Nearly Lossless Adaptive Bit Switching

Anonymous Author(s)
Affiliation
Address
email

Abstract

Model quantization is widely applied for compressing and accelerating deep neural networks (DNNs). However, conventional Quantization-Aware Training (QAT) focuses on training DNNs with uniform bit-width. The bit-width settings vary across different hardware and transmission demands, which induces considerable training and storage costs. Hence, the scheme of one-shot joint training multiple precisions is proposed to address this issue. Previous works either store a larger FP32 model to switch between different precision models for higher accuracy or store a smaller INT8 model but compromise accuracy due to using shared quantization parameters. In this paper, we introduce the ***Double Rounding*** quantization method, which fully utilizes the quantized representation range to accomplish nearly lossless bit-switching while reducing storage by using the highest integer precision instead of full precision. Furthermore, we observe a competitive interference among different precisions during one-shot joint training, primarily due to inconsistent gradients of quantization scales during backward propagation. To tackle this problem, we propose an Adaptive Learning Rate Scaling (**ALRS**) technique that dynamically adapts learning rates for various precisions to optimize the training process. Additionally, we extend our *Double Rounding* to one-shot mixed precision training and develop a Hessian-Aware Stochastic Bit-switching (**HASB**) strategy. Experimental results on the ImageNet-1K classification demonstrate that our methods have enough advantages to state-of-the-art one-shot joint QAT in both multi-precision and mixed-precision. Our codes are available at here.

1 Introduction

Recently, with the popularity of mobile and edge devices, more and more researchers have attracted attention to model compression due to the limitation of computing resources and storage. Model quantization [1; 2] has gained significant prominence in the industry. Quantization maps floating-point values to integer values, significantly reducing storage requirements and computational resources without altering the network architecture.

Generally, for a given pre-trained model, the quantization bit-width configuration is predefined for a specific application scenario. The quantized model then undergoes retraining, *i.e.*, QAT, to mitigate the accuracy decline. However, when the model is deployed across diverse scenarios with different precisions, it often requires repetitive retraining processes for the same model. A lot of computing resources and training costs are wasted. To address this challenge, involving the simultaneous training of multi-precision [3; 4] or one-shot mixed-precision [3; 5] have been proposed. Among these approaches, some involve sharing weight parameters between low-precision and high-precision models, enabling dynamic bit-width switching during inference.

However, bit-switching from high precision (or bit-width) to low precision may introduce significant accuracy degradation due to the *Rounding* operation in the quantization process. Additionally, there is severe competition in the convergence process between higher and lower precisions in multi-precision

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

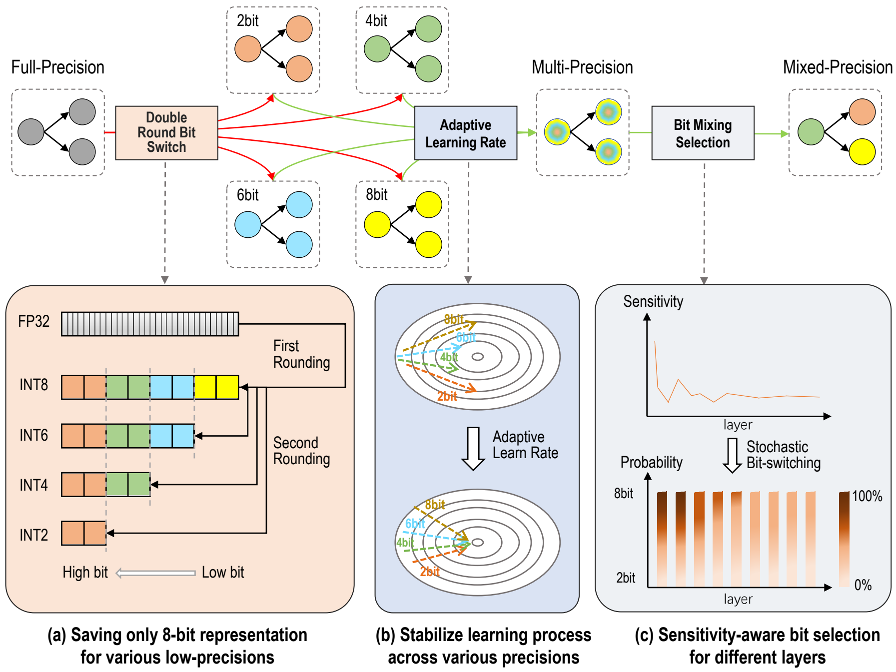


Figure 1: Overview of our proposed lossless adaptive bit-switching strategy.

scheme. In mixed-precision scheme, previous methods often incur vast searching and retraining costs due to decoupling the training and search stages. Due to the above challenges, bit-switching remains a very challenging problem. Our motivation is designing a bit-switching quantization method that doesn't require storing a full-precision model and achieves nearly lossless switching from high-bits to low-bits. Specifically, for different precisions, we propose unified representation, normalized learning steps, and tuned probability distribution so that an efficient and stable learning process is achieved across multiple and mixed precisions, as depicted in Figure 1.

To solve the bit-switching problem, prior methods either store the floating-point parameters [6; 7; 4; 8] to avoid accuracy degradation or abandon some integer values by replacing *rounding* with *floor*[3; 9] but leading to accuracy decline or training collapse at lower bit-widths. We propose *Double Rounding*, which applies the *rounding* operation twice instead of once, as shown in Figure1 (a). This approach ensures nearly lossless bit-switching and allows storing the highest bit-width model instead of the full-precision model. Specifically, the lower precision weight is included in the higher precision weight, reducing storage constraints.

Moreover, we empirically find severe competition between higher and lower precisions, particularly in 2-bit precision, as also noted in [10; 4]. There are two reasons for this phenomenon: The optimal quantization interval itself is different for higher and lower precisions. Furthermore, shared weights are used for different precisions during joint training, but the quantization interval gradients for different precisions exhibit distinct magnitudes during training. Therefore, we introduce an Adaptive Learning Rate Scaling (ALRS) method, designed to dynamically adjust the learning rates across different precisions, which ensures consistent update steps of quantization scales corresponding to different precisions, as shown in the Figure 1 (b).

Finally, we develop an efficient one-shot mixed-precision quantization approach based on *Double Rounding*. Prior mixed-precision approaches first train a SuperNet with predefined bit-width lists, then search for optimal candidate SubNets under restrictive conditions, and finally retrain or fine-tune them, which incurs significant time and training costs. However, we use the Hessian Matrix Trace [11] as a sensitivity metric for different layers to optimize the SuperNet and propose a Hessian-Aware Stochastic Bit-switching (HASB) strategy, inspired by the Roulette algorithm [12]. This strategy enables tuned probability distribution of switching bit-width across layers, assigning higher bits to more sensitive layers and lower bits to less sensitive ones, as shown in Figure 1 (c). And, we add the sensitivity to the search stage as a constraint factor. So, our approach can omit the last stage.

In conclusion, our main contributions can be described as:

- *Double Rounding* quantization method for multi-precision is proposed, which stores a single integer weight to enable adaptive precision switching with nearly lossless accuracy.
- Adaptive Learning Rate Scaling (ALRS) method for the multi-precision scheme is introduced, which effectively narrows the training convergence gap between high-precision and low-precision, enhancing the accuracy of low-precision models without compromising high-precision model accuracy.
- Hessian-Aware Stochastic Bit-switching (HASB) strategy for one-shot mixed-precision SuperNet is applied, where the access probability of bit-width for each layer is determined based on the layer's sensitivity.
- Experimental results on the ImageNet1K dataset demonstrate that our proposed methods are comparable to state-of-the-art methods across different mainstream CNN architectures.

2 Related Works

Multi-Precision. Multi-Precision entails a single shared model with multiple precisions by one-shot joint Quantization-Aware Training (QAT). This approach can dynamically adapt uniform bit-switching for the entire model according to computing resources and storage constraints. AdaBits [13] is the first work to consider adaptive bit-switching but encounters convergence issues with 2-bit quantization on ResNet50 [14]. Bit-Mixer [9] addresses this problem by using the LSQ [2] quantization method but discards the lowest state quantized value, resulting in an accuracy decline. Multi-Precision joint QAT can also be viewed as a multi-objective optimization problem. Any-precision [6] and MultiQuant [4] combine knowledge distillation techniques to improve model accuracy. Among these methods, MultiQuant's proposed "Online Adaptive Label" training strategy is essentially a form of self-distillation [15]. Similar to our method, AdaBits and Bit-Mixer can save an 8-bit model, while other methods rely on 32-bit models for bit switching. Our *Double Rounding* method can store the highest bit-width model (e.g., 8-bit) and achieve almost lossless bit-switching, ensuring a stable optimization process. Importantly, this leads to a reduction in training time by approximately 10% [7] compared to separate quantization training.

