AMPA: Adaptive Mixed Precision Allocation for Low-Bit Integer Training

Li Ding [1] **Wen Fei** [1] **Yuyang Huang** [1] **Shuangrui Ding** [2]
Wenrui Dai [1] **Chenglin Li** [1] **Junni Zou** [1] **Hongkai Xiong** [1]

Abstract

Low-bit integer training emerges as a promising approach to mitigate the heavy burden during network training by quantizing the weights, activations, and gradients. However, existing methods cannot well achieve mixed-precision quantization for low-bit training and are commonly limited to INT8 precision. In this paper, we propose a novel low-bit integer training framework that, for the first time, achieves adaptive mixed-precision allocation (AMPA) for weights, activations, and gradients, and pushes the boundaries to a precision level below INT8. We develop a novel magnitude-based sensitivity measurement with regard to the quantization losses of weight, activation, and gradient quantization and the average gradient magnitudes, which is demonstrated as an upper bound of quantization influence in theory. We further design a layer-wise precision update strategy under observations on the quantization losses and their effects on model performance in low-bit training. Extensive experiments on different backbones and datasets show that, compared to INT8 quantization, the proposed method can achieve more than 38% BitOPs reduction with a tolerable loss below 2% in image classification, image segmentation, and language modeling.

1. Introduction

In recent years, deep learning techniques have demonstrated remarkable progress across various tasks. Nevertheless, the success of deep neural networks (DNNs) originates from their explosive volume of model parameters. The substantial growth in computational demands and resource prerequisites (Sevilla et al., 2022) poses unprecedented challenges to the practical deployment of deep learning methods. To mitigate this problem, network quantization has been widely studied as a promising solution that alleviates computational burdens without altering network architectures.

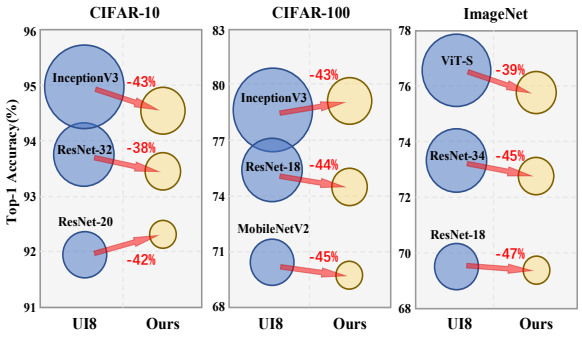


Figure 1: Image classification with the proposed AMPA low-bit training framework. For each dataset, the areas of bubbles are positively correlated with BitOPs of training the network. AMPA can save more than 38% BitOPs with tolerable loss below 2% compared with INT8 training (Zhu et al., 2020).

Network quantization can be employed in both inference and training stages (i.e., inference quantization and training quantization). Different from inference quantization that quantizes the weights or activations for acceleration in deployment (Jacob et al., 2018; Choi et al., 2018), training quantization (Cambier et al., 2020; Fu et al., 2021) simultaneously projects gradients, weights, and activations into low-precision representations during training. Since the backward computation (backpropagation) consumes nearly twice the computational cost of the forward computation (Zhao et al., 2021), low-bit training is crucial to reduce resource consumption and enable efficient training of DNNs. Compared with low-bit floating-point (FP16/FP8) training, low-bit integer training enjoys lower computation cost and better efficiency on hardware. Recent attempts (Zhu et al., 2020; Zhao et al., 2021) successfully maintain model performance under INT8 training. In this paper, we explore integer training with further lower bits and equivalent performance.

DNNs could suffer from severely degraded performance

[1]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. [2]Department of Information Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong. Correspondence to: Wenrui Dai <daiwenrui@sjtu.edu.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

when directly quantized into lower precisions than INT8 during training. The idea of layer-wise mixed precision quantization, typically adopted in inference quantization (Dong et al., 2020; Tang et al., 2022), is a promising alternative. However, there are significant differences in mixed-precision quantization during training. **First**, training quantization should not introduce much computational overhead, rendering mixed-precision quantization methods based on solution-space search (Wu et al., 2018a; Elthakeb et al., 2018) impractical. **Second**, gradients during training are non-zero, making Hessian-based sensitivity evaluation methods (Dong et al., 2019; 2020) not effective. **Third**, the impact of quantization evolves throughout the training process, necessitating dynamic adjustments of bitwidths. (Zhang et al., 2020) realized layer-wise mixed-precision gradients by introducing a predefined threshold of distribution difference from quantization. However, the predefined threshold cannot accommodate all models, and a more reasonable sensitivity measurement for training quantization is needed.

In this paper, we propose a novel integer training framework with layer-wise mixed-precision quantization of weights, activations, and gradients. Different from (Zhang et al., 2020) that uses a preset fixed threshold to determine the bitwidths for different layers, we propose an adaptive sensitivity-based scheme for layer-wise bitwidth allocation. Specifically, we develop a novel magnitude-based sensitivity measurement that involves the quantization losses associated with quantized weights, activations, and gradients and the average gradient magnitudes. We demonstrate via theoretical analysis that the proposed measurements provide upper bounds of the influence of the quantization loss without introducing excessive computations.

Moreover, we design an adaptive layer-wise selection strategy for multiple precisions under observations on the weight, activation, and gradient quantization losses, and their different effects on model performance. Finally, we present the Adaptive Mixed-Precision Allocation (AMPA) training framework for integer training with an average precision lower than INT8. Experimental results demonstrate that, compared with INT8 quantization (Zhu et al., 2020), the proposed framework can further reduce bit operations (BitOPs) by 38% with a tolerable less than 2% performance loss on various backbones and datasets as illustrated in Figure 1. The contributions of this paper are summarized below.

1. To our best knowledge, we are the first to propose an integer training framework that achieves Adaptive Mixed-Precision Allocation (AMPA) for weights, activations, and gradients with an average precision lower than INT8.

2. We develop a novel magnitude-based sensitivity measurement for weight, activation, and gradient quantization in the training stage and demonstrate it provides an upper bound of the quantization influence in theory.

3. We develop a mixed-precision low-bit training method based on the developed measurement and well-designed layer-wise precision selection strategy and validate its effectiveness with comprehensive experiments on various backbones and datasets.

2. Related Work

Inference Quantization. Different from network pruning (Kim et al., 2022; Ding et al., 2023), inference quantization quantizes the weights or activations of the trained neural networks from full precision to lower-bit representations for inference speedup. Representative works (Hubara et al., 2016; Rastegari et al., 2016; Li et al., 2016) employ binary or ternary bitwidths for weights and activations to accelerate network inference. TQT (Jain et al., 2020) improves the former work and achieves near-floating point accuracy. Jacob et al. (2018) develop a standard INT8 quantization scheme that balances accuracy and inference latency.

Mixed Precision Quantization. Mixed precision quantization involves allocating appropriate bit widths for different layers or even rows within layers (Chang et al., 2021) and is commonly utilized in inference quantization. Existing layer-wise mixed precision quantization approaches can be classified into two main categories. The first category is based on the solution space search. (Wu et al., 2018a) and (Yu et al., 2020) employ the network architecture search method to identify optimal bitwidths, while reinforcement learning is adopted in (Wang et al., 2019; Elthakeb et al., 2018) to determine the allocation scheme. Additionally, (Yang & Jin, 2021) leverage continuous and differentiable bitwidths to search the mixed precision. Another effective approach involves using sensitivity evaluation to assign bitwidths. Hessian matrix is adopted in (Dong et al., 2019; 2020; Yao et al., 2021) to evaluate the layer sensitivity.

Training Quantization. There are three main categories of methods in existing studies of low-bit training. The first category utilizes low-bit floating point representations or new data format for training (Wang et al., 2018; Mellempudi et al., 2019; Cambier et al., 2020), which can achieve almost lossless accuracy but sacrifices the acceleration performance compared to integer quantization. The second category quantizes the entire network with low-bit integers (Zhou et al., 2016; Wu et al., 2018b; Yang et al., 2020) which offers a higher compression ratio with significantly degraded model performance compared to full precision models. The third category focuses on quantizing only the convolutional layers (Zhu et al., 2020; Zhao et al., 2021) or the linear layers (Xi et al., 2023) in the model which can achieve a good balance between model accuracy and compression ratio. This paper is closest to the third category and designs the adaptive mixed precision allocation training framework, which is applicable to both convolutional and linear layers.

3. Adaptive Mixed Precision Allocation for Low-bit Integer Training

This section elaborates on the proposed adaptive mixed precision allocation (AMPA) framework for low-bit integer training. We first introduce the preliminaries of quantization and present the observation for the proposed framework. Then, we propose the sensitivity measurement for weights, activations, and gradients. Finally, we present the overall allocation scheme and introduce the fair allocation threshold.

