Split [28], AdaQuant [12], QDrop [39], BRECQ [19], UWC [20]) are proposed, which greatly improve the accuracy when the quantized weights go down to 4-bit. An analysis [26] on the second order Taylor series expansion of the task loss indicates reconstruction-based methods could reduce the error introduced by rounding because they leverage the interaction between weights. QDrop [39] proposes to take activation quantization into consideration to increase the flatness of loss landscape. Nevertheless, when the compression rate goes higher , e.g., the activations quantized to 4 bits or the weights quantized to 2 bits, or compressing more complex models, e.g., the recent prevailing Vision Transformer models [6, 23, 33], it still remains a non-negligible accuracy gap with original model [24, 42]. Fig. 1 shows that, as the bit-width goes lower, the loss landscapes have more ragged surface. Therefore, in lower bits' optimization process, the networks are easily trapped into bad local minima and result in performance degradation.

As mentioned above, optimizing a low-bit model in PTQ is very challenging. In forward process, full-precision weights and activations are quantized to a small set of fixed-point values, which introduces large quantization noise to the network. The quantization noise causes the loss distorted, i.e. a rugged loss landscape shown in Fig. 1. As a consequence, the distorted loss makes the optimization unstable, misleading the network to poor local minima. In QAT, some progressive approaches [14, 47] are proposed. For instance, CTMQ [14] trains multiple precision quantized models from high-bit to low-bit, and use the weight of trained higher bit model to initialize the next low-bit model. The progressive process is experimentally verified to be helpful for the quantized network to reduce the quantization noise, resulting in better local minima. In QAT, to fully optimize the low-bit quantization network, the progressive approaches preset a complex bit dropping schedule, including long precision sequence and tedious training iterations. With multiplied training cost, traditional progressive methods are not practical in PTQ.

In this paper, a sharpness term, detaching the quantization noise's impact on loss, is defined to precisely estimate the degree of the loss distortion. Based on the sharpness term, a self-adapted progressive quantization scheduler for PTQ, named Bit-shrinking, is designed to help the low-bit model find better local minima. Instead of directly quantizing the network to the target low bit-width, Bit-shrinking relaxes bit-width to continuous value and gradually shrink it to the target bit-width during optimization to limit the sharpness. Each time the bit-width is shrunk, the "instant" sharpness introduced is limited within a preset threshold. Shrinking the bit-width and adjusting the weights are iteratively performed until the target bit arrives. Consequently, with the "instant" sharpness term limited, the loss surface is smoothed which helps to find good minima. Different from traditional progressive approaches which evenly drops the integer bit-width, Bit-shrinking has more proper dropping scheduler on continuous bit-width. It tackles the additional training cost issue by alleviating over-optimizing some trivial bit-width.

This paper proposes a novel and practical Post-Training quantization framework to improve the accuracy of low-bit quantized network. The motivation is to reduce the impact of quantization noise during optimization by a Bit-shrinking strategy. We show that Bit-shrinking can help the low-bit network to find better local minima comparing with the direct quantization approach. Our main contributions are summarized as follows:

- Based on the observation of the loss landscape in low-bit quantization, we find that excessive sharpness of the loss landscape misleads the optimization direction. To precisely estimate and further limit the impact of quantization noise during optimization, we detach a sharpness term from the loss.
- To calibrate the optimization direction, we propose Bit-shrinking, a self-adapted progressive quantization scheduler for PTQ, to limit sharpness term small and stable during optimization. The landscape is smoothed as the progressive scheduler iteratively adds small "instant" sharpness and adjusts the network. As a result, Bit-shrinking helps quantized low-bit network find better local minima comparing with direct quantization approach, and results in better performance.
- Widely experiments, including Vision Transformer and CNN classification models on ImageNet dataset and Faster RCNN, RetinaNet detection models on COCO dataset, demonstrate the superiority of the Bit-shrinking without end-to-end fine-tuning. On the Vision Transformer models, our INT8 and INT6 models drop within 0.5% and 1.5% Top-1 accuracy, respectively. Our INT4 quantized model drops within 1.3% and 3.5% Top-1 accuracy on ResNet18 and MobileNetV2, which achieves the SOTA performance.

2. Background and Notation

Notation. $\mathbf{x}$ and $\mathbf{y}$ denote the input and the target variable, respectively. $\mathbb{E}[\cdot]$ denotes the expectation operator.

We use capital bold letters denoting tensors (or matrices) and small bold letters denoting the flattened version, e.g., W, $\mathbf{w}$. The quantized activations and weights are represented by $\hat{\mathbf{x}}_b$, $\hat{\mathbf{w}}_b$, with the subscript denoting the bit-width.

2.1. Related Work

Standard DNNs represented with float-point values are inefficient in memory storage and consumes considerable computational resources. Quantization is an effective way to save the consumption.

Post-Training Quantization. Post-training quantization (PTQ) is a lightweight approach since it doesn't require the original training pipeline. ACIQ [1] fits Gaussian and Laplacian models to the distribution for optimal clip threshold. [43] leverages model expansion to improve quantization. [28] split the bits of weight to compensate quantization error. [26] optimizes the rounding operation to improve the final loss. BRECQ [19] proposes a block-wise reconstruction algorithm implicitly leveraging the cross-layer interaction in a block. Unit-wise Calibration (UWC) [20] stresses to make use of the interaction across blocks for higher performance. Qdrop [39] proposes to take activation quantization into consideration to increase the flatness of loss landscape when optimization weights. In most cases, PTQ methods are sufficient to achieve near-original accuracy under 8-bit quantization, while the performance reduction becomes non-negligible when the bit-width goes less than 4-bit. For the currently prevailing Vision Transformer architectures [6, 23, 33], Liu et al. [24] uses the Pearson correlation coefficient and ranking loss as the metrics to determine the scaling factors. While PTQ4ViT [42] reformulates the format of uniform quantization and introduces two scaling factors to quantize the wide range data and narrow range data respectively in each layer.

