MetaMix: Meta-State Precision Searcher for Mixed-Precision Activation Quantization

Han-Byul Kim[1,2*], **Joo Hyung Lee**[2], **Sungjoo Yoo**[1], **Hong-Seok Kim**[2]

[1]Department of Computer Science and Engineering, Seoul National University, Seoul, Korea
[2]Google, Mountain View, California, USA
shinestarhb@gmail.com, rinolee@google.com, sungjoo.yoo@gmail.com, hongseok@google.com

Abstract

Mixed-precision quantization of efficient networks often suffer from activation instability encountered in the exploration of bit selections. To address this problem, we propose a novel method called MetaMix which consists of bit selection and weight training phases. The bit selection phase iterates two steps, (1) the mixed-precision-aware weight update, and (2) the bit-search training with the fixed mixed-precision-aware weights, both of which combined reduce activation instability in mixed-precision quantization and contribute to fast and high-quality bit selection. The weight training phase exploits the weights and step sizes trained in the bit selection phase and fine-tunes them thereby offering fast training. Our experiments with efficient and hard-to-quantize networks, i.e., MobileNet v2 and v3, and ResNet-18 on ImageNet show that our proposed method pushes the boundary of mixed-precision quantization, in terms of accuracy vs. operations, by outperforming both mixed- and single-precision SOTA methods.

Introduction

The ever increasing demand of efficiency requires the quantization of efficient models, e.g., MobileNet-v1 (Howard et al. 2017), v2 (Sandler et al. 2018) and v3 (Howard et al. 2019), in lower precision. Mixed-precision quantization looks promising due to the supports available on existing computing systems, e.g., 8-bit, 4-bit and 1-bit integer (NVIDIA 2018; Sharma et al. 2018).

In this work, we address per-layer mixed-precision quantization for activation while utilizing a single bit-width of weight. Mixed-precision quantization of efficient models, however, is challenging in that the design space of bit-width selection (in short, bit selection) is prohibitively large. Especially, each candidate of bit selection needs to be trained, e.g., on ImageNet dataset, for evaluation, which incurs prohibitively high training cost.

In order to address the training cost problem, we often explore bit selections while training the network weights (Habi, Jennings, and Netzer 2020; Uhlich et al. 2020; Wang, Lu, and Blankevoort 2020; Wang et al. 2019; Yang and Jin 2021; Huang et al. 2022; Shin et al. 2023; Wu et al. 2018; Cai and Vasconcelos 2020; Yu et al. 2020). In such a combination of bit selection and weight training, the bit-width of each layer can be changed (due to bit selection) across training iterations. Such a bit-width change during model training incurs a new problem called *activation instability due to bit selection*.

Figure 1 illustrates the input activation distribution before quantizer in the MobileNet-v2 model across different bit-widths of weight and activation. The figure shows, for a given activation bit-width, the activation distribution remains consistent across different weight bit-widths, e.g., the distribution of 8-bit activation across FP (full-precision), 8-bit and 4-bit weights in the first columns. However, given a bit-width (e.g., 8-bit) of weight, the distribution of the activation significantly varies across different bit-widths (e.g., 8-bit, 4-bit and 3-bit) of activations, which demonstrates activation instability due to bit selection.

Activation instability, due to weight quantization, has been reported in previous works (Park and Yoo 2020; Li et al. 2019). The problem is encountered when single-precision models are quantized in low bits. In this paper, we report a new activation instability problem encountered in the exploration of bit selections under model training. In order to address these activation instabilities, we propose a new training method called *MetaMix*. Our contribution is summarized as follows.

- We demonstrate a new problem called *activation instability due to bit selection* which disrupts exploring bit selection while training the network thereby offering suboptimal results.
- Our proposed MetaMix mitigates activation instability problem. MetaMix consists of two phases: bit selection and weight training. The bit selection phase offers fast and high-quality bit selection by both bit-meta training step and bit-search training step.
- In the bit-meta training step, we train the network weights at multiple bit-widths to obtain a state called *meta-state* which provides consistent activation distribution across different activation bit-widths. Then, in the bit-search training step, on the fixed meta-state, we learn architectural parameters for per-layer bit-width probabilities. We iterate both steps, which proves effective for fast and high-quality bit selection while mitigating the effect of activation instability.