One-shot Mixed-Precision. Previous works mainly utilize costly approaches, such as reinforcement learning [16; 17] and Neural Architecture Search (NAS) [18; 19; 20], or rely on partial prior knowledge [21; 22] for bit-width allocation, which may not achieve global optimality. In contrast, our proposed one-shot mixed-precision method employs Hessian-Aware optimization to refine a SuperNet via gradient updates, and then obtain the optimal conditional SubNets with less search cost without retraining or fine-tuning. Additionally, Bit-Mixer [9] and MultiQuant [4] implement layer-adaptive mixed-precision models, but Bit-Mixer uses a naive search method to attain a sub-optimal solution, while MultiQuant requires 300 epochs of fine-tuning to achieve ideal performance. Unlike NAS approaches [20], which focus on altering network architecture (e.g., depth, kernel size, or channels), our method optimizes a once-for-all SuperNet using only quantization techniques without altering the model architecture.

3 Methodology

3.1 *Double Rounding*

Conventional separate precision quantization using Quantization-Aware Training (QAT) [23] attain a fixed bit-width quantized model under a pre-trained FP32 model. A pseudo-quantization node is inserted into each layer of the model during training. This pseudo-quantization node comprises two operations: the quantization operation $quant(x)$, which maps floating-point (FP32) values to lower-bit integer values, and the dequantization operation $dequant(x)$, which restores the quantized integer value to its original floating-point representation. It can simulate the quantization error incurred when compressing float values into integer values. As quantization involves a non-differentiable $Rounding$ operation, Straight-Through Estimator (STE) [24] is commonly used to handle the non-differentiability.

However, for multi-precision quantization, bit-switching can result in significant accuracy loss, especially when transitioning from higher bit-widths to lower ones, *e.g.*, from 8-bit to 2-bit. To

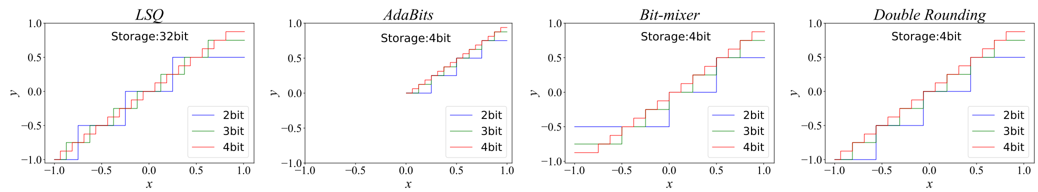


Figure 2: Comparison of four quantization schemes:(from left to right) used in *LSQ* [2], *AdaBits* [3], *Bit-Mixer* [9] and Ours *Double Rounding*. In all cases $y = dequant(quant(x))$.

mitigate this loss, prior works have mainly employed two strategies: one involves bit-switching from a floating-point model (32-bit) to a lower-bit model each time using multiple learnable quantization parameters, and the other substitutes the $Rounding$ operation with the $Floor$ operation, but this results in accuracy decline (especially in 2-bit). In contrast, we propose a nearly lossless bit-switching quantization method called *Double Rounding*. This method overcomes these limitations by employing a $Rounding$ operation twice. It allows the model to be saved in the highest-bit (*e.g.*, 8-bit) representation instead of full-precision, facilitating seamless switching to other bit-width models. A detailed comparison of *Double Rounding* with other quantization methods is shown in Figure 2.

Unlike AdaBits, which relies on the Dorefa [1] quantization method where the quantization scale is determined based on the given bit-width, the quantization scale of our *Double Rounding* is learned online and is not fixed. It only requires a pair of shared quantization parameters, *i.e.*, *scale* and *zero-point*. Quantization scales of different precisions adhere to a strict "Power of Two" relationship. Suppose the highest-bit and the target low-bit are denoted as h-bit and l-bit respectively, and the difference between them is $\Delta = h - l$. The specific formulation of *Double Rounding* is as follows:

$$\widetilde{W}_h = \text{clip}(\left\lfloor \frac{W - \mathbf{z}_h}{\mathbf{s}_h} \right\rceil, -2^{h-1}, 2^{h-1} - 1) \tag{1}$$

$$\widetilde{W}_l = \text{clip}(\left\lfloor \frac{\widetilde{W}_h}{2^{\Delta}} \right\rceil, -2^{l-1}, 2^{l-1} - 1) \tag{2}$$

$$\widehat{W}_l = \widetilde{W}_l \times \mathbf{s}_h \times 2^{\Delta} + \mathbf{z}_h \tag{3}$$

where the symbol $\lfloor . \rceil$ denotes the $Rounding$ function, and clip$(x, low, upper)$ means x is limited to the range between low and $upper$. Here, W represents the FP32 model's weights, $\mathbf{s}_h \in \mathbb{R}$ and $\mathbf{z}_h \in \mathbb{Z}$ denote the highest-bit (*e.g.*, 8-bit) quantization *scale* and *zero-point* respectively. $\widetilde{W}_h$ represent the quantized weights of the highest-bit, while $\widetilde{W}_l$ and $\widehat{W}_l$ represent the quantized weights and dequantized weights of the low-bit respectively.

Hardware shift operations can efficiently execute the division and multiplication by 2^{Δ}. Note that in our *Double Rounding*, the model can also be saved at full precision by using unshared quantization parameters to run bit-switching and attain higher accuracy. Because we use symmetric quantization scheme, the $\mathbf{z}_h$ is 0. Please refer to Section A.4 for the gradient formulation of *Double Rounding*.

Unlike fixed weights, activations change online during inference. So, the corresponding *scale* and *zero-point* values for different precisions can be learned individually to increase overall accuracy. Suppose X denotes the full precision activation, and $\widetilde{X}_b$ and $\widehat{X}_b$ are the quantized activation and dequantized activation respectively. The quantization process can be formulated as follows:

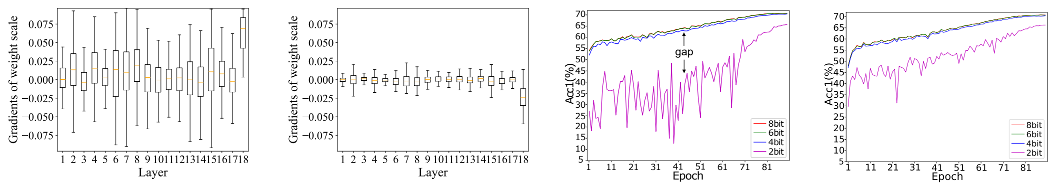
$$\widetilde{X}_b = \text{clip}(\left\lfloor \frac{X - \mathbf{z}_b}{\mathbf{s}_b} \right\rceil, 0, 2^b - 1) \tag{4}$$

$$\widehat{X}_b = \widetilde{X}_b \times \mathbf{s}_b + \mathbf{z}_b \tag{5}$$

where $\mathbf{s}_b \in \mathbb{R}$ and $\mathbf{z_b} \in \mathbb{Z}$ represent the quantization *scale* and *zero-point* of different bit-widths activation respectively. Note that $\mathbf{z}_b$ is 0 for the ReLU activation function.

3.2 Adaptive Learning Rate Scaling for Multi-Precision

Although our proposed *Double Rounding* method represents a significant improvement over most previous multi-precision works, the one-shot joint optimization of multiple precisions remains constrained by severe competition between the highest and lowest precisions [10; 4]. Different precisions simultaneously impact each other during joint training, resulting in substantial differences in convergence rates between them, as shown in Figure 3 (c). We experimentally find that this competitive relationship stems from the inconsistent magnitudes of the quantization scale's gradients between high-bit and low-bit quantization during joint training, as shown in Figure 3 (a) and (b). For other models statistical results please refer to Section A.6 in the appendix.



(a) 2-bit (b) 4-bit (c) w/o ALRS (d) w. ALRS

Figure 3: The statistics of ResNet18 on ImageNet-1K dataset. (a) and (b): The quantization scale gradients' statistics for the weights, with outliers removed for clarity. (c) and (d): The multi-precision training processes of our *Double Rounding* without and with the ALRS strategy.