Progressive Quantization. In order to tackle disturbance of quantization noise during training in low precision quantization, progressive quantization approaches are proposed [14, 47] in QAT. For instance, Zhuang et al. [46, 47] and Qu et al. [29] propose to progressively quantize and train the network from high-precision to low-precision. CTMQ [14] divides the quantization process into multiple steps. In each quantization step, the trained weights of a model are used to initialize the weights of the next model with the quantization bit depth reduced by one. However, these bit dropping schedulers is set beforehand. To ensure sufficient training for all bits, it takes a very long time. Despite the effectiveness, the tedious training process makes it unpractical in PTQ. Jung et al. [13] proposed an adaptive loss-aware quantization scheme for multi-bit networks. It gradually removes binary basis during training. It can be viewed as a combination of pruning and quantization. The cost on training is also very huge.

Flatness. The concept of "flat" local minima is pioneered by [10] for better generalization in neural networks. After that, many works such as adversarial training [40], improving robustness under perturbation [41, 44], have benefited from flat local minima. Quantization, that turns full-precision weights and activations to fixed pointed ones, could also be viewed as adding perturbation might prefer flat model. In Sharpness-aware Quantization (SAQ) [22], they find better optimization direction by encouraging the flatness via adding proper perturbation on quantized weights. In PTQ, Qdrop [39] helps the quantized model generalizing to activation quantization noise with more flatten loss landscape.

2.2. Background

Uniform quantization. For convenience, we revisit the uniform quantization. Given a bit-width b, it turns the weights or activations from full-precision space $\mathbb{R}^d$ to $\mathbb{V}^d$, where d represents dimension, $\mathbb{V} = \alpha_b \times \{-2^{b-1}, \dots, 2^{b-1}-1\}$ is the unified discrete potential value space of each element. α_b is a floating-point scale factor, representing the quantization interval. Omitting layer index, for example, quantizing a weight $\mathbf{w}$ is formulated as

$$\hat{\mathbf{w}} = Q(\mathbf{w}, b) = clip(\alpha_b \lfloor \frac{\mathbf{w}}{\alpha_b} \rceil, n, p), \tag{1}$$

where n and p are the negative and positive thresholds for clipping, corresponding to the min and max values in $\mathbb{V}$.

In this paper, we calculate α_b like previous low-bit quantization method [19], which considers the trade-off between the representation precision and the range. Generally, A higher bit-width will have higher precision, corresponding to a smaller α_b.

Network quantization. In this paper, we formulate network PTQ problem as follows. Given a small set of calibration data $\mathbb{S} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$, and a pre-trained neural network $\mathcal{G}$ parameterized with $\mathbf{w}_{orig}$, the goal of PTQ is to find a quantized network $\hat{\mathcal{G}}$ that could imitate the mapping between the input and output in $\mathcal{G}$. In $\hat{\mathcal{G}}$, weights and activations will be turned into fixed-point ones during inference. We omit the layer index for simplification.

Therefore, without loss of generality, we formulate PTQ as minimizing

$$\mathbb{E}_{\mathbf{x} \sim \mathcal{D}_{\mathbb{S}}} \left[L(\mathbf{w} + \Delta\mathbf{w}, \mathbf{x}, 1 + u(\mathbf{x})) - L(\mathbf{w}_{orig}, \mathbf{x}, 1) \right], \tag{2}$$

where L the loss function of the task, $\Delta\mathbf{w}$ is the injected quantization noise on weights. And $u(\mathbf{x})$ presents the injected noise on the intermediate activations in forward process, which is variable with $\mathbf{x}$ [39]. Eq. (2) minimizes the distance between the quantized network's output and the original network's output. Therefore, PTQ is to find an optimal $\mathbf{w}$ that remains original input-output mapping when quantization noises are added in inference .

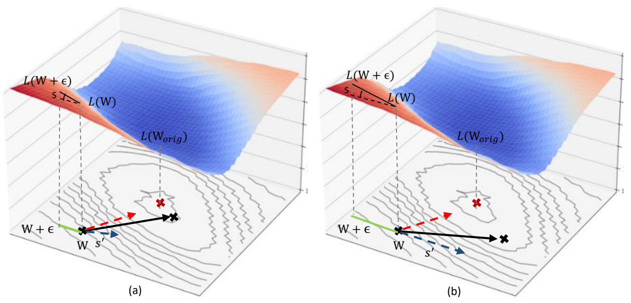

Figure 2. Schematic of the optimization process. (a) and (b) are high-bit case and low-bit case, respectively. ϵ represented by the green lines are the perturbation added on $\mathbf{w}$. With different magnitude of ϵ, the sharpness terms s, represented as the distance between $\mathcal{L}(\mathbf{w}+\epsilon)$ and $\mathcal{L}(\mathbf{w})$, are different in (a) and (b). The black arrow line is the optimization direction which is the resultant of the gradient produced by s (the blue dotted arrow line) and the gradient produced by (4-2) (red dotted arrow line). For high-bit optimization in (a), with small perturbation on $\mathbf{w}$, optimization direction is hardly disturbed. While low-bit optimization in (b), the perturbation is excessive that leads the weight to poor minima.