3.1. Preliminaries

Integer Quantization maps from a floating-point value to an integer number. Uniform quantization is popular in network quantization (Krishnamoorthi, 2018; Zhu et al., 2020). It can be classified into symmetric quantization and asymmetric quantization based on whether the zero points before and after quantization are corresponding. The asymmetric quantization of the full precision data d_f is expressed as:

$$\hat{d}_q = \text{round}\left(\frac{1}{s} \cdot \text{clamp}(d_f, m, M) + z\right), \quad (1)$$

where $\text{clamp}(\cdot)$ is the clipping function, $\text{round}(\cdot)$ is the rounding function, $s = (M-m)/(2^N - 1)$ is the scaling factor, N is the bitwidth for quantization, and M and m are the maximal and minimal clipping values respectively. Nearest rounding is usually used for weight and activation quantization while stochastic rounding (Gupta et al., 2015) is often adopted for the gradient quantization. The de-quantization result is $d_q = (\hat{d}_q - z) \cdot s$. The symmetric quantization obtains zeros for the zero-point z:

$$\hat{d}_q = \text{round}\left(\frac{1}{s} \cdot \text{clamp}(d_f, -R, R)\right), \quad (2)$$

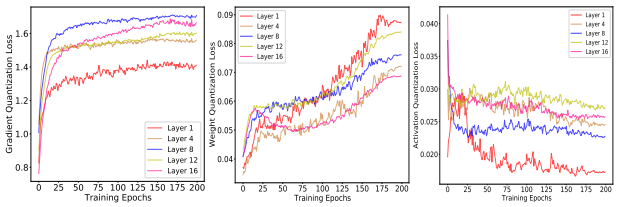
where $R \in (0, \max(|d_f|)$ is the maximal clipping value, scaling factor is $s = R/(2^{N-1} - 1)$. The de-quantization result is $d_q = \hat{d}_q \cdot s$.

3.2. Observations for Layer-wise Bitwidth Selection

We begin with three observations on the characteristics of low-bit training to motivate the layer-wise bitwidth selection that increases from low-bits to high-bits.

Observation i): Quantization loss in different layers during the training process. As illustrated in Figure 2, in the same training epoch, the relative quantization loss varies significantly in different layers. Moreover, for a given layer, the gradient and weight quantization loss tend to increase during training while there is no clear trend in the activation quantization. Overall, the quantization loss is different across different layers and training stages.

Observation ii): Increasing bitwidths throughout training. We conduct preliminary experiments on the dataset CIFAR-10 by switching the weight, activation, and gradient bitwidths from INT4 to INT8 at 40% of the training process and from INT8 to INT4 at 60% of the training process, with a total of 100 training epochs. As illustrated in Table 1, utilizing lower bitwidths in the later stages has more impact on the model performance. Overall, at different stages of training, the network exhibits varying levels of tolerance to the quantization loss.



(a) Gradients (b) Weights (c) Activations

Figure 2: The average quantization loss $\mathbf{E}(|x_f - x_q|/|x_f|)$ of (a) gradients, (b) weights, and (c) activations with full INT8 quantization when training ResNet-20 on CIFAR-10.

Observation iii): Adjusting weights, activations, and gradients in different numbers of layers. The data flow of weights, activations, and gradients exhibit distinct characteristics. We conduct preliminary experiments on the model ResNet-20 and MobileNetV2 using CIFAR-10 with 100 epochs by adjusting the bitwidths of weights, activations, and gradients from INT4 to INT8 at different time points. Each row in Table 2 is obtained by fixing the other rows at INT8. 'Time Point = 0.5' signifies changing the bitwidth at the midpoint of the training process while 'Time Point = 1.0' denotes maintaining INT4. The results presented in Table 2 highlight the model's heightened sensitivity to gradients, with weight quantization exhibiting a relatively smaller impact compared to activations and gradients. Overall, the influence of quantization loss of weights, activations, and gradients vary with respect to the model performance.

3.3. Magnitude-based Sensitivity Measurements

Consider the training set of N-pair input data x_i and corresponding label y_i for supervised learning. We achieve layer-wise quantization by minimizing the loss function L.

$$L = \frac{1}{N}\sum_{i=1}^{N} f(x_i, y_i). \quad (3)$$

Following (Yao et al., 2021; Zhe et al., 2019), we assume that the quantization of weights, activations, and gradients in each layer influence the model independently. We first show that the Hessian-based sensitivity measurement cannot be employed for quantization in low-bit integer training and then propose a magnitude-based sensitivity measurement for weights, activations, and gradients.

Table 1: Classification accuracy(%) by altering bitwidths from INT4 to INT8 or from INT8 to INT4 on CIFAR-10.

	ResNet-20	ResNet-32	MobileNetV2
INT4→INT8	90.75	91.74	92.95
INT8→INT4	89.52	89.97	90.90

Table 2: Classification accuracy(%) of ResNet-20 and MobileNetV2 under various time points to alter bitwidths (INT4→INT8) of weights, activations and gradients.

Time Point	ResNet-20			MobileNetV2		
	0.5	0.8	1.0	0.5	0.8	1.0
W	91.79	91.70	91.57	93.74	93.68	93.60
A	91.66	91.64	91.39	93.52	93.37	91.04
G	90.47	89.32	89.29	89.17	87.38	86.63

3.3.1. Analysis on Hessian-based Sensitivity Measurement

According to (Dong et al., 2019; 2020), we consider the Hessian-based sensitivity for one batch in training. To facilitate understanding, the symbols are used according to the following rules: i) Generally, the subscripts f and q indicate that it is full-precision or quantized, respectively, and ii) $d \in \{w, a, g\}$ means d is weight, activation or gradient.

The impact of quantizing d_f^i into d_q^i in the i-th layer on the loss function ΔL_d^i is

$$\Delta L_d^i = L(d_q^1, \cdots, d_q^i, \cdots, d_q^n) - L(d_q^1, \cdots, d_f^i, \cdots, d_q^n). \quad (4)$$

Hessian-based methods (Dong et al., 2019; 2020) consider the Taylor expansion of ΔL_d^i with regard to weights and activations (*i.e.*, d is w or a). Let us define $\Delta d^i = d_q^i - d_f^i$. For the i-th layer,

$$\Delta L_d^i = g_{d_f^i}^T \cdot \Delta d^i + \frac{1}{2}(\Delta d^i)^T H^i \Delta d^i + o(\|\Delta d^i\|)^3. \quad (5)$$

For inference quantization, Hessian-based methods (Dong et al., 2019; 2020) assume that a DNN is well trained such that the gradients $g_{d_f^i}$ vanish. Thus, the second-order term $(\Delta d^i)^T H^i \Delta d^i$ based on the Hessian H^i is selected as the sensitivity measurement for any ith layer. Contrary to inference quantization, training quantization requires non-zero gradients to update the low-bit weights and activations. The first-order term $g_{d_f^i}^T \cdot \Delta d^i$ becomes dominant in ΔL_d^i compared with the second-order term $(\Delta d^i)^T H^i \Delta d^i$ (about the order of $1e$-2 for INT4 and $1e$-4 for INT8). This fact raises two problems on the Hessian-based sensitivity measurement for training quantization as below.

i) Weight and activation quantization: A gradient-based sensitivity measurement is necessarily required to allow correct evaluation and fast computation of the quantization impact. The estimation of the trace of the Hessian, such as the Hutchinson algorithm (Dong et al., 2020), involves much computation overload.

ii) Gradient quantization: Hessian-based sensitivity neglects the gradient-based first-order term in Equation (5) and is incapable of evaluating the layer-wise sensitivity of gradient quantization.

To address these problems, we propose a novel sensitivity measurement using the product of magnitudes of gradients and quantization loss for weights, activations, and gradients for layer-wise quantization during low-bit training.

3.3.2. Magnitude-based Sensitivity Measurement

We begin with Proposition 3.1 to demonstrate the impact of quantization on the training loss.

Proposition 3.1. *The following statements hold for layer-wise quantization with the proposed magnitude-based sensitivity measurement.*

i) For weight or activation quantization (d is w or a) in the i-th and j-th layers, $\Delta L_d^i \leq \Delta L_d^j$, if

$$g_{d_q^i}^T \cdot \Delta d^i \leq g_{d_q^j}^T \cdot \Delta d^j. \quad (6)$$

ii) For gradient quantization (d is g), $\Delta L_g^i \leq \Delta L_g^j$, if

$$g_{w_q^i}^T \cdot \Delta g_{w_q^i} \leq g_{w_q^j}^T \cdot \Delta g_{w_q^j}. \quad (7)$$

Proof. Please refer to Appendix A.1. □

We then elaborate on the sensitivity measurement for quantizing weights, activations, and gradients.