Motivated by these observations, we introduce a technique termed Adaptive Learning Rate Scaling (ALRS), which dynamically adjusts learning rates for different precisions to optimize the training process. This technique is inspired by the Layer-wise Adaptive Rate Scaling (LARS) [25] optimizer. Specifically, suppose the current batch iteration's learning rate is λ, we set learning rates λ_b of different precisions as follows:

$$\lambda_b = \eta_b \left(\lambda - \sum_{i=1}^{L} \frac{\min\left(\text{max\_abs}\left(\text{clip\_grad}(\nabla \mathbf{s}_b^i, 1.0)\right), 1.0\right)}{L} \right), \tag{6}$$

$$\eta_b = \begin{cases} 1 \times 10^{-\frac{\Delta}{2}}, & \text{if } \Delta \text{ is even} \\ 5 \times 10^{-(\frac{\Delta+1}{2})}, & \text{if } \Delta \text{ is odd} \end{cases} \tag{7}$$

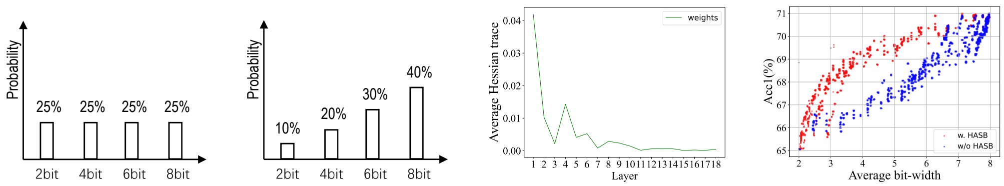
where the L is the number of layers, clip_grad$(.)$ represents gradient clipping that prevents gradient explosion, max_abs$(.)$ denotes the maximum absolute value of all elements. The $\nabla \mathbf{s}_b^i$ denotes the quantization scale's gradients of layer i and η_b denotes scaling hyperparameter of different precisions, *e.g.*, 8-bit is 1, 6-bit is 0.1, and 4-bit is 0.01. Note that the ALRS strategy is only used for updating quantization scales. It can adaptively update the learning rates of different precisions and ensure that model can optimize quantization parameters at the same pace, ultimately achieving a minimal convergence gap in higher bits and 2-bit, as shown in Figure 3 (d).

In multi-precision scheme, different precisions share the same model weights during joint training. For conventional multi-precision, the shared weight computes n forward processes at each training iteration, where n is the number of candidate bit-widths. The losses attained from different precisions are then accumulated, and the gradients are computed. Finally, the shared parameters are updated. For detailed implementation please refer to Algorithm A.1 in the appendix. However, we find that if different precision losses separately compute gradients and directly update shared parameters at each forward process, it attains better accuracy when combined with our ALRS training strategy. Additionally, we use dual optimizers to update the weight parameters and quantization parameters simultaneously. We also set the weight-decay of the quantization scales to 0 to achieve stable convergence. For detailed implementation please refer to Algorithm A.2 in the appendix.

3.3 One-Shot Mixed-Precision SuperNet

Unlike multi-precision, where all layers uniformly utilize the same bit-width, mixed-precision SuperNet provides finer-grained adaptive by configuring the bit-width at different layers. Previous methods typically decouple the training and search stages, which need a third stage for retraining or fine-tuning the searched SubNets. These approaches generally incur substantial search costs in selecting the optimal SubNets, often employing methods such as greedy algorithms [26; 9] or genetic algorithms [27; 4]. Considering the fact that the sensitivity [28], *i.e.*, importance, of each layer is different, we propose a Hessian-Aware Stochastic Bit-switching (HASB) strategy for one-shot mixed-precision training.

Specifically, the Hessian Matrix Trace (HMT) is utilized to measure the sensitivity of each layer. We first need to compute the pre-trained model's HMT by around 1000 training images [11], as shown in



(a) Unsensitive (b) Sensitive (c) Hessian trace (d) Mixed precision

Figure 4: The HASB stochastic process and Mixed-precision of ResNet18 for {2,4,6,8}-bit.

Figure 4 (c). Then, the HMT of different layers is utilized as the probability metric for bit-switching. Higher bits are priority selected for sensitive layers, while all candidate bits are equally selected for unsensitive layers. Our proposed Roulette algorithm is used for bit-switching processes of different layers during training, as shown in the Algorithm 1. If a layer's HMT exceeds the average HMT of all layers, it is recognized as sensitive, and the probability distribution of Figure 4 (b) is used for bit selection. Conversely, if the HMT is below the average, the probability distribution of Figure 4 (a) is used for selection. Finally, the Integer Linear Programming (ILP) [29] algorithm is employed to find the optimal SubNets. Considering each layer's sensitivity during training and adding this sensitivity to the ILP's constraint factors (*e.g.*, model's FLOPs, latency, and parameters), which depend on the actual deployment requirements. We can efficiently attain a set of optimal SubNets during the search stage without retraining, thereby significant reduce the overall costs. All the searched SubNets collectively constitute the Pareto Frontier optimal solution, as shown in Figure 4 (d). For detailed mixed-precision training and searching process (*i.e.*, ILP) please refer to the Algorithm A.3 and the Algorithm 2 respectively.

Algorithm 1 Roulette algorithm for bit-switching

Require: Candidate bit-widths set $b \in B$, the HMT of current layer: t_l, average HMT: t_m;

Sample $r \sim U(0,1]$ from a uniform distribution;
if $t_l < t_m$ **then**
  Compute bit-switching probability of all candidate b_i with $p_i = 1/n$;
  Set $s = 0$, and $i = 0$;
  while $s < r$ **do**
    $i = i + 1$;
    $s = p_i + s$;
  end while
else
  Compute bit-switching probability of all candidate b_i with $p_i = b_i / \|B\|_1$;
  Set $s = 0$, and $i = 0$;
  while $s < r$ **do**
    $i = i + 1$;
    $s = p_i + s$;
  end while
end if
return b_i;

Note that n and L represent the number of candidate bit-widths and model layers respectively, and $\|\cdot\|_1$ is L_1 norm.

Algorithm 2 Our searching process for SubNets

Input: Candidate bit-widths set $b \in B$, the HMT of different layers of FP32 model: $t_l \in \{T\}_{l=1}^{L}$, the constraint average bit-width: ω, each layer parameters: $n_l \in \{N\}_{l=1}^{L}$;

Initial searched SubNets'solutions: $S = \phi$
Minimal objective : $O = \sum_{l=1}^{L} \frac{t_l}{n_l} \cdot b_l$
Constraints: $\omega \equiv \frac{\sum_{l=1}^{L} b_l}{L}$
The first solve: $\mathbf{s_1} = pulp.solve(O, \omega)$ and $S.append(\mathbf{s_1})$
for c_i in $\mathbf{s_1}$ **do**
  for b in $B[: idenx(max(\mathbf{s_1}))]$ **do**
    if $b \neq c_i$ **then**
      Add constraint: $b \equiv c_i$
      Solve: $\mathbf{s} = pulp.solve(O, \omega, b)$
      if $\mathbf{s}$ not in S **then**
        $S.append(\mathbf{s})$
      end if
      Pop last constraint: $b \equiv c_i$
    end if
  end for
end for
return S

4 Experimental Results

Setup. In this paper, we mainly focus on ImageNet-1K [30] classification task using both classical networks (ResNet18/50 [14]) and lightweight networks (MobileNetV2 [31]), which same as previous works. Experiments cover joint quantization training for multi-precision and mixed precision. We explore two candidate bit configurations, *i.e.*, {8,6,4,2}-bit and {4,3,2}-bit, each number represents the quantization level of the weight and activation layers. Like previous methods, we exclude batch normalization layers from quantization, and the first and last layers are kept at full precision. We initialize the multi-precision models with a pre-trained FP32 model, and initialize the mixed-precision models with a pre-trained multi-precision model. All models use the *Adam* optimizer [32] with a batch size of 256 for 90 epochs and use a cosine scheduler without warm-up phase. The initial learning rate is 5e-4 and weight decay is 5e-5. Data augmentation uses the standard set of transformations including random cropping, resizing to 224×224 pixels, and random flipping. Images are resized to 256×256 pixels and then center-cropped to 224×224 resolution during evaluation.

4.1 Multi-Precision

Results. For {8,6,4,2}-bit configuration, the Top-1 validation accuracy is shown in Table 1. The network weights and the corresponding activations are quantized into w-bit and a-bit respectively. Our *double-rounding* combined with ALRS training strategy surpasses the previous state-of-the-art (SOTA) methods. For example, in ResNet18, it exceeds Any-Precision [6] by 2.7%(or 2.83%) under w8a8 setting without(or with) using KD technique [15], and outperforms MultiQuant [4] by 0.63%(or 0.73%) under w4a4 setting without(or with) using KD technique respectively. Additionally, when the candidate bit-list includes 2-bit, the previous methods can't converge on MobileNetV2 during training. So, they use {8,6,4}-bit precision for MobileNetV2 experiments. For consistency, we also test {8,6,4}-bit results, as shown in the "Ours $_{\{8,6,4\}\text{-bit}}$" rows of Table 1. Our method achieves 0.25%/0.11%/0.56% higher accuracy than AdaBits [3] under the w8a8/w6a6/w4a4 settings.

Notably, our method exhibits the ability to converge but shows a big decline in accuracy on MobileNetV2. On the one hand, the compact model exhibits significant differences in the quantization scale gradients of different channels due to involving DeepWise Convolution [33]. On the other hand, when the bit-list includes 2-bit, it intensifies competition between different precisions during training. To improve the accuracy of compact models, we suggest considering the per-layer or per-channel learning rate scaling techniques in future work.

Table 1: Top1 accuracy comparisons on multi-precision of {8,6,4,2}-bit on ImageNet-1K datasets. 'KD' denotes knowledge distillation. The "−" represents the unqueried value.