3. Method

3.1. Optimization difficulty in low-bit quantization

To further improve PTQ, we first investigate the loss landscapes of various bit-width networks. As shown in Figure 1, the low-bit model has a much rugged loss landscape compared with higher-bit ones. Lower-bit quantization introduces bigger quantization noises $\Delta\mathbf{w}$, $u(\mathbf{x})$. In the lower-bit case, the large noises cause severe loss fluctuations, and hence mislead the gradients, making the optimization process unstable. As a result, the low-bit quantized model is more likely to get trapped in poor local minima, resulting in performance degradation.

To precisely estimate the impact of quantization noise $\Delta\mathbf{w}$, $u(\mathbf{x})$, we divide our objective Eq.(2) into two parts:

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{\mathbb{S}}}\underbrace{L(\mathbf{w}+\Delta\mathbf{w},\mathbf{x},1+u(\mathbf{x}))-L(\mathbf{w},\mathbf{x},1)}_{(3-1)}+\underbrace{L(\mathbf{w},\mathbf{x},1)-L(\mathbf{w}_{orig},\mathbf{x},1)}_{(3-2)} \tag{3}$$

where term (3-1) reflects the impact of quantization noises $\Delta\mathbf{w}$, $1+u(\mathbf{x})$ on the loss, term (3-2) finds optimal weight that minimize the distance between the model's output and the original one. Therefore, our objective Eq. (3) that overall optimizing the sum of the two terms could be viewed as finding optimal $\mathbf{w}$ which could generalize to jitters on weights and activations.

To simplify the problem, we convert the impact of activations' quantization noise $u(\mathbf{x})$ into an equivalent impact quantization noise on weights $v(\mathbf{x})$ as QDrop [39] Lemma 1 did, i.e., adding $v(\mathbf{x})$ on weights $\mathbf{w}$ produces the same response as adding $u(\mathbf{x})$ on activations $\mathbf{x}$. Therefore, quantizing weights and activations into b bits, approximates to adding resultant perturbation $\epsilon=\Delta\mathbf{w}+v(\mathbf{x})$ on weights. Then Eq.(3) is reformulated as

$$\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_{\mathbb{S}}}\underbrace{L(\mathbf{w}+\epsilon,\mathbf{x})-L(\mathbf{w},\mathbf{x})}_{(4-1)}+\underbrace{L(\mathbf{w},\mathbf{x})-L(\mathbf{w}_{orig},\mathbf{x})}_{(4-2)}, \tag{4}$$

where ϵ is random perturbation whose magnitude increases as the bit-width decreases. Then, minimizing Eq. (4) is formed as optimizing the goal of reconstructing the output of original model, represented by (4-2), with a sharpness term (4-1) as [7] proposed. We symbolize (4-1) under b bit-width as $s(\mathbf{w},b)$.

In a forward process, $s(\mathbf{w},b)$ captures the increase of the reconstruction loss (4-2) when weight $\mathbf{w}$ randomly moves to a nearby weight $\mathbf{w}+\epsilon$. With certain perturbation, $s(\mathbf{w})$ estimates the sharpness of the loss landscape.

During optimization, the weights updating is guided by the sum of gradients coming from the sharpness term (4-1) and the reconstruction loss (4-2) as shown in Figure 2. For minimizing the reconstruction loss (4-2), the sharpness term (4-1)'s gradients are noises that disturbs the weight $\mathbf{w}$ to reconstruct the original output. In the higher-bit case, the sharpness term (4-1)'s gradients are small. With gradients from reconstruction loss play a major role, term (4-1)'s gradients can hardly hinder the weight $\mathbf{w}$ to find the right optimizing direction. On the contrary, in the low-bit case, term (4-1)'s gradients become a major component, which makes the total gradients unreliable. In this case, it can hardly find out the right optimizing direction and is easily trapped into poor minima.

3.2. Proposed method

In PTQ, optimizing a low-bit (e.g., lower than 4) quantization model is very challenging. The reason is, as aforementioned, the excessive quantization noises introduced during the forward process, produces in-accurate gradients that disturb the weights' updating direction. Rather than directly quantizing the network to the target bit-width which introduces excessive quantization noises at one time, we propose a Bit-shrinking optimization strategy. In the following, we will show how the Bit-shrinking strategy helps to calibrate the optimization direction.

Given a low target bit-width b_l, the sharpness introduced by large perturbation is excessive to mislead the direction of updating weights. In order to reduce the impact of sharpness, we propose to seek out weight with more flatten loss landscape before optimizing b_l-bit model. Rather than injecting excessive perturbation at one time, we divide the whole perturbation into multiple stages in which proper magnitude of perturbation is added which only introduce small "instant" sharpness. After the minimization of the "instant" sharpness each stage, a more flatten loss landscape is prepared for the next stage. More specifically, since the perturbation increases as the bit-width decreases, in each stage, we gradually decrease the bit-width from high-bit to proper bit-width to implement adding the perturbation.