*Work done during internship at Google

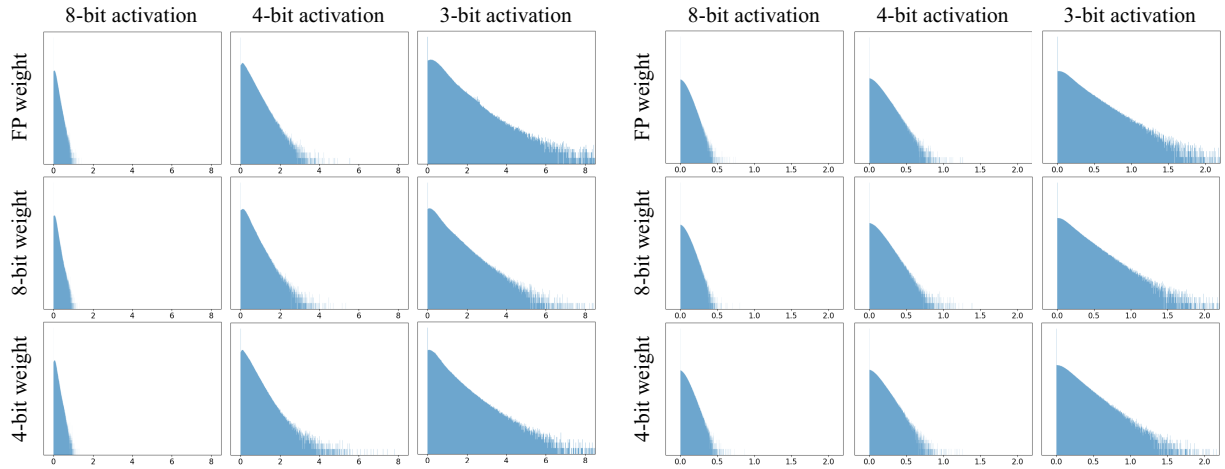


Figure 1: Input activation distribution before quantizer with 8-bit, 4-bit, and 3-bit single-precision activation quantization of MobileNet-v2 (a) in 5th layer (depth-wise convolution in 2nd block) and (b) in 32nd layer (depth-wise convolution in 11th block). Each row has the same fixed weight bit-width and each column has the same fixed activation bit-width. 'FP' represents full-precision. In all the figures, x-axis is for values and y-axis is for frequency in log scale.

- In the weight training phase, we continue to train, i.e., fine-tune network weights and step sizes[1] under the fixed per-layer bit-widths previously determined in the bit selection phase, which offers fast training due to the mixed-precision-aware initialization of weights and step sizes done in the previous phase.
- We evaluate our method on highly optimized and hard-to-quantize networks, i.e., MobileNet-v2 and v3 and ResNet-18 on ImageNet-1K dataset (Deng et al. 2009) and show that ours offers state-of-the-art results.

Related Works

Uniform quantization (Zhou et al. 2016; Choi et al. 2018; Esser et al. 2020) is hardware friendly since most of compute devices, e.g., GPU, support uniform grids (or levels). Non-uniform quantization (Zhang et al. 2019; Li, Dong, and Wang 2020; Yamamoto 2021; Liu et al. 2022) optimizes quantization grids in order to fit diverse distributions thereby enabling lower bit-widths while requiring specialized compute devices. In our work, we mainly target uniform method.

Trainable quantization: Training quantization parameters (Choi et al. 2018; Jung et al. 2018; Esser et al. 2020) like clip range or step size has a potential of lower precision and thus its possibilities have been actively investigated. Most of these works show good results in relatively redundant networks, e.g. ResNets, but fail to quantize highly optimized networks, e.g. MobileNet-v2 and v3, in 4-bit without accuracy loss. Recently, PROFIT (Park and Yoo 2020) offers 4-bit quantization in MobileNets with a special training recipe to address activation instability due to weight quantization. BASQ (Kim, Park, and Yoo 2022) enables low precision for MobileNets via quantization hyper-parameter search.

[1]The size of quantization range is determined by $(2^b - 1) * s$ in case of b-bits and a step size of s.