Model	Method	KD	Storage	Epoch	w8a8	w6a6	w4a4	w2a2	FP
ResNet18	Hot-Swap[34]	✗	32bit	−	70.40	70.30	70.20	64.90	−
	L1[35]	✗	32bit	−	69.92	66.39	0.22	−	70.07
	KURE[36]	✗	32bit	80	70.20	70.00	66.90	−	70.30
	Ours	✗	8bit	90	70.74	70.71	70.43	66.35	69.76
	Any-Precision[6]	✓	32bit	80	68.04	−	67.96	64.19	69.27
	CoQuant[7]	✓	8bit	100	67.90	67.60	66.60	57.10	69.90
	MultiQuant[4]	✓	32bit	90	70.28	70.14	69.80	66.56	69.76
	Ours	✓	8bit	90	**70.87**	**70.79**	**70.53**	**66.84**	69.76
ResNet50	Any-Precision[6]	✗	32bit	80	74.68	−	74.43	72.88	75.95
	Hot-Swap[34]	✗	32bit	−	75.60	75.50	75.30	71.90	−
	KURE[36]	✗	32bit	80	−	76.20	74.30	−	76.30
	Ours	✗	8bit	90	76.51	76.28	75.74	72.31	76.13
	Any-Precision[6]	✓	32bit	80	74.91	−	74.75	73.24	75.95
	MultiQuant[4]	✓	32bit	90	76.94	76.85	76.46	73.76	76.13
	Ours	✓	8bit	90	**76.98**	**76.86**	**76.52**	**73.78**	76.13
MobileNetV2	AdaBits[3]	✗	8bit	150	72.30	72.30	70.30	−	71.80
	KURE[36]	✗	32bit	80	−	70.00	59.00	−	71.30
	Ours $_{\{8,6,4\}\text{-bit}}$	✗	8bit	90	72.42	72.06	69.92	−	71.14
	MultiQuant[4]	✓	32bit	90	72.33	72.09	70.59	−	71.88
	Ours $_{\{8,6,4\}\text{-bit}}$	✓	8bit	90	**72.55**	**72.41**	**70.86**	−	71.14
	Ours $_{\{8,6,4,2\}\text{-bit}}$	✗	8bit	90	70.98	70.70	68.77	50.43	71.14
	Ours $_{\{8,6,4,2\}\text{-bit}}$	✓	8bit	90	71.35	71.20	69.85	**53.06**	71.14

For {4,3,2}-bit configuration, Table 2 demonstrate that our *double-rounding* consistently surpasses previous SOTA methods. For instance, in ResNet18, it exceeds Bit-Mixer [9] by 0.63%/0.7%/1.2%(or 0.37%/0.64%/1.02%) under w4a4/w3a3/w2a2 settings without(or with) using KD technique, and outperforms ABN[10] by 0.87%/0.74%/1.12% under w4a4/w3a3/w2a2 settings with using KD technique respectively. In ResNet50, Our method outperforms Bit-Mixer [9] by 0.86%/0.63%/0.1% under w4a4/w3a3/w2a2 settings.

Notably, the overall results of Table 2 are worse than the {8,6,4,2}-bit configuration for joint training. We analyze that this discrepancy arises from information loss in the shared lower precision model (*i.e.*, 4-bit) used for bit-switching. In other words, compared with 4-bit, it is easier to directly optimize 8-bit quantization parameters to converge to the optimal value. So, we recommend including 8-bit for multi-precision training. Furthermore, independently learning the quantization scales for different precisions, including weights and activations, significantly improves accuracy compared to using shared scales. However, it requires saving the model in 32-bit format, as shown in "Ours*" of Table 2.

Table 2: Top1 accuracy comparisons on multi-precision of {4,3,2}-bit on ImageNet-1K datasets.

Model	Method	KD	Storage	Epoch	w4a4	w3a3	w2a2	FP
ResNet18	Bit-Mixer[9]	✗	4bit	160	69.10	68.50	65.10	69.60
	Vertical-layer[37]	✗	4bit	300	69.20	68.80	66.60	70.50
	Ours	✗	4bit	90	69.73	69.20	66.30	69.76
	Q-DNNs[7]	✓	32bit	45	66.94	66.28	62.91	68.60
	ABN[10]	✓	4bit	160	68.90	68.60	65.50	–
	Bit-Mixer[9]	✓	4bit	160	69.40	68.70	65.60	69.60
	Ours	✓	4bit	90	**69.77**	**69.34**	**66.62**	69.76
ResNet50	Ours	✗	4bit	90	75.81	75.24	71.62	76.13
	AdaBits[3]	✗	32bit	150	76.10	75.80	73.20	75.00
	Ours*	✗	32bit	90	**76.42**	**75.82**	**73.28**	76.13
	Bit-Mixer[9]	✓	4bit	160	75.20	74.90	72.70	–
	Ours	✓	4bit	90	76.06	75.53	72.80	76.13

4.2 Mixed-Precision

Results. We follow previous works to conduct mixed-precision experiments based on the {4,3,2}-bit configuration. Our proposed one-shot mixed-precision joint quantization method with the HASB technique comparable to the previous SOTA methods, as presented in Table 3. For example, in ResNet18, our method exceeds Bit-Mixer [9] by 0.83%/0.72%/0.77%/7.07% under w4a4/w3a3/w2a2/3MP settings and outperforms EQ-Net[5] by 0.2% under 3MP setting. The results demonstrate the effectiveness of one-shot mixed-precision joint training to consider sensitivity with Hessian Matrix Trace when randomly allocating bit-widths for different layers. Additionally, Table 3 reveals that our results do not achieve optimal performance across all settings. We hypothesize that extending the number of training epochs or combining ILP with other efficient search methods, such as genetic algorithms, may be necessary to achieve optimal results in mixed-precision optimization.

Table 3: Top1 accuracy comparisons on mixed-precision of {4,3,2}-bit on ImageNet-1K dataset. "MP" denotes average bit-width for mixed-precision. The "−" represents the unqueried value.

Model	Method	KD	Training	Searching	Fine-tune	Epoch	w4a4	w3a3	w2a2	3MP	FP
ResNet18	Ours	✗	HASB	ILP	w/o	90	69.80	68.63	64.88	68.85	69.76
	Bit-Mixer[9]	✓	Random	Greedy	w/o	160	69.20	68.60	64.40	62.90	69.60
	ABN[10]	✓	DRL	DRL	w.	160	69.80	69.00	**66.20**	67.70	–
	MultiQuant[4]	✓	LRH	Genetic	w.	90	–	67.50	–	69.20	69.76
	EQ-Net[5]	✓	LRH	Genetic	w.	120	–	69.30	65.90	69.80	69.76
	Ours	✓	KD	KD	w/o	90	**70.03**	**69.32**	65.17	**69.92**	69.76
ResNet50	Ours	✗	HASB	ILP	w/o	90	75.01	74.31	71.47	75.06	76.13
	Bit-Mixer[9]	✓	Random	Greedy	w/o	160	75.20	**74.80**	72.10	73.20	–
	EQ-Net[5]	✓	LRH	Genetic	w.	120	–	74.70	**72.50**	75.10	76.13
	Ours	✓	HASB	ILP	w/o	90	**75.63**	74.36	72.32	**75.24**	76.13

4.3 Ablation Studies

ALRS vs. Conventional in Multi-Precision. To verify the effectiveness of our proposed ALRS training strategy, we conduct an ablation experiment without KD, as shown in Table 4, and observe overall accuracy improvements, particularly for the 2bit. Like previous works, where MobileNetV2 can't achieve stable convergence with {4,3,2}-bit, we also opt for {8,6,4}-bit to keep consistent. However, our method can achieve stable convergence with {8,6,4,2}-bit quantization. This demonstrates the superiority of our proposed *Double-Rounding* and ALRS methods.

Multi-Precision vs. Separate-Precision in Time Cost. We statistic the results regarding the time cost for multi-precision compared to separate-precision quantization, as shown in Table 5. Multi-precision training costs stay approximate constant as the number of candidate bit-widths.

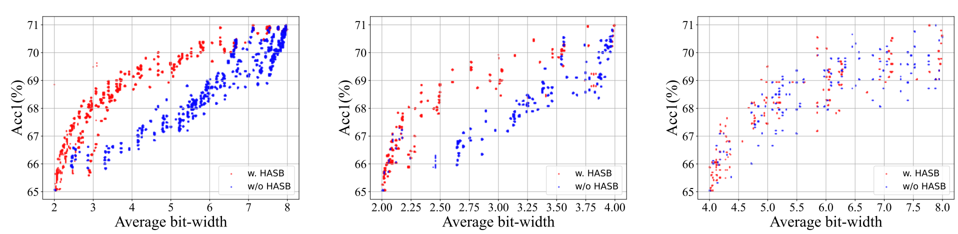
Table 4: Ablation studies of multi-precision, ResNet20 on CIFAR-10 dataset and other models on ImageNet-1K dataset. Note that MobileNetV2 uses {8,6,4}-bit instead of {4,3,2}-bit.