To tackle the large $s(\mathbf{w}, b_l)$, we decompose it into some components that are easier to be optimized by rewriting Eq. (4) into

$$
\begin{aligned}
s(\mathbf{w}, b_l) &= \int_0^{\epsilon_{b_l}} L^{'}(\mathbf{w}+\epsilon, \mathbf{x}) d\epsilon \\
&= \underbrace{\int_0^{\epsilon_{b_c}} L^{'}(\mathbf{w}+\epsilon, \mathbf{x}) d\epsilon}_{(5-1)} + \underbrace{\int_{\epsilon_{b_c}}^{\epsilon_{b_l}} L^{'}(\mathbf{w}+\epsilon, \mathbf{x}) d\epsilon}_{(5-2)},
\end{aligned}
\tag{5}
$$

where $L^{'}(\mathbf{w}+\epsilon, \mathbf{x})$ is the derivative of $L(\mathbf{w}+\epsilon, \mathbf{x})$ to the perturbation ϵ. ϵ_{b_c} is a smaller perturbation introduced when the weights and activations are quantized to a higher bit-width b_c. (5-1) is the sharpness term $s(\mathbf{w}, b_c)$ when optimizing b_c-bit model. As analyzed above, when the sharpness term is small (assuming smaller than τ), the optimization direction of b_c-bit model is merely disturbed. For the seek of granular control the sharpness term within τ, we relax the bit-width to continuous value.

Therefore, seeking to minimize the sharpness term $s(\mathbf{w}, b_l)$, we first minimize a small component (5-1) by optimizing a higher-bit model. After the convergence of b_c-bit model, the weights $\mathbf{w}$ come to a new state, $\mathbf{w}^*_{b_c} = \arg\min_{\mathbf{w}}(L(\mathbf{w}+\epsilon_{b_c}, \mathbf{x}) - L(\mathbf{w}_{orig}, \mathbf{x}))$. The corresponding sharpness term to quantize the new weights $\mathbf{w}^*_{b_c}$ to b_l bits is hereby changed to

$$
s(\mathbf{w}^*_{b_c}, b_l) = \underbrace{\int_0^{\epsilon_{b_c}} L^{'}(\mathbf{w}^*_{b_c}+\epsilon, \mathbf{x}) d\epsilon}_{(6-1)} + \underbrace{\int_{\epsilon_{b_c}}^{\epsilon_{b_l}} L^{'}(\mathbf{w}^*_{b_c}+\epsilon, \mathbf{x}) d\epsilon}_{(6-2)},
\tag{6}
$$

According to the work [7] that study the flatness of the loss landscape, they obtain both low loss and low curvature in neighborhood of the minima by adding perturbation with proper magnitude on weight. Therefore, adding the perturbation ϵ_{b_c} during optimization helps $f(\mathbf{w}^*_{b_c})$ learning a more flatten landscape in the l^2 Euclidean ball with radius $|\epsilon_{b_c}|$. Then, we have $L^{'}(\mathbf{w}^*_{b_c}+\epsilon, \mathbf{x}) < L^{'}(\mathbf{w}+\epsilon, \mathbf{x})$ when $\epsilon < |\epsilon_{b_c}|$. (6-1) is minimized with a new state $\mathbf{w}^*_{b_c}$.

To further minimize Eq. (6), we re-decompose it into small components by finding a lower bit-width b_{c_2}, where $b_c > b_{c_2} > b_l$. With the b_c shrunk to b_{c_2}, Eq. (6) is minimized when the weights adapted to the new bit-width b_{c_2}.

In such way, by gradually shrink the b_c to a series of bits b_{c_i}, where $b_{c_1} > b_{c_2} ...... > b_{c_i} >= b_l$, Eq. (6) is finally minimized when b_{c_i} reaches b_l. As a result, when we optimize the target low-bit model, the loss landscape is smoothed.

Self-adapted progressive quantization scheduler. In the following, we will detail how the sharpness guide a self-adapted progressive quantization scheduler from a higher-bit to the target bit b_l.

Given the weights from the original pre-trained model, the weight $\mathbf{w}$ and activation $\mathbf{x}$ are first quantized into 8-bit, e.g., $b_{c_0} = 8$. Then, the following bit-width b_{c_i} is adaptively selected by limiting the increase of sharpness within τ when shrinking the former optimized bit-width $b_{c_{i-1}}$,

$$
\begin{aligned}
b_{c_i} &= \min(b_c) \\
&s.t. S(\mathbf{w}, b_c) - S(\mathbf{w}, b_{c_{i-1}}) < \tau,
\end{aligned}
\tag{7}
$$

We name $S(\mathbf{w}, b_{c_i}) - S(\mathbf{w}, b_{c_{i-1}})$ as "instant" sharpness, meaning the increased sharpness when shrinking the bit-width b_c. The "instant" sharpness ceiling τ is a fixed value, calculated by a small constant times of the total sharpness. We optimize $\mathbf{w}$ to adjust the block quantization noise under b_{c_i}-bit to make the weights adapted to the newly added "instant" noise.

Our self-adapted progressive quantization scheduler automatically performs iterative shrinking the bit-width based on Eq. (7) and optimizing process until the target bit-width arrives.