1) Sensitivity Measurement for Weights and Activations. To efficiently estimate the upper bound and avoid computation overload, we use the Cauchy–Schwarz inequality:

$$\Delta L_d^i \leq \|g_{d_q^i}^T\|_2 \cdot \|\Delta d^i\|_2 \leq ||g_{d_q^i}^T||_1 \cdot ||\Delta d^i||_1. \quad (8)$$

For a fair comparison between layers, we divide the norm terms by the parameter number n_d^i to define the weight sensitivity s_d^i. The measurement is averaged over all the batches in the dataset to get $\overline{s_d^i}$. The calculation of s_d^i ($d \in \{w, a\}$) introduces few floating-point multiplications.

$$s_d^i = \frac{\|g_{d_q^i}^T\|_1}{n_d^i} \cdot \frac{\|\Delta d^i\|_1}{n_d^i} = \mathbf{E}(|g_{d_q^i}^T|) \cdot \mathbf{E}(|\Delta d^i|). \quad (9)$$

2) Sensitivity Measurement for Gradients. Similar to Equation (8), we have: $\Delta L_g^i \leq \|g_{w_q^i}^T\|_2 \cdot \eta \|\Delta g_{w_q^i}\|_2$. As we only have access to Δg_{o^i} (gradient quantization loss before

backward computation of the i-th layer), $\Delta g_{w_q^i}$ needs further calculation and is computed differently based on whether it is a linear layer or a convolutional layer.

For the linear layers, the weight gradient deviation resulting from the gradient quantization can be calculated with matrix multiplication: $\Delta g_{w_q^i} = \Delta g_{o^i} \cdot \mathbf{X}^T$. For the convolutional layers, the weight gradient deviation is calculated with the convolution calculation: $\Delta g_{w_q^i} = \mathbf{X}' \otimes \Delta g_{o^i}$. Through the derivation in Appendix A.2, we arrive at Equation (10) which applies to both linear layers and convolutional layers. Notably, the terms on the right side of Equations (8) and (10) are expanded into column vectors.

$$\Delta L_g^i \leq \|g_{w_q^i}^T\|_1 \cdot \eta \|\Delta g_{o^i}\|_1 \cdot \|\mathbf{X}\|_1. \tag{10}$$

Similarly, we divide the three norms in Equation (10) by the parameter numbers to get the gradient sensitivity:

$$s_g^i = \mathbf{E}(|g_{w_q^i}^T|) \cdot \mathbf{E}(|\Delta g_{o^i}|) \cdot \mathbf{E}(|\mathbf{X}|). \tag{11}$$

Algorithm 1 AMPA training framework.

Input: The initialized model, the number of epochs N, the update interval Itv, the fair allocation threshold Thr.
Output: The trained model with mixed-precision weights and activations.

```
Intialize the empty taboo lists TL_w, TL_a and TL_g.
for i = 1 to N do
    if i%Itv == 0 then
        Record the quantization loss E(|Δw^i|), E(|Δa^i|) and the activations E(|X|) in forward propagation.
        Record the gradients E(|g_{w_q^i}^T|), E(|g_{a_q^i}^T|) and the gradient quantization loss E(|Δg_{o^i}|) in the backward propagation.
        Calculate the average layer-wise sensitivity s_w^i, s_a^i and s_g^i over batches using Equation (9), (11).
        Sort the layers based on the sensitivity s_w^i, s_a^i and s_g^i, respectively.
        Delete layers in taboo lists from the sorted results.
        Increase the bitwidths of top α%, β% and γ% layers by Δbits sorted by s_w^i, s_a^i and s_g^i respectively.
        Add layers with selection times reaching Thr into the corresponding taboo list.
    end if
end for
```

3.4. Mixed Precision Allocation Scheme

Overall Allocation Scheme. The comprehensive scheme is outlined in Algorithm 1 and visually depicted in Figure 3. Initially, all layer bitwidths are set to INT4. At regular intervals of every Itv epochs, the bitwidths of weights, activations and gradients are updated. Specifically, the bitwidths of the top $\alpha\%$, $\beta\%$, and $\gamma\%$ most sensitive layers are elevated to higher levels (INT4 $\rightarrow$ INT6, INT6 $\rightarrow$ INT8). Here, α, β and γ are the update layer ratio hyperparameters that enable control over the compression rate. According to the varying degrees of influence brought about by the weight, activation and gradient quantization loss (Table 2), it is reasonable to set $\alpha < \beta < \gamma$.

Threshold for Fair Allocation. In specific training occasions, certain layers may keep occupying the opportunities for increasing bitwidths even after reaching INT8 (which is the highest bitwidth in the scheme). This situation can result in some other sensitive layers not being selected due to their relatively lower ranking. To ensure a fair allocation across layers, we introduce a threshold for the maximum selection times, denoted as Thr. When a layer has been selected for increasing bitwidths for Thr times after it reaches INT8, it will be included in the 'taboo' list, indicated in the bottom right corner of Figure 3. Consequently, it will no longer be eligible for selection to increase bitwidths.

4. Experiments

In this section, we begin with conducting experiments to validate the effectiveness of the proposed sensitivity measurement. Then, we explain how to determine the layer update ratio. Subsequently, we evaluate the efficacy of the proposed framework across different networks using various datasets. Finally, ablation studies and acceleration results are given to verify the feasibility of the method.

4.1. Evaluation Metrics

In our experiments, (a) the model performance and (b) the BitOPs reduced ratio (RR) are used as evaluation metrics. BitOPs (Yang & Jin, 2021) serves as an effective measure to approximate computation costs. The 'BOPS' denotes the BitOPs of training quantized parts that occupy most of the original networks' MACs.

For the multiplication of a single weight value of k_w bits and a single activation value of k_a bits, the BitOPs is $k_w k_a$. Under the same bitwidth, the BitOPs for the forward calculation, weight gradient computation, and activation gradient computation are approximately equal. For example, as the forward calculation of convolutional layers in ResNet-20 on CIFAR-10 has $40.11M$ MACs, the mean BitOPs for one forward and backward computation under full INT8 training is about $40.11M \times 8 \times 8 \times 3 = 7.70G$.

4.2. Verification of the Proposed Measurement

We assess the effectiveness of the sensitivity measurement as follows: **(i)** Firstly, we compare our proposed measurement with the distribution-based measurement $|\sum_i^n |x_f| - \sum_i^n |x_q||/\sum_i^n |x_f|$ used in (Zhang et al., 2020) and the

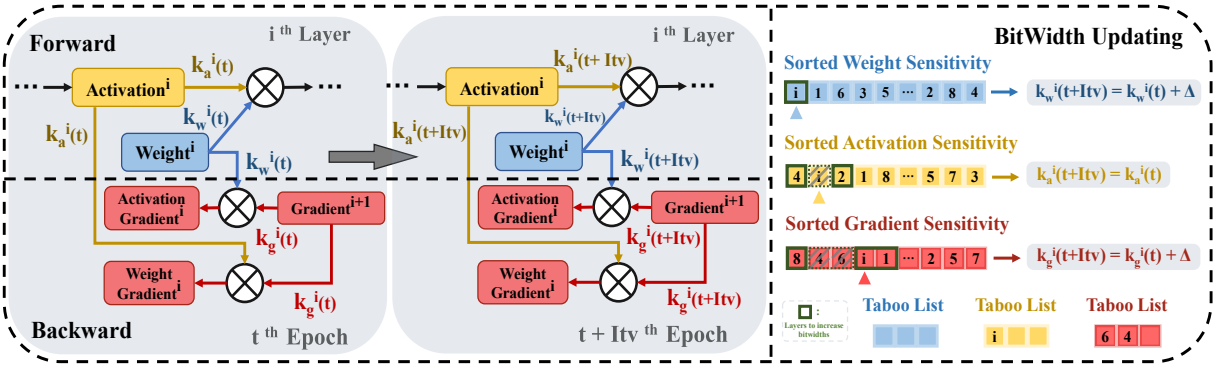


Figure 3: Overview of the proposed AMPA training framework. k_w^i, k_a^i, and k_g^i denote the bitwidths of weight, activation, and gradient quantization in the i-th layer respectively. The layer-wise sensitivity of weights, activations, and gradients is calculated every Itv epochs. The layers with the top sensitivity will increase the bitwidths by Δ which is 2 here. If a layer is selected for a certain number of times after reaching INT8, it is added to the taboo list and leaves chances to other layers.

Table 3: Comparison between the distribution-based measurement used in (Zhang et al., 2020), Hessian matrix trace (Dong et al., 2020) and our proposed measurement.

Model	Dataset	Measurement	Acc (%)	RR (%)
ResNet-20	CIFAR-10	Dist	92.25	41.19
		Hessian	91.35	50.42
		Ours(w/o Mag)	91.97	43.84
		Ours(w Mag)	92.26	41.59
ResNet18	CIFAR-100	Dist	74.26	41.48
		Hessian	73.72	52.73
		Ours(w/o Mag)	73.73	46.05
		Ours(w Mag)	74.53	44.42
MobileNetV2	CIFAR-100	Dist	70.12	34.54
		Hessian	67.51	52.45
		Ours(w/o Mag)	69.71	42.42
		Ours(w Mag)	70.32	44.66

Table 4: Verification of the scaling for the weight and activation sensitivity derivation.