Model	ALRS	{8,6,4,2}-bit				{4,3,2}-bit			FP
		w8a8	w6a6	w4a4	w2a2	w4a4	w3a3	w2a2	
ResNet20	w/o	92.17	92.20	92.17	89.67	91.19	90.98	88.62	92.30
	w.	92.25	92.32	92.09	90.19	91.79	91.83	88.88	92.30
ResNet18	w/o	70.05	69.80	69.32	65.83	69.38	68.74	65.62	69.76
	w.	70.74	70.71	70.43	66.35	69.73	69.20	66.30	69.76
ResNet50	w/o	76.18	76.08	75.64	70.28	75.48	74.85	70.64	76.13
	w.	76.51	76.28	75.74	72.31	75.81	75.24	71.62	76.13
MobileNetV2	w/o	70.55	70.65	68.08	45.00	72.06	71.87	69.40	71.14
	w.	70.98	70.70	68.77	50.43	72.42	72.06	69.92	71.14

Table 5: Training costs for multi-precision and separate-precision are averaged over three runs.

Model	Dataset	Bit-widths	#V100	Epochs	BatchSize	Avg. hours	Save cost (%)
ResNet20	Cifar10	Separate-bit	1	200	128	0.9	0.0
		{4,3,2}-bit	1	200	128	0.7	28.6
		{8,6,4,2}-bit	1	200	128	0.8	12.5
ResNet18	ImageNet	Separate-bit	4	90	256	19.0	0.0
		{4,3,2}-bit	4	90	256	15.2	25.0
		{8,6,4,2}-bit	4	90	256	16.3	16.6
ResNet50	ImageNet	Separate-bit	4	90	256	51.6	0.0
		{4,3,2}-bit	4	90	256	40.7	26.8
		{8,6,4,2}-bit	4	90	256	40.8	26.5

Pareto Frontier of Different Mixed-Precision Configurations. To verify the effectiveness of our HASB strategy, we conduct ablation experiments on different bit-lists. Figure 5 shows the search results of Mixed-precision SuperNet under {8,6,4,2}-bit, {4,3,2}-bit and {8,4}-bit configurations respectively. Where each point represents a SubNet. These results are obtained directly from ILP sampling without retraining or fine-tuning. As the figure shows, the highest red points are higher than the blue points under the same bit width, indicating that this strategy is effective.



(a) {8,6,4,2}-bit (b) {4,3,2}-bit (c) {8,4}-bit

Figure 5: Comparison of HASB and Baseline approaches for Mixed-Precision on ResNet18.

5 Conclusion

This paper first introduces *Double Rounding* quantization method used to address the challenges of multi-precision and mixed-precision joint training. It can store single integer-weight parameters and attain nearly lossless bit-switching. Secondly, we propose an Adaptive Learning Rate Scaling (ALRS) method for multi-precision joint training that narrows the training convergence gap between high-precision and low-precision, enhancing model accuracy of multi-precision. Finally, our proposed Hessian-Aware Stochastic Bit-switching (HASB) strategy for one-shot mixed-precision SuperNet and efficient searching method combined with Integer Linear Programming, achieving approximate Pareto Frontier optimal solution. Our proposed methods aim to achieve a flexible and effective model compression technique for adapting different storage and computation requirements.

References

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

[2] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *arXiv preprint arXiv:1902.08153*, 2019.

[3] Q. Jin, L. Yang, and Z. Liao, "Adabits: Neural network quantization with adaptive bit-widths," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2146–2156.

[4] K. Xu, Q. Feng, X. Zhang, and D. Wang, "Multiquant: Training once for multi-bit quantization of neural networks," in *IJCAI*, L. D. Raedt, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2022, pp. 3629–3635, main Track. [Online]. Available: https://doi.org/10.24963/ijcai.2022/504

[5] K. Xu, L. Han, Y. Tian, S. Yang, and X. Zhang, "Eq-net: Elastic quantization neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 1505–1514.

[6] H. Yu, H. Li, H. Shi, T. S. Huang, and G. Hua, "Any-precision deep neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 763–10 771.

[7] K. Du, Y. Zhang, and H. Guan, "From quantized dnns to quantizable dnns," *CoRR*, vol. abs/2004.05284, 2020. [Online]. Available: https://arxiv.org/abs/2004.05284

[8] X. Sun, R. Panda, C.-F. R. Chen, N. Wang, B. Pan, A. Oliva, R. Feris, and K. Saenko, "Improved techniques for quantizing deep networks with adaptive bit-widths," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 957–967.

[9] A. Bulat and G. Tzimiropoulos, "Bit-mixer: Mixed-precision networks with runtime bit-width selection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5188–5197.

[10] C. Tang, H. Zhai, K. Ouyang, Z. Wang, Y. Zhu, and W. Zhu, "Arbitrary bit-width network: A joint layer-wise quantization and adaptive inference approach," 2022. [Online]. Available: https://arxiv.org/abs/2204.09992

[11] Z. Dong, Z. Yao, D. Arfeen, A. Gholami, M. W. Mahoney, and K. Keutzer, "Hawq-v2: Hessian aware trace-weighted quantization of neural networks," *Advances in neural information processing systems*, vol. 33, pp. 18 518–18 529, 2020.

[12] Y. Dong, R. Ni, J. Li, Y. Chen, H. Su, and J. Zhu, "Stochastic quantization for learning accurate low-bit deep neural networks," *International Journal of Computer Vision*, vol. 127, pp. 1629–1642, 2019.

[13] Q. Jin, L. Yang, and Z. Liao, "Towards efficient training for neural network quantization," *arXiv preprint arXiv:1912.10207*, 2019.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[15] K. Kim, B. Ji, D. Yoon, and S. Hwang, "Self-knowledge distillation with progressive refinement of targets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6567–6576.

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

[17] A. Elthakeb, P. Pilligundla, F. Mireshghallah, A. Yazdanbakhsh, S. Gao, and H. Esmaeilzadeh, "Releq: an automatic reinforcement learning approach for deep quantization of neural networks," in *NeurIPS ML for Systems workshop, 2018*, 2019.

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

[19] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVI 16*. Springer, 2020, pp. 544–560.

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

[21] J. Liu, J. Cai, and B. Zhuang, “Sharpness-aware quantization for deep neural networks,” *arXiv preprint arXiv:2111.12273*, 2021.

[22] Z. Yao, Z. Dong, Z. Zheng, A. Gholami, J. Yu, E. Tan, L. Wang, Q. Huang, Y. Wang, M. Mahoney *et al.*, “Hawq-v3: Dyadic neural network quantization,” in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 875–11 886.

[23] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” *CoRR*, vol. abs/1712.05877, 2017. [Online]. Available: http://arxiv.org/abs/1712.05877

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

[25] Y. You, I. Gitman, and B. Ginsburg, “Large batch training of convolutional networks,” *arXiv preprint arXiv:1708.03888*, 2017.

[26] Z. Cai and N. Vasconcelos, “Rethinking differentiable search for mixed-precision neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2349–2358.

[27] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, “Single path one-shot neural architecture search with uniform sampling,” in *European conference on computer vision*. Springer, 2020, pp. 544–560.

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

[29] Y. Ma, T. Jin, X. Zheng, Y. Wang, H. Li, Y. Wu, G. Jiang, W. Zhang, and R. Ji, “Ompq: Orthogonal mixed precision quantization,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 7, 2023, pp. 9029–9037.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[31] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[32] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[33] T. Sheng, C. Feng, S. Zhuo, X. Zhang, L. Shen, and M. Aleksic, “A quantization-friendly separable convolution for mobilenets,” in *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*. IEEE, 2018, pp. 14–18.

[34] Q. Sun, X. Li, Y. Ren, Z. Huang, X. Liu, L. Jiao, and F. Liu, “One model for all quantization: A quantized network supporting hot-swap bit-width adjustment,” *arXiv preprint arXiv:2105.01353*, 2021.

[35] M. Alizadeh, A. Behboodi, M. van Baalen, C. Louizos, T. Blankevoort, and M. Welling, “Gradient l1 regularization for quantization robustness,” *arXiv preprint arXiv:2002.07520*, 2020.

[36] B. Chmiel, R. Banner, G. Shomron, Y. Nahshan, A. Bronstein, U. Weiser *et al.*, “Robust quantization: One model to rule them all,” *Advances in neural information processing systems*, vol. 33, pp. 5308–5317, 2020.

[37] H. Wu, R. He, H. Tan, X. Qi, and K. Huang, “Vertical layering of quantized neural networks for heterogeneous inference,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 15 964–15 978, 2023.

[38] Y. Bhalgat, J. Lee, M. Nagel, T. Blankevoort, and N. Kwak, “Lsq+: Improving low-bit quantization through learnable offsets and better initialization,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 696–697.

[39] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, “Slimmable neural networks,” *arXiv preprint arXiv:1812.08928*, 2018.