Algorithm 1 Self-adapted progressive quantization optimization.

```
Input: Pre-trained FP model $\mathbf{w}_{orig}$, calibration dataset, sharpness coefficient $\tau_c$, acceptable error coefficient $\epsilon_c$, optimization iteration $K$, target bit-width $b$.
Output: Quantized model $\hat{\mathbf{w}}$.
  for all $i = 1, \cdots, K$-th block in FP model do
    Collect calibration data to block's input $\hat{\mathbf{x}}^i_{inp}$ and output $\mathbf{y}^i_{out}$.
    Calculate the total sharpness $S_t = S(\mathbf{w}^i_{orig}, \hat{\mathbf{x}}^i_{inp}, b)$.
    Calculate sharpness ceiling $\tau \leftarrow \tau_c \times S_t$.
    Calculate sharpness searching acceptable error $\epsilon \leftarrow \epsilon_c \times S_t$.
    Initialize $b_c \leftarrow 8$, $\mathbf{w} \leftarrow \mathbf{w}_{orig}$.
    repeat
      Quantize activations $\mathbf{x}^i$ and weights $\mathbf{w}^i$ into $b_c$ bits.
      Update $\mathbf{w}^i$ by the gradients from block-wise reconstruction loss $\mathcal{L}^i$ for $K$ iterations.
      Shrinking $b_c$ by Algorithm 2.          ▷ sharpness within ceiling $\tau$
    until $b_c == b$
    Update $\mathbf{w}^i$ by $\mathcal{L}^i$ for another $K$ iterations.
  end for
```

3.3. Implementation Details

Following BRECQ [19], we divide the network into blocks, and reconstruct the input and output mapping block-wisely. Adam [15] optimizer and the ReduceLROnPlateau Scheduler with the beginning learning rate of $5e-6$ are used to optimize the weights. The algorithm is shown in Algorithm 1.

Given a desired bit-width b, we first calculate the total sharpness S_b using 32 images from the calibration data.

Algorithm 2 Self-adapted progressive quantization scheduler.

```
Input: Optimized weights $\mathbf{w}_{b_c}^{i*}$, current bit $b_c$, target bit $b$, sharpness ceiling $\tau$, acceptable error $\epsilon$. Block input and output $\hat{\mathbf{x}}_{inp}^i$, $\mathbf{y}_{out}^i$.
Output: Next optimizing bit-width $b_n$.
  if $S(\mathbf{w}_{b_c}^{i*}, \hat{\mathbf{x}}_{inp}^i, b) < \tau + \epsilon$ then
      return $b$
  end if
  Initialize the searching interval $b_{low} \leftarrow b_c, b_{high} \leftarrow b$.
  repeat
      $b_n \leftarrow (b_{low} + b_{high})/2$                    ▷ half of the interval.
      Calculate new sharpness $S_n = S(\mathbf{w}_{b_c}^{i*}, \hat{\mathbf{x}}_{inp}^i, b_n)$
      if $S_n > \tau + \epsilon$ then
          $b_{low} \leftarrow b_n$
      else $S_n < \tau - \epsilon$
          $b_{high} \leftarrow b_n$
      end if
  until $S_n \in [\tau - \epsilon, \tau + \epsilon]$
```

The sharpness ceiling τ is set to $0.04 \times S_b$, where 0.04 is a hyper-parameter chosen by conducting grid search over $\{0.01, 0.02, \cdots, 0.1, 0.5, 1\}$. The Bit-shrinking optimization process begins from 8 bits, e.g., $b_{c_0} = 8$. Then, the following continuous bit-width b_{c_i} is adaptively selected by limiting the increase of sharpness within τ when shrinking the former optimized bit-width $b_{c_{i-1}}$ by Eq. (7). The sharpness $S(b_c)$ is calculated using 32 images from the calibration data. We use half-interval search to appropriately match the desired sharpness within $\epsilon = 0.01$ of the total sharpness error shown in Algorithm 2.

For each bit-width, 3200 iterations' optimization are applied. When weights and activations are quantized to different bit-widths, e.g., 2 and 4 bits, we shrink the bit-width of weights and activations to 4 bits, and then shrink the weights' bit-width to 2 bits.

4. Experiments

In this section, we evaluate the effectiveness of our proposed method on various computer vision tasks and models. Section 4.1 presents ablation study on the Bit-shrinking optimization strategy on classification task on ImageNet. In Section 4.2, we compare Bit-shrinking optimization strategy among other post-training quantization methods on classification task. We also present the performance of Bit-shrinking optimization strategy on object detection task on COCO.

4.1. Ablation Study

4.1.1 Bit-shrinking vs. Direct.

We investigate the benefits of our proposed Bit-shrinking optimization algorithm by comparing with directly quantizing the model to the target bit-width optimization algorithm. In our experiments, five widely used convolutional models, including ResNet18, ResNet50, ResNet101 [8], InceptionV3 [32], MobileNetV2 [11] are used for comparison.

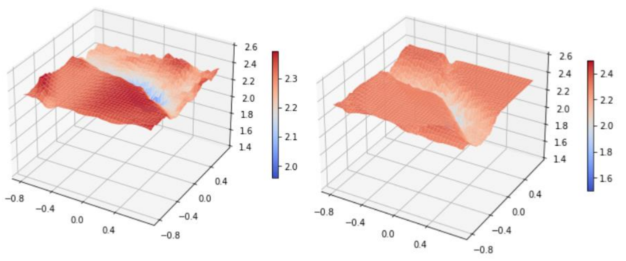
(a) 2-bit landscape of direct quantization. (b) 2-bit landscape of bit-shrinking quantization

Figure 3. Loss landscapes of direct quantization and bit-shrinking

All the pre-trained models are trained on ImageNet [5] with the open-source codes [3] and [4]. The Direct approach is performed by directly optimizing the target-bit quantized network as a baseline. In the Bit-shrinking approach, we first quantize the network to 8-bit.