Model	Dataset	AIR_w (%)	AIR_a (%)
ResNet-20	CIFAR-10	83.75	55.42
ResNet-56		93.97	79.53
MobileNetV2		83.50	56.50
ResNet-18	CIFAR-100	81.17	54.67
MobileNetV2		87.41	57.55

Hessian trace measurement $Tr(\nabla^2_{w/a} L(w/a))/n$ proposed in (Dong et al., 2020). The former is used for mixed gradient bitwidths, while the latter measurement is employed for mixed bitwidths selection for weights and activations. The results of this comparison are illustrated in Table 3. Using Hessian trace measurement for w/a results in a consistent selection of some layers, which leads to a higher compression rate but the performance loss is significant. Our measurement can achieve satisfactory compression rates with better performance in most cases. Moreover, when only the quantization loss is used as the measurement, there will be a noticeable performance decline with a close reduction ratio, manifesting the effectiveness of the magnitude factor of the gradients and activations.

(ii) Next, we compare the $g^T_{w^i_q} \cdot \Delta w^i$ with $s^i_w = \mathbf{E}(|g^T_{w^i_q}|) \cdot \mathbf{E}(|\Delta w^i|)$ for weight quantization and compare the $g^T_{a^i_q} \cdot \Delta a^i$ with $s^i_a = \mathbf{E}(|g^T_{a^i_q}|) \cdot \mathbf{E}(|\Delta a^i|)$ for activation quantization. We calculate the average intersection ratio (AIR) of the top 30% sensitive layers calculated by our measurement and the measurement before scaling. For example, if the average intersection ratio is 50%, it means that on average half of the top 30% sensitive layers obtained by the two measurements are consistent during training. As depicted in Table 4, the AIR surpasses 50% across all test scenarios, serving as a clear indication of the rationality of inequality scaling and the effectiveness of the sensitivity measurement.

4.3. Selection for the Layer Update Ratio

This section elucidates the process of selecting the layer update ratio $\alpha\%$, $\beta\%$, and $\gamma\%$. Before training, the range of the BitOPs reduced ratio (RR) can be estimated based on α, β, γ, Itv, and the computation cost of each layer. The theoretical maximum BitOPs reduction can be achieved by increasing the layer bitwidths with the least impact on BitOPs at each update. For example, the BitOPs reduction ratio of training ResNet-20 on CIFAR-10 with $N = 100$ epochs is in the range of 35.42%–53.09% under $\alpha = 10$, $\beta = 20$, $\gamma = 30$, $\text{Itv} = 5$; and 16.55%–31.42% under $\alpha = 20$, $\beta = 40$, $\gamma = 60$, $\text{Itv} = 5$.

Empirically, $\alpha : \beta : \gamma = 1 : 2 : 3$ is a suitable choice and the model performance is not sensitive to the value selection of α, β, and γ as shown in Appendix D. On a new task, we

Table 5: Top-1 classification accuracy (%) on CIFAR-10.

Model	Method	Top-1 Acc (%)		BOPS	RR (%)
		FP32	Quant		
ResNet-20	UI8	92.32	91.95	7.70G	0
	DAI8	92.35	92.76		
	AMPA	92.63	92.26	4.50G	41.59
MobileNetV2	UI8	94.39	93.38	16.20G	0
	DAI8	94.73	94.37		
	AMPA	95.01	94.62	8.78G	45.79
InceptionV3	UI8	94.89	95.00	650.1G	0
	DAI8	95.00	95.21		
	AMPA	95.58	94.56	370.2G	43.06
ResNet-32	UI8	-	93.75	13.14G	0
	AMPA	93.74	93.39	8.08G	38.49
ResNet-56	UI8	-	94.04	24.00G	0
	AMPA	94.23	93.63	14.21G	40.81
ViT	UI8	-	83.23	117.8G	0
	AMPA	83.68	82.34	76.13G	35.37

Table 6: Top-1 classification accuracy (%) on CIFAR-100.

Model	Method	Top-1 Acc (%)		BOPS	RR (%)
		FP32	Quant		
ResNet-18	UI8	-	75.49	106.3G	0
	AMPA	75.83	74.53	59.08G	44.42
MobileNetV2	UI8	-	70.91	12.00G	-
	AMPA	70.57	70.32	6.64G	44.66
InceptionV3	UI8	-	78.46	650.1G	-
	AMPA	79.01	78.71	371.5G	42.85

Table 7: Top-1 classification accuracy (%) on ImageNet.

Model	Method	Top-1 Acc (%)		BOPS	RR (%)
		FP32	Quant		
ResNet-18	WAGEUBN	68.70	67.40	325.1G	0
	UI8	70.30	69.67		
	DAI8	70.22	70.21		
	AMPA	70.52	69.58	171.0G	47.40
ResNet-34	WAGEUBN	71.99	68.50	681.5G	0
	UI8	73.68	73.29		
	DAI8	73.46	73.40		
	AMPA	73.57	72.55	375.9G	44.84
ResNet-50	WAGEUBN	74.66	69.07	761.9G	0
	UI8	76.60	76.34		
	DAI8	76.50	76.59		
	AMPA	76.60	74.94	451.7G	40.71
MobileNetV2	WAGEUBN	71.99	68.50	51.44G	0
	UI8	72.39	71.20		
	DAI8	72.44	71.92		
	AMPA	72.19	69.62	29.83G	42.01
ViT-S	UI8	-	76.92	803.7G	0
	AMPA	79.90	75.54	489.6G	39.08

can set the $Itv = 0.05$, $Thr = 3$, $\alpha : \beta : \gamma = 1 : 2 : 3$. The values of α, β, and γ can be adjusted according to the expected compression rate.

4.4. Training Results

The proposed adaptive mixed precision allocation (AMPA) training framework employs consistent hyperparameters: the fair allocation threshold is $\mathrm{Thr} = 3$, the update frequency(Itv/N) is set to 0.05 and the layer update ratio α, β, and γ are 10,20 and 30 for weights, activations and gradients respectively. The bitwidths are selected among **INT4**, **INT6**, and **INT8**. We employ the symmetric uniform quantization for weights and gradients while adopting asymmetric uniform quantization for activations.

4.4.1. Image Classification

We experiment with ResNet-18/20/32/56 (He et al., 2016), MobileNetV2 (Sandler et al., 2018), InceptionV3 (Szegedy et al., 2016), and ViT (Dosovitskiy et al., 2021) on CIFAR-10/100, and ResNet-18/34/50 (He et al., 2016), MobileNetV2 (Sandler et al., 2018) and ViT-S (Dosovitskiy et al., 2021) on ImageNet.

CIFAR-10. The training process consists of 200 epochs. As shown in Table 5, we achieve satisfactory accuracy compared to UI8 (Unified INT8) (Zhu et al., 2020) and DAI8 (Distributive adaptive INT8) (Zhao et al., 2021) with accuracy loss less than 1%.

CIFAR-100. The number of training epochs is also set to 200. Since the top-1 accuracy for CIFAR-100 is not provided in (Zhu et al., 2020; Zhao et al., 2021), we reproduce UI8 (Zhu et al., 2020) to obtain the top-1 accuracy on CIFAR-100. The proposed method achieves comparable top-1 accuracy with more than 40% BitOPs reduction. The training curves are shown in Appendix B and the bitwidth change processes are listed in Appendix C.

ImageNet. The number of training epochs is 100 for ResNet-18/34/50, 200 for MobileNetV2, and 300 for ViT-Small. Table 7 presents the top-1 accuracy of different quantized methods on ImageNet. It is worth mentioning that, despite DAI8 (Zhao et al., 2021) yields enhanced model performance, it relies on channel-wise gradient quantization and requires two separate quantizations for weight and activation gradient calculations. Our method attains satisfactory accuracy while achieving a higher compression ratio compared to full INT8 quantization.

Notably, after the training, we obtain a mixed precision model with weight storage reduced by more than 40% and smaller inference BitOPs with a reduction of over 30%, as shown in Table 8.

4.4.2. Image Segmentation

We further conduct experiments on the image segmentation task. We train the popular segmentation network U-

Table 8: The reduction of weight average bitwidths and inference BitOPs for the trained model.

Model	Dataset	Method	W-bits	Inf BOPS
ResNet-20	CIFAR-10	UI8	8.0	2.57G
		AMPA	4.20	1.62G (-36.76%)
MobileNetV2	CIFAR-100	UI8	8.0	4.00G
		AMPA	4.21	2.43G (-39.30%)
ResNet-50	ImageNet	UI8	8.0	254.0G
		AMPA	4.41	168.5G (-33.66%)

Table 9: Dice similarity score (%) for image segmentation using U-Net on Dsb2018 and Kvasir.

Dataset	Method	Dice (%)		BOPS	RR (%)
		FP32	Quant		
Dsb2018	UI8	-	91.41	1240.5G	-
	AMPA	91.72	91.24	734.1G	40.82
Kvasir	UI8	-	83.58	2203.2G	-
	AMPA	83.60	82.97	1125.2G	48.93

Net (Ronneberger et al., 2015) on the Dsb2018 dataset and Kvasir dataset (Jha et al., 2020) with 100 epochs. The image size is chosen at 96 and 128 for the two datasets, respectively. Dice similarity coefficient (DSC) is utilized to measure the model performance. Table 9 shows that our mixed precision training method achieves acceptable performance with over 40% reduction of BitOPs on the image segmentation task.