A Appendix / supplemental material

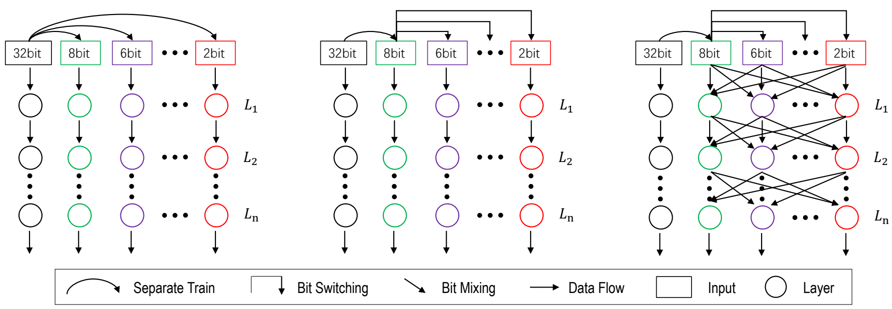
A.1 Overview

In this supplementary material, we present more explanations and experimental results.

- First, we provide a detailed explanation of the different quantization types under QAT.
- We then present a comparison of multi-precision and separate-precision on the ImageNet-1k dataset.
- Furthermore, we provide the gradient formulation of Double Rounding.
- And, the algorithm implementation of both multi-precision and mixed-precision training approaches.
- Finally, we provide more gradient statistics of learnable quantization scales in different networks.

A.2 Different Quantization Types

In this section, we provide a detailed explanation of the different quantization types during Quantization-Aware Training (QAT), as is shown in Figure 6.



(a) **Separate-Precision**: Each bit-width requires training a new network with separate weights by repeating multi-retrain.

(b) **Multi-Precision**: A shared network can be quantized to any bit-width at runtime without re-training or finetuning. All layers inside the network uniformly share the same bit-width.

(c) **Mixed-Precision**: A SuperNet whose individual layers can be quantized to any bit-width at runtime, and its searched subnets without re-training or fine-tuning.

Figure 6: Comparison between different quantization types during quantization-aware training.

A.3 Multi-Precision vs. Separate-Precision.

We provide the comparison of Multi-Precision and Separate-Precision on ImageNet-1K dataset. Table 6 shows that our Multi-Precision joint training scheme has comparable accuracy of different precisions compared to Separate-Precision with multiple re-train. This further proves the effectiveness of our proposed One-shot *Double Rounding* Multi-Precision method.

Table 6: Top1 accuracy comparisons on multi-precision of {8,6,4,2}-bit on ImageNet-1K datasets.

Model	Method	One-shot	Storage	Epoch	w8a8	w6a6	w4a4	w2a2	FP
ResNet18	LSQ[2]	✗	{8,6,4,2}-bit	90	**71.10**	–	**71.10**	**67.60**	70.50
	LSQ+[38]	✗	{8,6,4,2}-bit	90	–	–	70.80	66.80	70.10
	Ours	✓	8-bit	90	70.74	70.71	70.43	66.35	69.76
ResNet50	LSQ[2]	✗	{8,6,4,2}-bit	90	**76.80**	–	**76.70**	**73.70**	76.90
	Ours	✓	8-bit	90	76.51	76.28	75.74	72.31	76.13

A.4 The Gradient Formulation of Double Rounding

A general formulation for uniform quantization process is as follows:

$$\widetilde{W} = \text{clip}(\left\lfloor \frac{W}{\mathbf{s}} \right\rceil + \mathbf{z}, -2^{b-1}, 2^{b-1} - 1) \tag{8}$$

$$\widehat{W} = (\widetilde{W} - \mathbf{z}) \times \mathbf{s} \tag{9}$$

where the symbol $\lfloor . \rceil$ denotes the $Rounding$ function, clip$(x, low, upper)$ expresses x below low are set to low and above $upper$ are set to $upper$. b denotes the quantization level (or bit-width), $\mathbf{s} \in \mathbb{R}$ and $\mathbf{z} \in \mathbb{Z}$ represents the quantization *scale* (or interval) and *zero-point* associated with each b, respectively. W represents the FP32 model's weights, $\widetilde{W}$ signifies the quantized integer weights, and $\widehat{W}$ represents the dequantized floating-point weights.

The quantization scale of our *Double Rounding* is learned online and not fixed. And it only needs a pair of shared quantization parameters, *i.e.*, *scale* and *zero-point*. Suppose the highest-bit and the low-bit are denoted as h-bit and l-bit respectively, and the difference between them is $\Delta = h - l$. The specific formulation is as follows:

$$\widetilde{W}_h = \text{clip}(\left\lfloor \frac{W - \mathbf{z}_h}{\mathbf{s}_h} \right\rceil, -2^{h-1}, 2^{h-1} - 1) \tag{10}$$

$$\widetilde{W}_l = \text{clip}(\left\lfloor \frac{\widetilde{W}_h}{2^{\Delta}} \right\rceil, -2^{l-1}, 2^{l-1} - 1) \tag{11}$$

$$\widehat{W}_l = \widetilde{W}_l \times \mathbf{s}_h \times 2^{\Delta} + \mathbf{z}_h \tag{12}$$

where $\mathbf{s}_h \in \mathbb{R}$ and $\mathbf{z}_h \in \mathbb{Z}$ denote the highest-bit quantization *scale* and *zero-point* respectively. $\widetilde{W}_h$ and $\widetilde{W}_l$ represent the quantized weights of the highest-bit and low-bit respectively. Hardware shift operations can efficiently execute the division and multiplication by 2^{Δ}. And the $\mathbf{z}_h$ is 0 for the weight quantization in this paper. The gradient formulation of *Double Rounding* for one-shot joint training is represented as follows:

$$\frac{\partial \widehat{Y}}{\partial \mathbf{s}_h} \simeq \begin{cases} \left\lfloor \frac{Y - \mathbf{z}_h}{\mathbf{s}_h} \right\rceil - \frac{Y - \mathbf{z}_h}{\mathbf{s}_h} & if\, n < \frac{Y - \mathbf{z}_h}{\mathbf{s}_h} < p, \\ n \quad or \quad p & otherwise. \end{cases} \tag{13}$$

$$\frac{\partial \widehat{Y}}{\partial \mathbf{z}_h} \simeq \begin{cases} 0 & if\, n < \frac{Y - \mathbf{z}_h}{\mathbf{s}_h} < p, \\ 1 & otherwise. \end{cases} \tag{14}$$

where n and p denote the lower and upper bounds of the integer range $[N_{min}, N_{max}]$ for quantizing the weights or activations respectively. Y represents the FP32 weights or activations, and $\widehat{Y}$ represents the dequantized weights or activations. Unlike weights, activation quantization *scale* and *zero-point* of different precisions are learned independently. However, the gradient formulation is the same.

A.5 Algorithms

This section provides the algorithm implementations of multi-precision, one-shot mixed-precision joint training, and the search stage of SubNets.

A.5.1 Multi-Precision Joint Training

The multi-precision model with different quantization precisions shares the same model weight(*e.g.*, the highest-bit) during joint training. In conventional multi-precision, the shared weight (*e.g.*, multi-precision model) computes n forward processes at each training iteration, where n is the number of candidate bit-widths. Then, all attained losses of different precisions perform an accumulation, and update the parameters accordingly. For specific implementation details please refer to Algorithm A.1.

However, we find that if separate precision loss and parameter updates are performed directly after calculating a precision at each forward process, it will lead to difficulty convergence during training or suboptimal accuracy. In other words, the varying gradient magnitudes of quantization scales of different precisions make it hard to attain stable convergence during joint training. To address this issue, we introduce an adaptive approach (*e.g.*, Adaptive Learning Rate Scaling, ALRS) to alter the learning rate for different precisions during training, aiming to achieve a consistent update pace. This method allows us to directly update the shared parameters after calculating the loss after every forward. We update both the weight parameters and quantization parameters simultaneously using dual optimizers. We also set the weight-decay of the quantization scales to 0 to achieve more stable convergence. For specific implementation details, please refer to Algorithm A.2.

Algorithm A.1 Conventional Multi-precision training approach

Require: Candidate bit-widths set $b \in B$;

Initialize: Pretrained model M with FP32 weights W, the quantization scales $\mathbf{s}$ including of weights $\mathbf{s}_w$ and activations $\mathbf{s}_x$, BatchNorm layers:$\{BN\}_{b=1}^{n}$, optimizer:$optim(W, \mathbf{s}, wd)$, learning rate: λ, wd: weight decay, CE: CrossEntropyLoss, D_{train}: training dataset;
For one epoch:
Sample mini-batch data $(\mathbf{x}, \mathbf{y}) \in \{D_{train}\}$
for b in B **do**
    $forward(M, \mathbf{x}, \mathbf{y}, b)$:
    for each quantization layer **do**
        $\widehat{W}^b = dequant(quant(W, \mathbf{s}_w^b))$
        $\widehat{X}^b = dequant(quant(X, \mathbf{s}_x^b))$
        $O^b = Conv(\widehat{W}^b, \widehat{X}^b)$
    end for
    $\mathbf{o}^b = FC(W, O^b)$
    Update BN^b layer
    Compute loss: $\mathcal{L}^b = CE(\mathbf{o}^b, \mathbf{y})$
    Compute gradients: $\mathcal{L}^b.backward()$
end for
Update weights and scales: $optim.step(\lambda)$
Clear gradient: $optim.zero\_grad()$;

Note that n and L represent the number of candidate bit-widths and model layers respectively.