Mixed-precision quantization: Existing mixed-precision methods can be classified into four categories. ① Learning-based solutions (Uhlich et al. 2020; Wang, Lu, and Blankevoort 2020; Yang and Jin 2021; Zhang et al. 2021) optimize bit-widths as learnable parameters trained with gradient from task loss. SDQ (Huang et al. 2022) adopts differentiable parameter for bit-width probability. NIPQ (Shin et al. 2023) extends DiffQ (Défossez, Adi, and Synnaeve 2022) to mixed precision. ② RL-based solutions (Wang et al. 2019; Elthakeb et al. 2018) train an RL agent which learns bit-width assignment policy. ③ NAS-based solutions (Wu et al. 2018; Guo et al. 2020; Yu et al. 2020; Cai and Vasconcelos 2020) explore the design space of bit selection and attempt to solve selection problems via evolutionary search, differentiation, etc. (Real et al. 2019; Liu, Simonyan, and Yang 2019; Cai, Zhu, and Han 2019; Hu et al. 2020). ④ Metric-based solutions determine bit-width based on statistics, i.e., Hessian spectrum (Dong et al. 2019, 2020; Yao et al. 2021).

In this paper, we present a novel NAS-based mixed-precision method. Ours tries to overcome the limitations of existing NAS-based solutions, e.g., high search cost and sub-optimal results due to activation instability.

Activation Instability on Mixed-Precision Quantization

As Figure 1 illustrates, different activation bit-widths can yield different distributions even after normalization. Figure 2 exemplifies how the activation distributions vary in the process of bit selection under weight training. We change the activation bit-width of one layer of MobileNet-v2 every epoch of training by randomly picking the bit-width within 8-bit, 4-bit, and 3-bit. We experiment two cases of weight bit-width: full-precision (top of Figure 2) and 4-bit (middle).

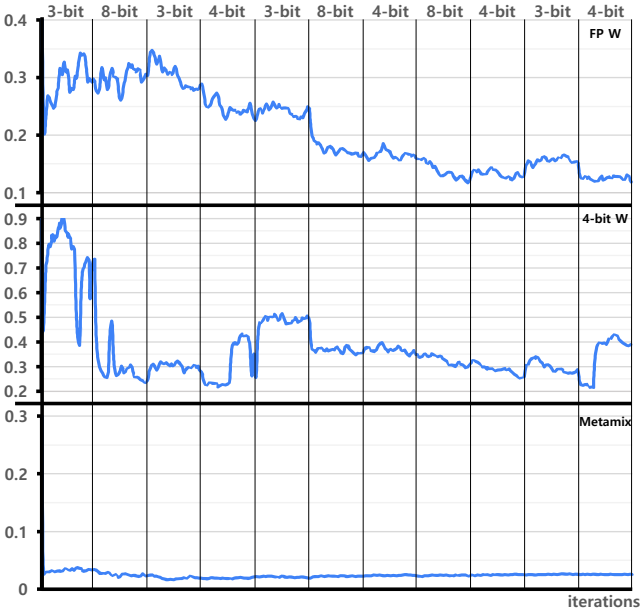


Figure 2: Trend of batch norm statistics over iterations when changing activation bit-width. We plot the running variance of batch norm which follows 5th (depth-wise) convolution layer in 2nd block (Top: FP weights, Middle: 4-bit weights, Bottom: applying MetaMix with FP weights).

The figure shows the running variance of batch normalization layer which follows the quantized depth-wise convolution layer inside of inverted residual block.

Figure 2 shows two trends in terms of activation instability. First, activation statistics has strong correlation with bit-width. Second, the correlation can be amplified when both activation and weight are trained and quantized in low bits. As the figure demonstrates, in mixed-precision quantization, activation instability results from two sources, one from bit selection and the other from weight quantization.

Activation instability due to weight quantization results from the fact that the output activations of a layer can become different, even when the same input activations are used, due to weight update and subsequent weight quantization (Park and Yoo 2020). Activation instability due to bit selection results from the fact that the lower precision tends to incur the more variance in the quantized data. For details about the instability, refer to the supplementary[2].

As will be shown in our experiments, the activation instability can yield poor quality of mixed-precision network. It is because, when the statistics of batch norm layer significantly vary due to activation instability during training, it is challenging to obtain a representative activation statistics[3] on the candidate bit-widths (during bit selection) as well as the final bit-widths (selected as a result of bit selection). In (Park and Yoo 2020; Li et al. 2019), in order to cope with activation instability due to weight quantization, the authors propose fine-tuning the quantized network (with a single precision) to stabilize batch norm statistics at the end of training. However, according to our experiments (also to be mentioned later), it does not prove effective in mixed-precision quantization. It is mainly because sensitive layers like depth-wise convolution tend to be assigned high precision, which makes activation instability due to weight quantization less significant on those layers thereby reducing the effects of fine-tuning. In this paper, we propose MetaMix to tackle the activation instability in mixed-precision quantization. MetaMix effectively stabilizes batch norm statistics (bottom of Figure 2) thereby providing representative activation statistics (Figure 6) and high-quality bit selection (Figure 7).