4.4.3. Language Modeling

For the language modeling task, we choose the widely known Transformer network (Vaswani et al., 2017) and train it on the Wikitext-2 (Merity et al., 2017), Wikitext-103 (Merity et al., 2017) and Penn Treebank datasets (Marcus et al., 1993) for 100 epochs. The transformers used consist of 6 layers for Wikitext-2, 16 layers for Wikitext-103 and 12 layers for Penn Treebank. Table 10 shows that the proposed method achieves satisfactory performance with about 40% reduction of BitOPs on the language modeling task.

4.5. Ablation Studies

Effects of the mixed precision allocation scheme. We compare the proposed mixed precision training framework with full INT6 quantization. As presented in Figure 4, our method outperforms full INT6 quantization, which results in a noticeable drop in performance. Additionally, our method works for any bitwidth setting, including mixed INT4 and INT8 quantization, which ensures broader compatibility. By setting the update Itv 0.1 for Mixed-48, we can achieve acceptable results as indicated in Figure 4.

Effects of the update interval. The update interval Itv in the training framework (Algorithm 1) impacts the compression rate. In Table 11, 'Update Itv' refers to dividing the interval by the total number of epochs: Itv/N. Table 11 vividly shows that too low update frequency will significantly impair the model performance and too high update frequency will yield a low compression rate. An interval of 0.05 appears to provide a balanced trade-off.

Table 10: Perplexity for language modeling using transformer on Wikitext-2, Wikitext-103 and Penn Treebank.

Dataset	Method	PPL		BOPS	RR (%)
		FP32	Quant		
Wikitext-2	UI8	-	120.36	349.2G	0
	AMPA	119.20	121.34	198.6G	43.14
Wikitext-103	UI8	-	25.56	2950.57G	0
	AMPA	25.00	27.68	1575.90G	46.59
Penn Treebank	UI8	-	97.25	698.5G	0
	AMPA	96.67	97.33	422.6G	39.50

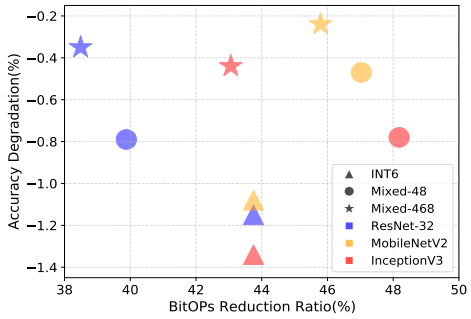


Figure 4: Comparison between full INT6 training and our mixed precision allocation method on CIFAR-10.

Effects of the layer update ratio. The proportion of updated layers that increase bitwidths in comparison to all layers influences the compression rate. Different layer update ratios are assigned for weights, activations, and gradients, denoted as $\alpha\%$, $\beta\%$, and $\gamma\%$ respectively. From Table 12, it is evident that setting $\alpha > \beta > \gamma$ is harmful to the model performance. Additionally, overly small or large update ratios are unfavorable, leading to performance degradation or compression rate reduction respectively. $\alpha = 10$, $\beta = 20$, $\gamma = 30$ can strike a good balance between the model performance and compression rate.

Effects of the fair allocation threshold. As illustrated in Table 13, in some networks like ResNet-56, the absence of a fair allocation threshold ($\mathrm{Thr} = \infty$) leads to substantial performance degradation or even training crashes, manifesting the necessity of the fair allocation threshold. Moreover, directly incorporating the INT8 layer into the taboo list ($\mathrm{Thr} = 0$) results in a notable reduction of the reduced ratio. $\mathrm{Thr} = 3$ achieves a better trade-off than 1 and 5. A more comprehensive ablation study can be found in Appendix E.

Table 11: Effects of the update interval Itv in the framework.

Model	Dataset	Update Itv	Acc (%)	RR (%)
ResNet-20	CIFAR-10	0.02	92.55	18.24
		0.05	92.26	41.59
		0.10	91.77	55.67
ResNet-18	CIFAR-100	0.02	75.16	21.94
		0.05	74.53	44.42
		0.10	NaN	61.06
MobileNetV2	CIFAR-100	0.02	70.31	20.34
		0.05	70.32	44.66
		0.10	69.37	57.66

Table 12: Effects of the layer update ratio α, β and γ in the training framework.

Model	Dataset	α	β	γ	Acc (%)	RR (%)
ResNet-20	CIFAR-10	30	20	10	91.07	42.34
		5	10	15	91.58	61.94
		10	20	30	92.26	41.59
		20	40	60	92.30	28.18
ResNet-18	CIFAR-100	30	20	10	NaN	50.47
		5	10	15	72.33	63.63
		10	20	30	74.53	44.42
		20	40	60	74.58	27.40
MobileNetV2	CIFAR-100	30	20	10	69.33	44.13
		5	10	15	69.30	58.57
		10	20	30	70.32	44.66
		20	40	60	70.31	30.59

Table 13: Effects of the fair allocation threshold. $\mathrm{Thr} = \infty$ denotes without 'fair allocation threshold'.

Model	Dataset	Thr	Acc (%)	RR (%)
ResNet-56	CIFAR-10	0	93.40	30.45
		1	93.27	35.92
		3	93.63	40.81
		5	NaN	43.88
		∞	NaN	46.64
MobileNetV2	CIFAR100	0	70.53	36.00
		1	70.32	40.82
		3	70.32	44.46
		5	70.19	46.73
		∞	70.04	50.48

4.6. Acceleration Results

We simulate the training process on the FPGA device with the proposed AMPA training framework. Table 14 illustrates the results of applying our method and full INT8 training (Zhu et al., 2020) to train ResNet-20 on CIFAR-10. The batch size is 64 and the FPGA chip is selected as xc7vx485tffg1157. Overall, our method can achieve a 19.16% latency reduction compared to INT8 training.

Table 14: Training acceleration of our method on ResNet-20 compared with full INT8 training (Zhu et al., 2020).

	conv0	conv8	conv15	Overall
Forward	7.11%	26.51%	27.43%	23.36%
Backward	5.12%	18.84%	21.76%	17.06%

5. Conclusion

In this paper, we present the novel adaptive mixed precision allocation (AMPA) integer training framework based on the observed characteristics of low-bit training. We put forward the effective layer-wise magnitude-based sensitivity measurements for the weight, activation, and gradient quantization respectively. The layer-wise bitwidths are adaptively updated at intervals based on the sensitivity comparison. The proposed method achieves satisfactory performance with a much more compression rate compared to full INT8 training. Comprehensive experiments across various models and datasets confirm the versatility of our method.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62125109, Grant 61931023, Grant 61932022, Grant 62371288, Grant 62320106003, Grant 62301299, Grant T2122024, Grant 62120106007, Grant 62250055.

Impact Statement

This paper presents work aimed at progressing the Machine Learning field. There are very few potential societal consequences of our work, and none of them must be specifically highlighted here.

References

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Cambier, L., Bhiwandiwalla, A., Gong, T., Nekuii, M., Elibol, O. H., and Tang, H. Shifted and squeezed 8-bit floating point format for low-precision training of deep neural networks. In *The 8th International Conference on Learning Representations*, Online, 2020. OpenReview.net.

Chang, S.-E., Li, Y., Sun, M., Jiang, W., Liu, S., Wang, Y., and Lin, X. RMSMP: A novel deep neural network quantization framework with row-wise mixed schemes and multiple precisions. In *Proceedings of the IEEE/CVF*

International Conference on Computer Vision, pp. 5251–5260, 2021.

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

Ding, S., Zhao, P., Zhang, X., Qian, R., Xiong, H., and Tian, Q. Prune spatio-temporal tokens by semantic-aware temporal accumulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16945–16956, 2023.

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

Dong, Z., Yao, Z., Arfeen, D., Gholami, A., Mahoney, M. W., and Keutzer, K. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. In *Advances in Neural Information Processing Systems 33*, pp. 18518–18529, 2020.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *The 9th International Conference on Learning Representations*, 2021.

Elthakeb, A. T., Pilligundla, P., Mireshghallah, F., Yazdanbakhsh, A., and Esmaeilzadeh, H. ReLeQ: A reinforcement learning approach for deep quantization of neural networks. In *NeurIPS Workshop on ML for Systems 2018*, 2018.

Fu, Y., Guo, H., Li, M., Yang, X., Ding, Y., Chandra, V., and Lin, Y. CPT: Efficient deep neural network training via cyclic precision. In *The 9th International Conference on Learning Representations*. OpenReview.net, 2021.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1737–1746. PMLR, 2015.

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

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks. In *Advances in Neural Information Processing Systems 29*, pp. 4107–4115, 2016.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Jain, S., Gural, A., Wu, M., and Dick, C. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. *Proceedings of Machine Learning and Systems*, 2:112–128, 2020.