In both Direct approach and Bit-shrinking approach, the scale factors of weights and activations, i.e. $\alpha_{b_l,\mathbf{w}}$, $\alpha_{b_l,\mathbf{x}}$ are calculated by the method in [19]. In Bit-shrinking approach, the scale factors α_{b_c} of higher-bit b_c are calculated based on α_{b_l} by $\alpha_{b_c} = \alpha_{b_l} \times 2^{b_l - b_c}$. The results are shown in Table 1.

Shown in Table 1, the proposed Bit-shrinking method provide obvious accuracy improvement on all models even on the less redundant models, e.g. MobilenetV2 comparing with direct method, which demonstrate than limiting the impact of quantization noise by Bit-shrinking during optimization could help the quantized network find better minima. Under 4-bit quantization of weights and activations, the quantized models induce negligible performance degradation with 1.3% to 3.5% Top-1 accuracy drop on various networks. For the more aggressive 2-bit and 3-bit quantization of weights and activations, the Bit-shrinking shows more obvious performance improvement. For example, under the 3-bit quantization, Bit-shrinking strategy has more than 1% improvement in Top-1 accuracy on the direct quantization approach on all networks. Another phenomenon is that the gap becomes larger when the original model becomes more compact. For example, on InceptionV3 and MobilenetV2, Bit-shrinking achieves 1% uplift under 4-bit quantization. When the bit-width decreases to 2-bit quantization, the improvement expands to more than 10%.

4.1.2 Bit-shrinking vs. Progressive quantization

Progressive quantization is proven in QAT to be helpful for the quantized network to reduce the disturbance of quantization noise and results in better local minima. However, it needs a complex bit dropping schedule, including long precision sequence and tedious training iterations, making it unpractical in PTQ. Bit-shrinking avoid the tedious process of traditional progressive quantization by adopting a more

Table 1. Comparison with Direct post-training quantization approaches on ImageNet classification benchmark. Top-1 accuracy (%) are reported. Bold values indicate best results. Weights and activations are quantized to 2 to 4 bits.

Algorithms	W	A	ResNet18	ResNet50	ResNet101	InceptionV3	MobileNetV2
FP.	32	32	71.24	77.25	77.37	77.57	72.58
Direct	4	4	69.80	75.68	76.11	73.81	68.15
Bit-shrinking	4	4	**69.94**	**76.04**	**76.52**	**74.79**	**69.02**
Direct	3	3	66.33	71.74	72.89	46.77	32.15
Bit-shrinking	3	3	**67.12**	**72.91**	**74.14**	**54.17**	**58.66**
Direct	2	4	64.63	69.65	70.15	49.88	48.17
Bit-shrinking	2	4	**65.77**	**71.11**	**71.26**	**52.16**	**54.88**
Direct	2	2	45.27	38.18	40.35	33.31	2.17
Progressive	2	2	52.41	53.15	52.58	35.49	15.32
Bit-shrinking	2	2	**57.33**	**59.03**	**56.98**	**37.68**	**18.23**

proper scheduler.

Bit-shrinking adaptively selects the bit-width by limiting the increase of sharpness, and achieves lower training loss and much less optimizing iterations than progressive quantization method, as shown in Fig. 4, We present the realistic loss landscapes of direct quantization and bit-shrinking. As shown in Fig. 3, the magnitude of injected noise is smaller in (b) with more flatten loss landscape.

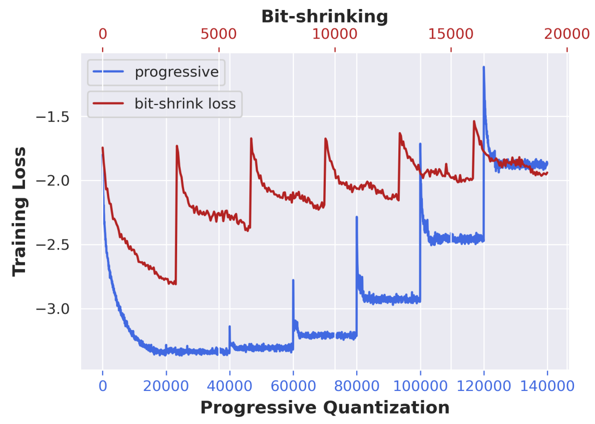

Figure 4. **Loss carves of progressive quantization.** In the progressive quantization, the bit-width scheduler is [8, 7, 6, 5, 4, 3, 2]. For each bit-width, 20000 iterations' optimization are applied to ensure fully optimize the network. In the Bit-shrinking quantization, the bit-width scheduler is [8, 2.87, 2.58, 2.36, 2.18, 2]. For each bit-width, 3200 iterations' optimization are applied.