Algorithm A.2 Our Multi-precision training approach

Require: Candidate bit-widths set $b \in B$

Initialize: Pretrained model M with FP32 weights W, the quantization scales $\mathbf{s}$ including of weights $\mathbf{s}_w$ and activations $\mathbf{s}_x$, BatchNorm layers: $\{BN\}_{b=1}^{n}$, optimizers: $optim_1(W, wd)$, $optim_2(\mathbf{s}, wd = 0)$, learning rate: λ, wd: weight decay, CE: CrossEntropyLoss, D_{train}: training dataset;
For every epoch:
Sample mini-batch data $(\mathbf{x}, \mathbf{y}) \in \{D_{train}\}$
for b in B **do**
    $forward(M, \mathbf{x}, \mathbf{y}, b)$:
    for each quantization layer **do**
        $\widehat{W}^b = dequant(quant(W, \mathbf{s}_w^b))$
        $\widehat{X}^b = dequant(quant(X, \mathbf{s}_x^b))$
        $O^b = Conv(\widehat{W}^b, \widehat{X}^b)$
    end for
    $\mathbf{o}^b = FC(W, O^b)$
    Update BN^b layer
    Compute loss: $\mathcal{L}^b = CE(\mathbf{o}^b, \mathbf{y})$
    Compute gradients: $\mathcal{L}^b.backward()$
    Compute learning rate: λ_b # please see formula (6) of the main paper
    Update weights and quantization scales: $optim_1.step(\lambda)$; $optim_2.step(\lambda_b)$
    Clear gradient: $optim_1.zero\_grad()$; $optim_2.zero\_grad()$
end for

Note that n and L represent the number of candidate bit-widths and model layers respectively.

A.5.2 One-shot Joint Training for Mixed Precision SuperNet

Unlike multi-precision joint quantization, the bit-switching of mixed-precision training is more complicated. In multi-precision training, the bit-widths calculated in each iteration are fixed, *e.g.*, {8,6,4,2}-bit. In mixed-precision training, the bit-widths of different layers are not fixed in each iteration, *e.g.*, {8,random-bit,2}-bit, where "random-bit" is any bits of *e.g.*, {7,6,5,4,3,2}-bit, similar to the *sandwich* strategy of [39]. Therefore, mixed precision training often requires more training epochs to reach convergence compared to multi-precision training. Bit-mixer [9] conducts the same probability of selecting bit-width for different layers. However, we take the sensitivity of each layer into consideration which uses sensitivity (*e.g.* Hessian Matrix Trace [11]) as a metric to identify the selection probability of different layers. For more sensitive layers, preference is given to higher-bit widths, and vice versa. We refer to this training strategy as a Hessian-Aware Stochastic Bit-switching

(HASB) strategy for optimizing one-shot mixed-precision SuperNet. Specific implementation details can be found in Algorithm A.3. In additionally, unlike multi-precision joint training, the BN layers are replaced by TBN (Transitional Batch-Norm) [9], which compensates for the distribution shift between adjacent layers that are quantized to different bit-widths. To achieve the best convergence effect, we propose that the threshold of bit-switching (*i.e.*, σ) also increases as the epoch increases.

Algorithm A.3 Our one-shot Mixed-precision SuperNet training approach

Require: Candidate bit-widths set $b \in B$, the HMT of different layers of FP32 model: $t_l \in \{T\}_{l=1}^{L}$, average HMT: $t_m = \frac{\sum_{l=1}^{L} t_l}{L}$;

Initialize: Pretrained model M with FP32 weights W, the quantization scales $\mathbf{s}$ including of weights $\mathbf{s}_w$ and activations $\mathbf{s}_x$, BatchNorm layers:$\{BN\}_{b=1}^{n^2}$, the threshold of bit-switching:σ, optimizer:$optim(W, \mathbf{s}, wd)$, learning rate: λ, wd: weight decay, CE: CrossEntropyLoss, D_{train}: training dataset;
For one epoch:
Attain the threshold of bit-switching: $\sigma = \sigma \times \frac{epoch+1}{total\_epochs}$
Sample mini-batch data $(\mathbf{x}, \mathbf{y}) \in \{D_{train}\}$
for b in B **do**
    $forward(M, \mathbf{x}, \mathbf{y}, b, T, t_m)$:
    for each quantization layer **do**
        Sample $r \sim U[0, 1]$;
        if $r < \sigma$ **then**
            $b = Roulette(B, t_l, t_m)$ # Please refer to Algorithm 1 of the main paper
        end if
        $\widehat{W}^b = dequant(quant(W, \mathbf{s}_w^b))$
        $\widehat{X}^b = dequant(quant(X, \mathbf{s}_x^b))$
        $O^b = Conv(\widehat{W}^b, \widehat{X}^b)$
    end for
    $\mathbf{o}^b = FC(W, O^b)$
    Update BN^b layer
    Compute loss: $\mathcal{L}^b = CE(\mathbf{o}^b, \mathbf{y})$
    Compute gradients: $\mathcal{L}^b.backward()$
    Update weights and scales: $optim.step(\lambda)$
    Clear gradient: $optim.zero\_grad()$;
end for

Note that n and L represent the number of candidate bit-widths and model layers respectively.

A.5.3 Efficient one-shot searching for Mixed Precision SuperNet

After training the mixed-precision SuperNet, the next step is to select the appropriate optimal SubNets based on conditions, such as model parameters, latency, and FLOPs, for actual deployment and inference. To achieve optimal allocations for candidate bit-width under given conditions, we employ the Iterative Integer Linear Programming (ILP) approach. Since each ILP run can only provide one solution, we obtain multiple solutions by altering the values of different average bit widths. Specifically, given a trained SuperNet (*e.g.*, RestNet18), it takes less than two minutes to solve candidate SubNets. It can be implemented through the Python `PULP` package. Finally, these searched SubNets only need inference to attain final accuracy, which needs a few hours. This forms a Pareto optimal frontier. From this frontier, we can select the appropriate subnet for deployment. Specific implementation details of the searching process by ILP can be found in Algorithm 2.

A.6 The Gradient Statistics of Learnable Scale of Quantization

In this section, we analyze the changes in gradients of the learnable scale for different models during the training process. Figure 7 and Figure 8 display the gradient statistical results for ResNet20 on CIFAR-10. Similarly, Figure 9 and Figure 10 show the gradient statistical results for ResNet18 on ImageNet-1K, and Figure 11 and Figure 12 present the gradient statistical results for ResNet50 on ImageNet-1K. These figures reveal a similarity in the range of gradient changes between higher-bit quantization and 2-bit quantization. Notably, they illustrate that the value range of 2-bit quantization is noticeably an order of magnitude higher than the value ranges of higher-bit quantization.

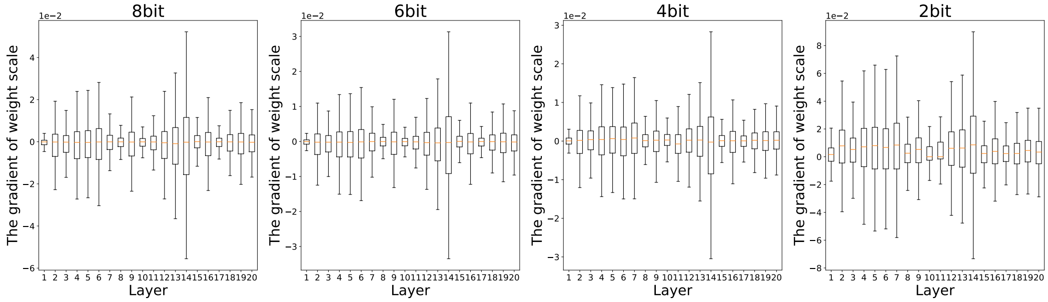


Figure 7: The scale gradient statistics of weight of ResNet20 on CIFAR-10 dataset. Note that the outliers are removed for exhibition.

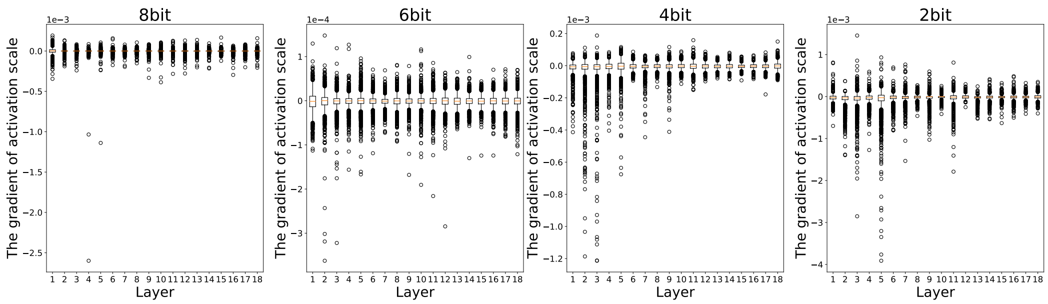


Figure 8: The scale gradient statistics of activation of ResNet20 on CIFAR-10 dataset. Note that the first and last layers are not quantized.