Jha, D., Smedsrud, P. H., Riegler, M. A., Halvorsen, P., de Lange, T., Johansen, D., and Johansen, H. D. Kvasir-SEG: A segmented polyp dataset. In *MultiMedia Modeling*, pp. 451–462. Springer, 2020.

Kim, S., Shen, S., Thorsley, D., Gholami, A., Kwon, W., Hassoun, J., and Keutzer, K. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 784–794, 2022.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Li, F., Zhang, B., and Liu, B. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Mellempudi, N., Srinivasan, S., Das, D., and Kaul, B. Mixed precision training with 8-bit floating point. *arXiv preprint arXiv:1905.12334*, 2019.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. In *The 5th International Conference on Learning Representations*, 2017.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. XNOR-Net: ImageNet classification using binary convolutional neural networks. In *14th European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Ronneberger, O., Fischer, P., and Brox, T. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241. Springer, 2015.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and 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.

Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., and Villalobos, P. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the Inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Tang, C., Ouyang, K., Wang, Z., Zhu, Y., Ji, W., Wang, Y., and Zhu, W. Mixed-precision neural network quantization via learned layer-wise importance. In *17th European Conference on Computer Vision*, pp. 259–275. Springer, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pp. 5998–6008, 2017.

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

Wang, N., Choi, J., Brand, D., Chen, C.-Y., and Gopalakrishnan, K. Training deep neural networks with 8-bit floating point numbers. In *Advances in Neural Information Processing Systems 31*, pp. 7675–7684, Red Hook, NY, USA, 2018. Curran Associates, Inc.

Wu, B., Wang, Y., Zhang, P., Tian, Y., Vajda, P., and Keutzer, K. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018a.

Wu, S., Li, G., Chen, F., and Shi, L. Training and inference with integers in deep neural networks. In *The 6th International Conference on Learning Representations*, Online, 2018b. OpenReview.net.

Xi, H., Li, C., Chen, J., and Zhu, J. Training transformers with 4-bit integers. In *Advances in Neural Information Processing Systems 36*, pp. 49146–49168, 2023.

Yang, L. and Jin, Q. FracBits: Mixed precision quantization via fractional bit-widths. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 10612–10620, Menlo Park, CA, USA, 2021. AAAI Press.

Yang, Y., Deng, L., Wu, S., Yan, T., Xie, Y., and Li, G. Training high-performance and large-scale deep neural networks with full 8-bit integers. *Neural Networks*, 125: 70–82, 2020.

Yao, Z., Dong, Z., Zheng, Z., Gholami, A., Yu, J., Tan, E., Wang, L., Huang, Q., Wang, Y., Mahoney, M., and Keutzer, K. HAWQ-V3: Dyadic neural network quantization. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 11875–11886. PMLR, 2021.

Yu, H., Han, Q., Li, J., Shi, J., Cheng, G., and Fan, B. Search what you want: Barrier panelty NAS for mixed precision quantization. In *16th European Conference Computer Vision*, pp. 1–16. Springer, 2020.

Zhang, X., Liu, S., Zhang, R., Liu, C., Huang, D., Zhou, S., Guo, J., Guo, Q., Du, Z., Zhi, T., and Chen, Y. Fixed-point back-propagation training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2330–2338, Piscataway, NJ, USA, 2020. IEEE.

Zhao, K., Huang, S., Pan, P., Li, Y., Zhang, Y., Gu, Z., and Xu, Y. Distribution adaptive INT8 quantization for training CNNs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, Menlo Park, CA, USA, 2021. AAAI.

Zhe, W., Lin, J., Chandrasekhar, V., and Girod, B. Optimizing the bit allocation for compression of weights and activations of deep neural networks. In *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 3826–3830. IEEE, 2019.

Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Zhu, F., Gong, R., Yu, F., Liu, X., Wang, Y., Li, Z., Yang, X., and Yan, J. Towards unified INT8 training for convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1969–1979, Piscataway, NJ, USA, 2020. IEEE.

A. Proof of Proposition 3.1 and Equation (10)

A.1. Proof of Proposition 3.1

(a) For the quantization of $d \in \{w, a\}$ in the i-th layer $d_f^i \to d_q^i = d_f^i + \Delta d^i$, ΔL_d^i can be expressed as:

$$\Delta L_d^i = L(\cdots, d_f^i + \Delta d^i, \cdots) - L(\cdots, d_f^i, \cdots). \tag{12}$$

By applying Taylor's expansion, we can have Equation (13) where H^i denotes the Hessian matrix of the i-th layer:

$$\Delta L_d^i = g_{d_f^i}^T \cdot \Delta d^i + \frac{1}{2} \Delta {d^i}^T H^i \Delta d^i + o(\|\Delta d^i\|)^3. \tag{13}$$

In the training process, considering $g_{d_f^i}$ is not at zero and the quantization loss Δd^i is a small quantity (about the order of 1e-2 for INT4 and 1e-4 for INT8), the contribution of second-order term is negligible. By adopting the straight-through estimator (STE) (Bengio et al., 2013), we have $g_{d_q^i} \approx g_{d_f^i}$. Therefore, Equation (6) holds.

(b) The gradient quantization loss in i-th layer, denoted as $g_{o_f^i} \to g_{o_q^i} = g_{o_f^i} + \Delta g_{o^i}$ initially affects the weight update and then influences the loss function. As a result, the loss deviationΔL_g^i can be represented as:

$$\begin{aligned} \Delta L_g^i = L(\cdots, w_q^i + \eta(g_{w_f^i} + \Delta g_{w_q^i}), \cdots) \\ - L(\cdots, w_q^i + \eta g_{w_f^i}, \cdots). \end{aligned} \tag{14}$$

Similar to Equation (13), and considering that we can neglect the once update under a small learning rate, we have:

$$\Delta L_g^i \approx g_{w_q^i + \eta g_{w_q^i}}^T \cdot \eta \Delta g_{w_q^i} \approx g_{w_q^i}^T \cdot \eta \Delta g_{w_q^i}. \tag{15}$$

As a result, Equation (7) is valid.

A.2. Proof of Equation (10)

i) In the linear layers, supposing that the weight is denoted as $\mathbf{W} \in \mathbf{R}^{n \times m}$, the activation as $\mathbf{X} \in \mathbf{R}^{m \times 1}$ and the output as $\mathbf{O} \in \mathbf{R}^{n \times 1}$, the weight gradient deviation caused by the gradient quantization can be calculated with $\Delta g_{w_q^i} = \Delta g_{o^i} \cdot \mathbf{X}^T$. This yields:

$$\begin{aligned} \Delta L_g^i &\le ||g_{w_q^i}^T|| \cdot \eta ||\Delta g_{o^i} \cdot \mathbf{X}^T|| = ||g_{w_q^i}^T|| \cdot \eta \sqrt{\sum_{j=1}^{n} \sum_{k=1}^{m} (\Delta g_{o_j^i}^2 \mathbf{X}_k^2)} \\ &\le ||g_{w_q^i}^T|| \cdot \eta \sum_{j=1}^{n} \sum_{k=1}^{m} |\Delta g_{o_j^i}| |\mathbf{X}_k| = ||g_{w_q^i}^T|| \cdot \eta \sum_{j=1}^{n} |\Delta g_{o_j^i}| \cdot \sum_{k=1}^{m} |\mathbf{X}_k| \\ &\le \sum |g_{w_q^i}^T| \cdot \eta \sum_{j=1}^{n} |\Delta g_{o_j^i}| \cdot \sum_{k=1}^{m} |\mathbf{X}_k|. \end{aligned} \tag{16}$$

ii) For the convolutional layer, let us consider the weight denoted as $\mathbf{W} \in \mathbf{R}^{F \times F}$, the activation as $\mathbf{X} \in \mathbf{R}^{H_1 \times W_1}$, the output as $\mathbf{O} \in \mathbf{R}^{H_2 \times W_2}$. The weight gradient deviation is calculated with $\Delta g_{w_q^i} = \mathbf{X}' \otimes \Delta g_{o^i}$. For $\Delta g_{w_q^i}$, it is easy to have:

$$\begin{aligned} ||\Delta g_{w_q^i}|| &\le \sum_{j=1}^{F} \sum_{k=1}^{F} |\Delta g_{w_{q_{jk}}^i}| = ||\mathbf{X}_{11} \Delta g_{o_{11}^i} + \mathbf{X}_{12} \Delta g_{o_{12}^i} + \cdots|| + ||\mathbf{X}_{12} \Delta g_{o_{11}^i} + \mathbf{X}_{13} \Delta g_{o_{12}^i} + \cdots|| + \cdots \\ &\le ||\mathbf{X}_{11} \Delta g_{o_{11}^i}|| + ||\mathbf{X}_{12} \Delta g_{o_{12}^i}|| + \cdots + ||\mathbf{X}_{12} \Delta g_{o_{11}^i}|| + ||\mathbf{X}_{13} \Delta g_{o_{12}^i}|| + \cdots. \end{aligned} \tag{17}$$