4.2. Comparison with State-of-the-arts

4.2.1 Vision Transformer on ImageNet Classification.

We evaluate our algorithm on currently prevailing Vision Transformer architectures, including ViT [6] DeiT [33], and Swin [23] and compare the performance with previous method PTQ4ViT [42] and Liu [24]. PTQ4ViT reformulates the format of uniform quantization and introduces two scaling factors to quantize the wide range data and narrow range data respectively in each layer. Liu et al. [24] uses the Pearson correlation coefficient and ranking loss as the metrics to determine the scaling factors. The results are demonstrated in Tab. 2. For the W8A8 quantization, Bit-shrinking and PTQ4ViT both achieve less than 0.5% Top-1 accuracy drop. The accuracy drop of liu is 2.4%(W8A8) and 1.7%(W8A8 mixed-precision) on DeiT. For W6A6 quantization, Bit-shrinking results in 1% Top-1 accuracy drop on average on all architectures while 2.1% on average for PTQ4ViT. The accuracy drop of Liu et al. [24] is 5.2%(W6A6) and 4.7%W6A6 mixed-precision on DeiT. Even on the experiments of W4A6 quantization, our accuracy drops are less than 2%.

4.2.2 CNN on ImageNet Classification.

Here, we evaluate our algorithm and compare with the State-of-the-arts post-training quantization approaches, including ACIQ-Mix [1], ZeroQ [2], LAPQ [27], Bit-split [28], AdaRound [26], BRECQ [19], AdaQuant [19] and QDrop [39]. Shown in Table 3, the proposed Bit-shrinking strategy outperforms all competing methods for all bit-width setting. Under 4-bit quantization, the compared methods still report good performance on the relatively redundant models, e.g., ResNets. Among all methods, our method leads to the smallest accuracy drop within 1.3%. For the more challenging networks, InceptionV3 and MobilenetV2, 4-bit quantization has a bigger impact. In this case, our method shows prominent superiority comparing with other methods. For MobilenetV2, all other methods lead to un-tolerable performance degradation, while our approach obtain the best result with only 2.46% drop in accuracy. When the bit-width goes down to 3, Bit-shrinking

Table 2. Comparison among typical post-training quantization strategy on Vision Transformer models on ImageNet-1k validation set in terms of Top-1 Accuracy.

Algorithms	Bits (W/A)	ViT-S	ViT-B	DeiT-S	DeiT-B	Swin-T	Swin-S
Full Prec.	32/32	81.39	84.54	79.80	81.80	81.39	83.23
PTQ4ViT [42]	8/8	81.00	84.25	79.47	81.48	81.24	83.10
Ours	8/8	81.09	84.33	79.49	81.56	81.20	83.12
PTQ4ViT [42]	6/6	78.63	81.65	76.28	80.25	80.47	82.38
Ours	6/6	80.44	83.16	78.51	80.47	80.61	82.44
Ours	4/6	80.07	82.77	77.93	79.97	80.49	82.30

Table 3. Comparison with State-of-the-arts post-training quantization approaches on ImageNet classification benchmark. Top-1 accuracy (%) are reported. Bold values indicate best results (with the least accuracy drop).

Algorithms	W/A	ResNet18	ResNet50	MobileNetV2
FP.	32/32	71.24	77.25	72.58
ACIQ-Mix [1]	4/4	67.00	73.80	64.33
ZeroQ [2]	4/4	21.71	2.94	26.24
LAPQ [27]	4/4	60.30	70.00	49.70
Bit-split [28]	4/4	67.56	73.71	-
AdaRound [26]	4/4	69.36	74.76	64.33
BRECQ [19]	4/4	69.60	75.05	66.57
QDrop [39]	4/4	69.62	75.45	68.84
AdaQuant [19]	4/4	68.60	75.90	-
Ours	4/4	**69.94**	**76.04**	**69.02**
AdaRound [26]	3/3	60.09	67.46	2.23
BRECQ [19]	3/3	65.87	68.96	23.41
QDrop [39]	3/3	66.75	72.38	57.98
Ours	3/3	**67.12**	**72.91**	**58.66**
AdaRound [26]	2/4	64.14	68.40	41.52
BRECQ [19]	2/4	64.80	70.29	53.34
QDrop [39]	2/4	65.25	70.65	54.22
Ours	2/4	**65.77**	**71.11**	**54.88**
BRECQ [19]	2/2	42.54	29.01	0.24
QDrop [39]	2/2	54.72	58.67	13.05
Ours	2/2	**57.33**	**59.03**	**18.23**

Table 4. Comparison among typical post-training quantization strategies on MS COCO validation set in terms of mAP. Following BRECQ and QDrop, the weights and activations in head are remain un-quantized.

Algorithms	Bits (W/A)	Faster RCNN		RetinaNet	
		R18	R50	R18	R50
Full Prec.	32/32	34.7	38.8	33.4	37.0
AdaRound	4/4	32.6	34.5	31.0	33.5
BRECQ	4/4	32.6	34.6	31.2	33.5
QDrop	4/4	33.4	37.0	32.0	35.7
Ours	4/4	**33.8**	**37.4**	**32.5**	**36.0**
BRECQ	2/4	29.92	30.23	28.73	29.47
QDrop	2/4	31.01	34.33	29.69	33.01
Ours	2/4	**31.95**	**35.61**	**30.78**	**34.15**

results in less performance degradation. Our method outperforms QDrop, BRECQ and AdaRound since the loss landscape is more flatten to achieve better minima. In the more aggressive 2-bit weights quantization, even both ours and QDrop's show obvious loss of performance, the bit-shrinking strategy helps us get higher accuracy.