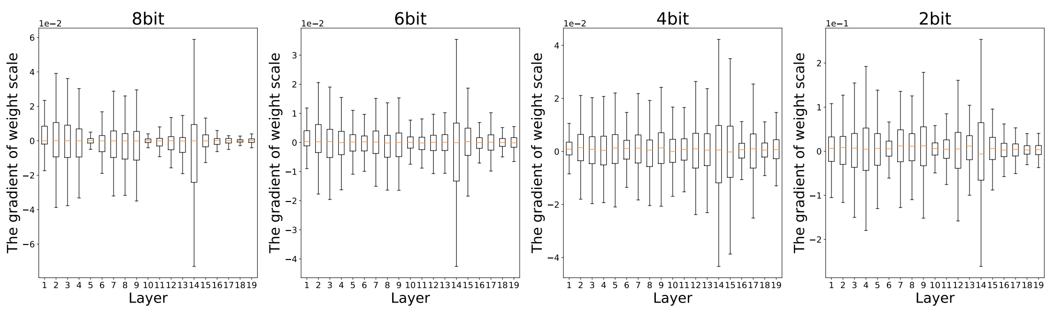


Figure 9: The scale gradient statistics of weight of ResNet18 on ImageNet dataset. Note that the outliers are removed for exhibition.

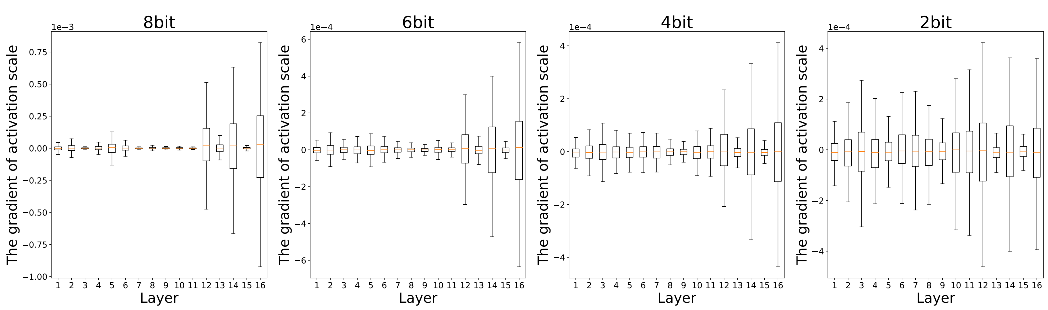


Figure 10: The scale gradient statistics of activation of ResNet18 on ImageNet dataset. Note that the outliers are removed for exhibition.

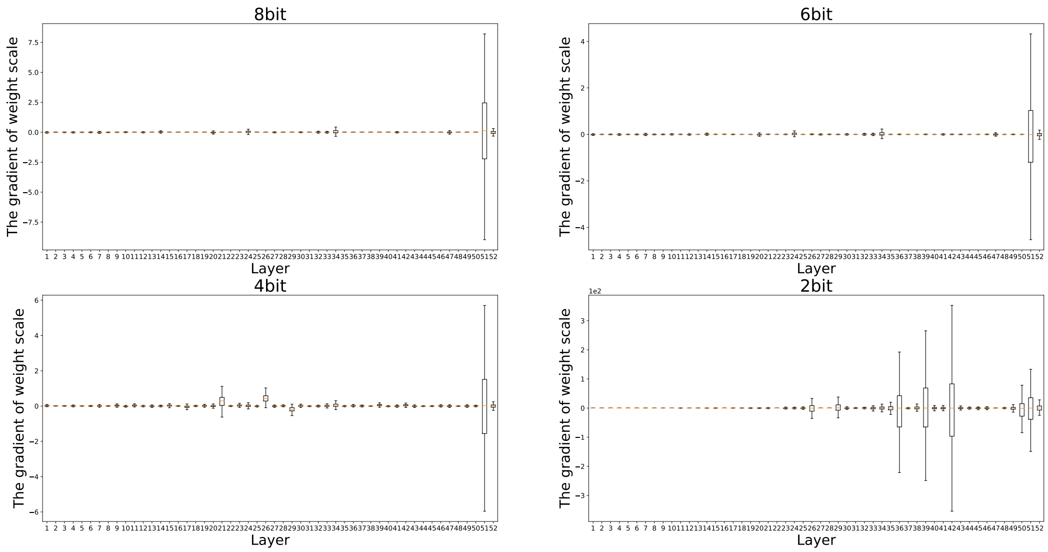


Figure 11: The scale gradient statistics of weight of ResNet50 on ImageNet dataset. Note that the outliers are removed for exhibition, and the first and last layers are not quantized.

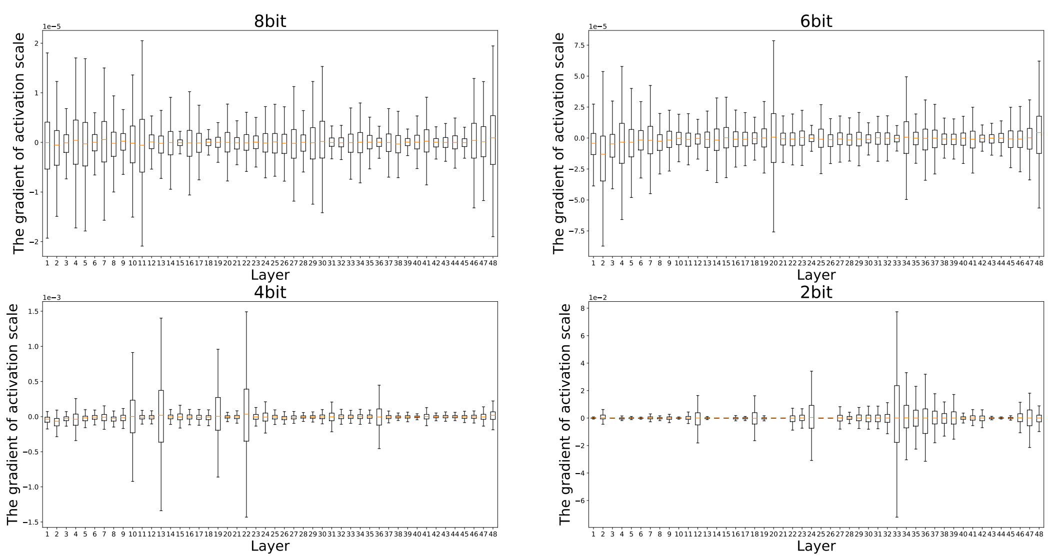


Figure 12: The scale gradient statistics of activation of ResNet50 on ImageNet dataset. Note that the outliers are removed for exhibition.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: **[TODO]** Please refer to the Abstract Section and Section 1, where related material for the question can be found.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: **[TODO]**[Yes]

   Justification: **[TODO]** Although our proposed methods have achieved comparable results in multi-precision and mixed-precision, this paper has several limitations and improvements. (1) Due to time and computing resource constraints, our methods are only tested on common CNNs-based networks and aren't tested on ViTs-based networks. (2) For multi-precision, compact networks, *e.g.*, MobileNet, still have a big drop in 2bit. We will try to use per-layer or per-channel adaptive learning rate adjustment in the future. (3) For mixed precision, relying only on one-shot ILP-based SubNets search may yield a suboptimal solution. We further need to combine it with other efficient search methods, *e.g.*, genetic algorithms, to achieve global optimal.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: **[TODO]** We display index numbers wherever formulas and theoretical support are needed. For example, please refer to Section 3.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes] To ensure that our experimental results can be reproduced: (1) we describe the experimental training settings and algorithm pseudocode in detail in Section 4 and Section A.5, and (2) we also provide the code related to all experiments in this paper, allowing the community to improve and conduct further research.

   Justification: **[TODO]**

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: **[TODO]** Our code are available at here and the data is open source dataset.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: **[TODO]** Our codes are available here and include all related training and test details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: **[TODO]** Please refer to the Section A.6 in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: **[TODO]** Please refer to the Table 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: **[TODO]** We have read the NeurIPS Code of Ethics and conform to it.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: [TODO] Due to space limitations, this social impact aspect is not discussed in the main paper. This paper doesn't involve negative societal impacts including potential malicious or unintended uses. Our proposed methods aim to achieve an efficient and effective model compression technique to flexible adaptive different storage and computation requirements, which are beneficial to social advancement.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: [TODO] This paper doesn't have any high risk for misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: [TODO] We conform to the CC-BY 4.0 license.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: **[TODO]** This paper does not release new assets

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: **[TODO]** This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: **[TODO]** This paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.