The occurrence times of $\mathbf{X}_{jk}$ in the right-hand term of Equation (17) is dependent on the convolution operation. The middle element $\mathbf{X}_{H/2, W/2}$ has the most existing times which depend on the filter size F and the stride S. Only when the

convolution $\mathbf{X}' \otimes \Delta g_o^i$ is a 1*1 convolution, does the number of occurrence times of $\mathbf{X}_{jk}$ reach $H_2 \times W_2$. Therefore, it holds:

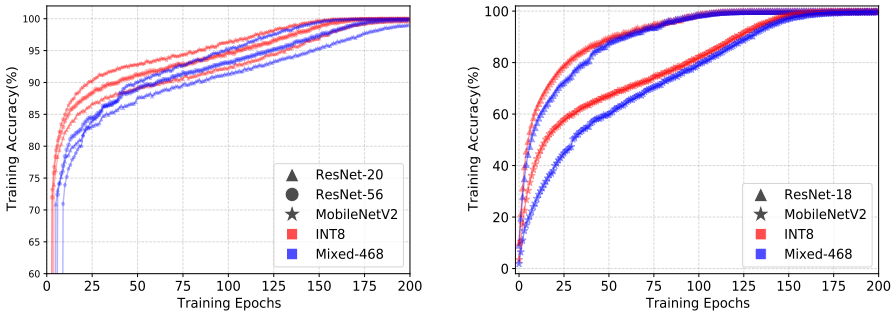
$$||\Delta g_{w_q^i}|| \leq \sum_{j=1}^{H_1}\sum_{k=1}^{W_1}\sum_{s=1}^{H_2}\sum_{t=1}^{W_2}|\mathbf{X}_{jk}||\Delta g_{o_{st}^i}| = \sum_{j=1}^{H_1}\sum_{k=1}^{W_1}|\mathbf{X}_{jk}|\sum_{s=1}^{H_2}\sum_{t=1}^{W_2}|\Delta g_{o_{st}^i}|. \tag{18}$$

The impact of the quantization on the loss function ΔL can be formulated as:

$$\Delta L \leq ||g_{w_q^i}^T|| \cdot \eta \sum_{j=1}^{H_1}\sum_{k=1}^{W_1}|\mathbf{X}_{jk}|\sum_{s=1}^{H_2}\sum_{t=1}^{W_2}|\Delta g_{o_{st}^i}| \leq \sum |g_{w_q^i}^T| \cdot \eta \sum_{j=1}^{H_1}\sum_{k=1}^{W_1}|\mathbf{X}_{jk}|\sum_{s=1}^{H_2}\sum_{t=1}^{W_2}|\Delta g_{o_{st}^i}|. \tag{19}$$

We quantize the linears layers in the models like Transformer, or quantize the convolutional layers in the models like ResNet-18. In our proposed scheme, there is no sensitivity comparison between linear layers and convolutional layers.

B. Training Curves

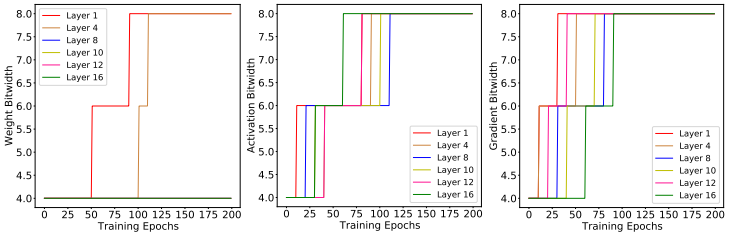


(a) CIFAR-10 (b) CIFAR-100

Figure 5: The training curves of INT8 training and Mixed-468 training on (a) CIFAR-10 and (b) CIFAR-100.

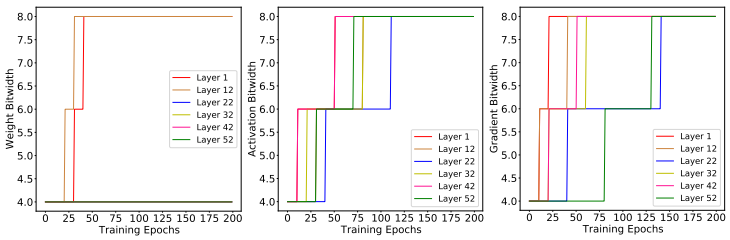
Figure 5 shows the training curve of the INT8 training and our proposed Mixed-468 training. It can be observed that the two curves are generally close, and there is a small accuracy degradation for Mixed-468 compared to INT8 training. This further illustrates the feasibility of our approach.

C. Bitwidth Change Process



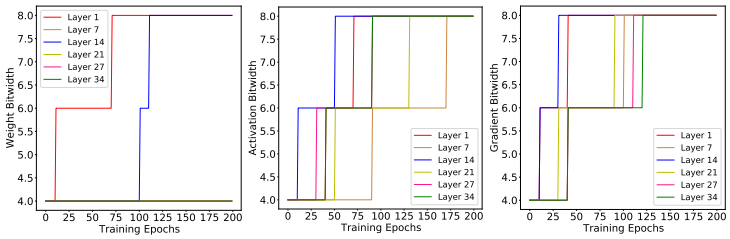
(a) Weights (b) Activations (c) Gradients

Figure 6: The bitwidth change process (INT4 $\rightarrow$ INT6, INT6 $\rightarrow$ INT8) of (a) weights, (b) activations, and (c) gradients through training ResNet-20 on CIFAR-10.



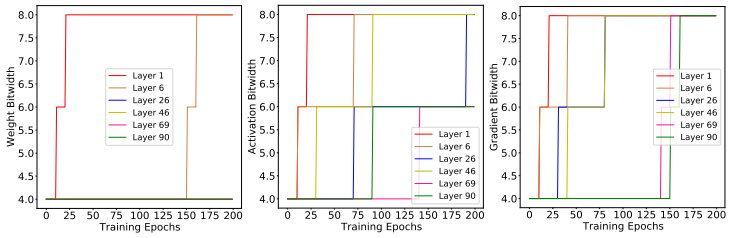
(a) Weights (b) Activations (c) Gradients

Figure 7: The bitwidth change process (INT4 → INT6, INT6 → INT8) of (a) weights, (b) activations, and (c) gradients through training ResNet-56 on CIFAR-10.



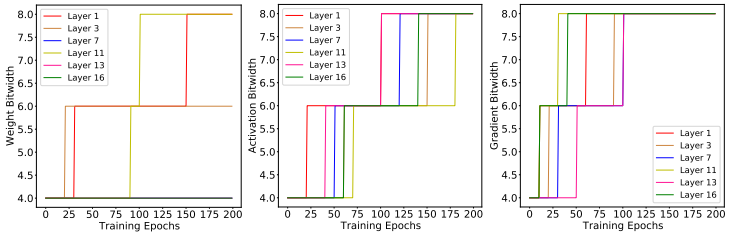
(a) Weights (b) Activations (c) Gradients

Figure 8: The bitwidth change process (INT4 → INT6, INT6 → INT8) of (a) weights, (b) activations, and (c) gradients through training MobileNetV2 on CIFAR-10.



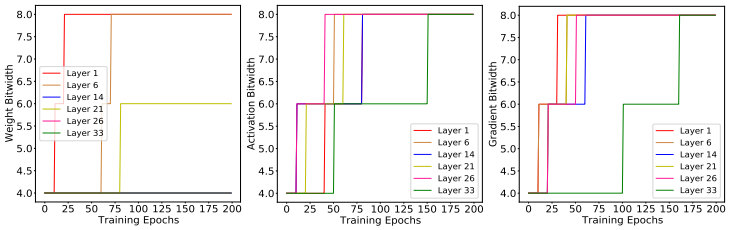
(a) Weights (b) Activations (c) Gradients

Figure 9: The bitwidth change process (INT4 → INT6, INT6 → INT8) of (a) weights, (b) activations, and (c) gradients through training InceptionV3 on CIFAR-10.



(a) Weights (b) Activations (c) Gradients

Figure 10: The bitwidth change process (INT4 → INT6, INT6 → INT8) of (a) weights, (b) activations, and (c) gradients through training ResNet-18 on CIFAR-100.

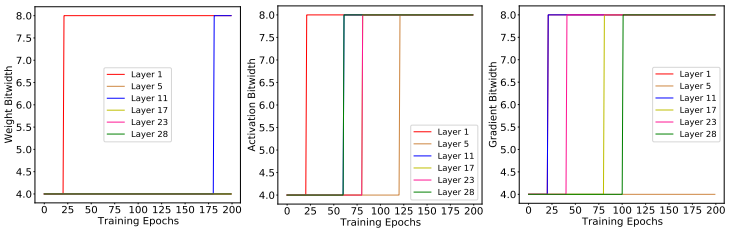


(a) Weights (b) Activations (c) Gradients

Figure 11: The bitwidth change process (INT4 → INT6, INT6 → INT8) of (a) weights, (b) activations, and (c) gradients through training MobileNetV2 on CIFAR-100.