4.2.3 Object Detection.

To validate the effectiveness and applicability of Bit-shrinking strategy, the experiments of object detection task are applied. Bit-shrinking strategy has been evaluated on object detection with one-stage RetinaNet [21] and two-stage Faster R-CNN [31], models. For all networks, we choose Resnet18 and Resnet50 as backbone. MS COCO is adopted as the testing set to evaluate our method. For calibration and validation data, we resize them to 1333×800. Since the input images are much bigger than classification images, only 400 images are sampled as calibration data. The bounding box mAP performance for object detection is reported in Table 4. According to Table 4, we can see that there are within 1% mAP degradation without re-training the network, which demonstrate our method nearly achieves near-to-original performance with 4-bit weight and 4-bit activation quantization.

Conclusion

In this paper, we propose a Bit-shrinking strategy to improve PTQ under low bit-width. In order to alleviate the impact of excessive quantization noise, we divide the whole perturbation introduced by low-bit quantization into multiple stages in which proper magnitude of perturbation is added to limit the "instant" sharpness small. As a result, the impact of excessive quantization noise is alleviated with more flatness landscape.

References

[1] R. Banner, Yury Nahshan, E. Hoffer, and Daniel Soudry. Aciq: Analytical clipping for integer quantization of neural networks. *ArXiv*, abs/1810.05723, 2018. 3, 7, 8

[2] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework, 2020. 1, 7, 8

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 6

[4] MMClassification Contributors. Openmmlab's image classification toolbox and benchmark. `https://github.com/open-mmlab/mmclassification`, 2020. 6

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR09*, 2009. 6

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3, 7

[7] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020. 4, 5

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[9] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE ICCV*, pages 1389–1397, 2017. 1

[10] Sepp Hochreiter and Jürgen Schmidhuber. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994. 3

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 6

[12] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Improving post training neural quantization: Layer-wise calibration and integer programming, 2020. 1, 2

[13] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu 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*, pages 4350–4359, 2019. 3

[14] HyunJin Kim, Jungwoo Shin, and Alberto A Del Barrio. Ctmq: Cyclic training of convolutional neural networks with multiple quantization steps. *arXiv preprint arXiv:2206.12794*, 2022. 2, 3

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 5

[16] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Block-wisely supervised neural architecture search with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1989–1998, 2020. 1

[17] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1

[18] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018. 1

[19] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021. 2, 3, 5, 6, 7, 8

[20] Chen Lin, Zheyang Li, Bo Peng, Haoji Hu, Wenming Tan, Ye Ren, and Shiliang Pu. Uwc: Unit-wise calibration towards rapid network compression. *arXiv preprint arXiv:2201.06376*, 2022. 2, 3

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Doll´ar. Focal loss for dense object detection. 2017. 8

[22] Jing Liu, Jianfei Cai, and Bohan Zhuang. Sharpness-aware quantization for deep neural networks. *arXiv preprint arXiv:2111.12273*, 2021. 3

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2, 3, 7

[24] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for vision transformer. *Advances in Neural Information Processing Systems*, 34:28092–28103, 2021. 2, 3, 7

[25] Eldad Meller, Alexander Finkelstein, Uri Almog, and Mark Grobman. Same, same but different - recovering neural network quantization error through weight factorization, 2019. 1

[26] Markus Nagel, Rana Ali Amjad, Mart van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization, 2020. 1, 2, 3, 7, 8

[27] Yury Nahshan, Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Alex M Bronstein, and Avi Mendelson. Loss aware post-training quantization. *Machine Learning*, 110(11):3245–3262, 2021. 7, 8

[28] Wang Peisong, Qiang Chen, Xiangyu He, and Cheng Jian. Towards accurate post-training network quantization via bit-split and stitching. In *Proceedings of the 37nd International Conference on Machine Learning (ICML)*, pages 243–252, July 2020. 2, 3, 7, 8

[29] Zhongnan Qu, Zimu Zhou, Yun Cheng, and Lothar Thiele. Adaptive loss-aware quantization for multi-bit networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7988–7997, 2020. 3

[30] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016. 1

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. 2015. 8

[32] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015. 6

[33] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 2, 3, 7

[34] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed precision dnns: All you need is a good parametrization, 2020. 1

[35] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8612–8620, 2019. 1

[36] Peisong Wang and Jian Cheng. Fixed-point factorized networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4012–4020, 2017. 1

[37] Peisong Wang, Qinghao Hu, Yifan Zhang, Chunjie Zhang, Yang Liu, and Jian Cheng. Two-step quantization for low-bit neural networks. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 4376–4384, 2018. 1

[38] Yunhe Wang, Chang Xu, Jiayan Qiu, Chao Xu, and Dacheng Tao. Towards evolutionary compression. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2476–2485, 2018. 1

[39] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*, 2022. 2, 3, 4, 7, 8

[40] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020. 3

[41] Haichao Yu, Linjie Yang, and Humphrey Shi. Is in-domain data really needed? a pilot study on cross-domain calibration for network quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3043–3052, 2021. 3

[42] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. Ptq4vit: Post-training quantization framework for vision transformers. *arXiv preprint arXiv:2111.12293*, 2021. 2, 3, 7, 8

[43] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *ICML*, 2019. 1, 3

[44] Yaowei Zheng, Richong Zhang, and Yongyi Mao. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8156–8165, 2021. 3

[45] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 1

[46] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7920–7928, 2018. 3

[47] Bohan Zhuang, Mingkui Tan, Jing Liu, Lingqiao Liu, Ian Reid, and Chunhua Shen. Effective training of convolutional neural networks with low-bitwidth weights and activations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 2, 3