MetaMix – a Meta-State Precision Searcher

Overall Training Flow

Figure 3 shows the overall training flow. Given a trained network, our proposed MetaMix determines per-layer bit-width of activations in the bit selection phase and fine-tunes network weights and step sizes in the weight training phase.

Algorithm 1 shows the overall process of bit selection phase. The bit selection phase iteratively executes two steps: *bit-meta training* and *bit-search training*. The bit-meta training step trains network weights in a mixed-precision-aware manner. The bit-search training step learns the architectural parameters for per-layer bit-width probabilities on the fixed mixed-precision-aware weights learned in the bit-meta training step. In the weight training phase, using the per-layer bit-widths determined in the bit selection phase, we fine-tune the weights and step sizes for both weights and activations.

Figure 3 also shows that, given a network, we augment each layer with multiple branches, each for a specific bit-width, for the purpose of bit selection.[4] Figure 4 illustrates the details of multi-branch block. We augment the quantizer into B branches where B is the number of available bit-widths. In the figure, we assume three bit-widths and thus obtain three branches for 8-bit, 4-bit and 3-bit. Each branch is also associated with an architectural parameter for the bit-width probability. Unlike the multi-branch quantizers, we do not duplicate but share weights across the branches.

Bit-Meta Training

Bit-meta training step learns network weights to be used in the subsequent bit-search training. In order for bit-search to be successful, the network weights need to be learned in a mixed-precision-aware manner in order to facilitate the training of architectural parameters for per-layer bit-width probabilities in the bit-search training step. Since our problem of learning network weights in the bit-meta training step is similar to that of meta-learning, e.g., MAML (Finn, Abbeel, and Levine 2017), which tries to facilitate fine-tuning for the given target task (analogous to the training of architectural parameters for bit-width probabilities in the

[2]Refer to our arXiv version for the supplementary materials.

[3]Note that the activation statistics obtained in training time is used in the batch norm layer in test time.

[4]In the weight training phase, the original network architecture is utilized since only one of the branches is selected in the bit selection phase.

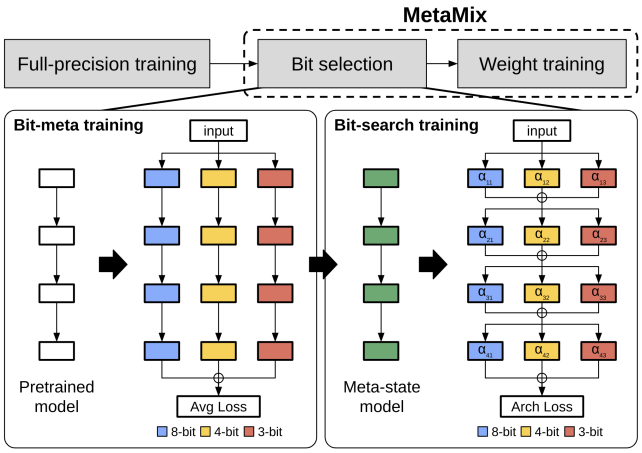


Figure 3: MetaMix flow diagram and working mechanism.

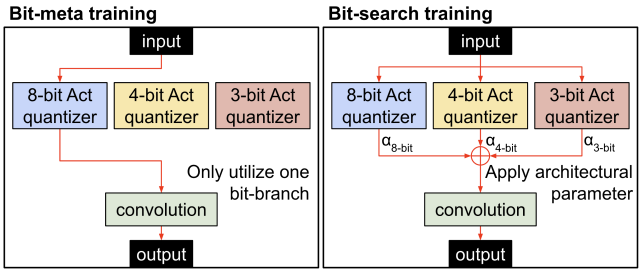


Figure 4: MetaMix block structure design and operations on bit selection phase. 'Act' represents activation.

bit selection phase in our case), we apply meta-learning to learn our mixed-precision-aware weights and thus this step is called bit-meta training.