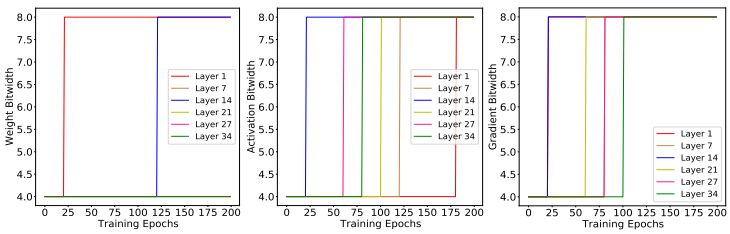
Figure 6-11 show the bitwidth change processes changing from INT4 to INT6 to INT8. The value of dividing the interval Itv by the total number of epochs N is 0.05. It can be seen from Figure 6-11 that the gradient bitwidths exhibit an earlier change, and weight bitwidths have a comparatively slower change. This pattern is consistent with our precision update strategy where the update layer ratios satisfy $\alpha < \beta < \gamma$. Moreover, the weight bitwidths in the shallower layers of the network are increased in the relatively earlier training stage. In some deeper layers, the weight bitwidths maintain INT4 throughout the training process.

Figure 12-14 illustrate the bitwidth change processes changing from INT4 to INT8. The value of dividing interval Itv by the total number of periods N is 0.1. Similarly, gradient bitwidths change relatively earlier and weight bitwidths change relatively slower.



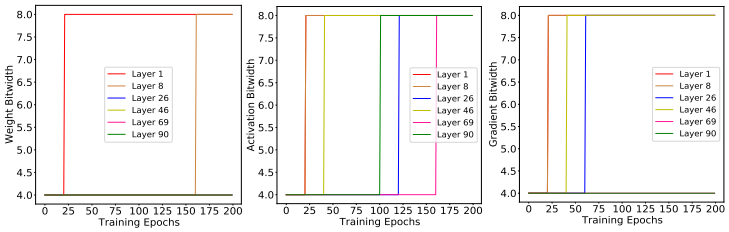
(a) Weights (b) Activations (c) Gradients

Figure 12: The bitwidth change process (INT4 → INT8) of (a) weights, (b) activations, and (c) gradients through training ResNet-32 on CIFAR-10.



(a) Weights (b) Activations (c) Gradients

Figure 13: The bitwidth change process (INT4 → INT8) of (a) weights, (b) activations, and (c) gradients through training MobileNetV2 on CIFAR-10.



(a) Weights (b) Activations (c) Gradients

Figure 14: The bitwidth change process (INT4 $\rightarrow$ INT8) of (a) weights, (b) activations, and (c) gradients through training InceptionV3 on CIFAR-10.

D. Selection of the layer update ratio α, β, and γ

Table 15: Effects of the value selection of α, β, and γ

Model	Dataset	Itv/N	0.05	0.05	0.05	0.05	0.05	0.05
ResNet-20	CIFAR-10	α	10	10	10	10	10	15
		β	15	20	25	20	25	20
		γ	25	30	30	40	40	30
		Acc(%)	91.84	92.26	92.07	92.67	92.18	91.96
		RR(%)	46.83	41.59	41.43	40.89	39.09	38.89
MobileNetV2	CIFAR-100	α	10	10	10	10	10	15
		β	15	20	25	20	25	20
		γ	25	30	30	40	40	30
		Acc(%)	70.12	70.32	70.42	70.75	70.38	70.32
		RR(%)	48.81	44.66	43.85	42.50	41.71	41.48

Table 15 shows that, under different values of $\alpha : \beta : \gamma$, the gap of model performance is 0.83% (ranging from 91.84% to 92.67%) for ResNet-20 on CIFAR-10 and 0.63% (ranging from 70.12% to 70.75%) for MobileNetV2 on CIFAR-100. These results demonstrate that the ratio $\alpha : \beta : \gamma$ does not evidently affect model performance. Remarkably, we set $\alpha : \beta : \gamma$ to 1:2:3 without specifically tuning the model (note that we can achieve better accuracy with $\alpha : \beta : \gamma = 1 : 2 : 4$).

E. Ablation Studies Results

Table 16: Effects of the update interval Itv in the training framework.

Model	Dataset	Update Freq (Itv/N)	Acc (%)	RR (%)
ResNet-20	CIFAR-10	0.02	92.55	18.24
		0.05	92.17	41.89
		0.10	91.77	55.67
ResNet-32	CIFAR-10	0.02	93.30	13.22
		0.05	93.39	38.49
		0.10	92.33	52.48
MobileNetV2	CIFAR-10	0.02	94.91	21.42
		0.05	94.62	45.79
		0.10	93.96	58.69
ResNet-18	CIFAR-100	0.02	75.16	21.94
		0.05	74.53	44.42
		0.10	NaN	61.06
MobileNetV2	CIFAR-100	0.02	70.31	20.34
		0.05	70.32	44.66
		0.10	69.37	57.66

Table 16-18 provide a more comprehensive presentation of the results in Table 11-13. Table 16 shows that too long update interval ($\mathrm{Thr}/N = 0.10$) will significantly impair the model performance and too short update interval ($\mathrm{Thr}/N = 0.02$) will yield a low compression rate. A balanced trade-off is achieved at $\mathrm{Thr}/N = 0.05$.

Table 17: Effects of the layer update ratio α, β and γ in the training framework.

Model	Dataset	α	β	γ	Acc (%)	RR (%)
ResNet-20	CIFAR-10	30	20	10	91.07	42.34
		5	10	15	91.58	61.94
		10	20	30	92.26	41.59
		20	40	60	92.30	28.18
ResNet-56	CIFAR-10	30	20	10	92.51	39.68
		5	10	15	93.05	55.43
		10	20	30	93.63	40.81
		20	40	60	93.88	26.67
MobileNetV2	CIFAR-10	30	20	10	93.02	45.05
		5	10	15	93.86	60.84
		10	20	30	94.62	45.79
		20	40	60	94.71	30.81
ResNet-18	CIFAR-100	30	20	10	NaN	50.47
		5	10	15	72.33	63.63
		10	20	30	74.53	44.42
		20	40	60	74.58	27.40
MobileNetV2	CIFAR-100	30	20	10	69.33	44.13
		5	10	15	69.30	58.57
		10	20	30	70.32	44.66
		20	40	60	70.31	30.59

Table 17 demonstrates that setting $\alpha > \beta > \gamma$ harms the model performance. Additionally, too small layer update ratio can lead to performance degradation, while too large layer update ratio can significantly reduce the compression rate. $\alpha = 10$, $\beta = 20$, $\gamma = 30$ is a good balanced choice.

Table 18: Effects of the fair allocation threshold. $\mathrm{Thr} = \infty$ denotes without 'fair allocation threshold'.

Model	Dataset	Thr	Acc (%)	RR (%)
ResNet-20	CIFAR-10	0	92.01	37.13
		1	92.10	39.80
		3	92.26	41.59
		5	92.25	43.82
		∞	92.12	45.10
ResNet-56	CIFAR-10	0	93.40	30.45
		1	93.27	35.92
		3	93.63	40.81
		5	NaN	43.88
		∞	NaN	46.64
MobileNetV2	CIFAR-10	0	94.33	36.93
		1	94.88	41.41
		3	94.62	45.79
		5	94.41	48.47
		∞	94.51	50.82
ResNet-18	CIFAR-100	0	74.43	39.71
		1	74.16	41.86
		3	74.53	44.42
		5	74.09	45.94
		∞	74.06	47.15
MobileNetV2	CIFAR-100	0	70.53	36.00
		1	70.32	40.82
		3	70.32	44.46
		5	70.19	46.73
		∞	70.04	50.48

As illustrated in Table 18, in some networks like ResNet-56 and ResNet-18, the absence of a fair allocation threshold ($\mathrm{Thr} = \infty$) will lead to substantial performance degradation or even training crashes, manifesting the necessity of the fair allocation threshold. Moreover, directly incorporating the INT8 layer into the taboo list ($\mathrm{Thr} = 0$) results in a notable reduction of the reduced ratio.

Table 19: Effects of different quantization granularity in the proposed method.

Model	Dataset	Granularity	Acc (%)	RR (%)
ResNet-20	CIFAR-10	block-wise	92.22	35.86
		layer-wise	92.26	41.59
ResNet-56	CIFAR-10	block-wise	93.31	37.30
		layer-wise	93.63	40.81
MobileNetV2	CIFAR-10	block-wise	94.56	42.26
		layer-wise	94.62	45.79
ResNet-18	CIFAR-100	block-wise	74.28	37.50
		layer-wise	74.53	44.42
MobileNetV2	CIFAR-100	block-wise	70.19	40.81
		layer-wise	70.32	44.66

The block-wise bitwidth allocation is also compatible with the proposed mixed-precision allocation. Table 19 compares the training results of block-wise and layer-wise allocation. It can be observed that layer-wise allocation can achieve higher performance at a higher compression rate. Additionally, considering that some models may not be appropriately divided into blocks, we select layer-wise bitwidth allocation in our method.