Equation 1 shows the loss of bit-meta training. We select a bit-width b_i among B candidate bit-widths in end-to-end network and utilize it to build a single-precision network during the forward pass of training. After obtaining all the training losses (B losses from B candidate bit-widths), we utilize the total average loss in Equation 1 to update the network weight w.

$$\mathcal{L}(w) = \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}(w, b_i) \tag{1}$$

We call the mixed-precision-aware weights obtained from the bit-meta training *meta-state* and the model with the meta-state *meta-state model* as illustrated in Figure 3. Our key idea behind the bit-meta training is to minimizing the negative impact of activation bit-width to weight on bit selection phase as much as possible. The mitigation is realized by training the weights in the bit-meta training and fixing them during the bit-search training. Thus, unlike existing methods (all the methods in Figure 5) which suffer from activation instabilities by exploring bit selections under the weights affected by bit-width, our proposed method can mitigate activation instabilities, by keeping model to search bit selection on meta-state model, thereby outperforming the existing methods as will be shown in the experiments.

Note that both the iteration of bit-meta and bit-search training and the usage of fixed weights in bit-search training contribute to the reduction of activation instabilities during bit selection. Specifically, the bit-meta training can reduce activation instability due to bit selection (Figure 6), which benefits the subsequent bit-search training. In addition, the usage of fixed full-precision weights in the bit-search training allows the bit-search training to avoid activation instability due to both bit selection and weight quantization, which contributes to the quality of selected bit-widths (Figure 7).

Algorithm 1: Pseudo code of MetaMix (bit selection phase)

Input: the number of training iterations in 1-epoch $\mathcal{T}$, batched input image X, activation bit-width selection candidates $\{b_1, \cdots, b_B\}$ and the number of candidates B, per-bit & per-layer architectural parameter α, per-bit & per-layer activation quantizer $q()$, network weight w, number of layers in model L, L1 regularization term $r()$ and regularization scale factor λ_r

Output: trained weights, determined per-layer bit selections

function B-META-FWD(w, input, b_i, i)
  $x_0 = \text{input}$
  for $l = 1 : L$ **do**
    $x_l = \mathit{Forward}(w_l, q_{l,i}(x_{l-1}, b_i))$
  return x_L

function B-SEARCH-FWD(w, input, $\{b_1, \cdots, b_B\}$, α)
  $x_0 = \text{input}$
  for $l = 1 : L$ **do**
    $\bar{q}_l(x_{l-1}) = \sum_{i=1}^{B} \frac{\exp(\alpha_{l,b_i})}{\sum_{j=1}^{B}\exp(\alpha_{l,b_j})} \cdot q_{l,i}(x_{l-1}, b_i)$
    $x_l = \mathit{Forward}(w_l, \bar{q}_l(x_{l-1}))$
  return x_L

 # Bit selection phase
for $epoch = 1 : 2$ **do**
  for $iter = 1 : \mathcal{T}$ **do**
    # Bit-meta training
    for $i = 1 : B$ **do**
      $out =$ B-META-FWD (w, X_{iter}, b_i, i)
      $\mathcal{L}(w, b_i) = Loss(out)$
    $\mathcal{L}(w) = \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}(w, b_i)$
    $w \leftarrow \mathit{Backward}(\mathcal{L}(w))$ **# update weight**
    # Bit-search training
    if $epoch > 1$ **then**
      $out =$ B-SEARCH-FWD$(w, X_{iter}, \{b_1, \cdot\cdot, b_B\}, \alpha)$
      $\mathcal{L}(w, \alpha) = Loss(out)$
      $\mathcal{L}(\alpha) = \mathcal{L}(w, \alpha) + \lambda_r \cdot r(\alpha)$
      $\alpha \leftarrow \mathit{Backward}(\mathcal{L}(\alpha))$ **# update arch. param**
for $l = 1 : L$ **do**
  $bit\_sel_l = \text{argmax}_b \alpha_l$ **# max α as per-layer bit**
return W, bit_sel **# pass to next phase**

Bit-Search Training

Bit-search training learns an architectural parameter for per-layer bit-width probability on each of the branches in the

Phase	Epochs	Step / Network	Train or Fix?
Bit selection	epoch #1	Bit-meta training	Train FP weight and s_a with fixed α
	epoch #2	Bit-meta training $\rightleftarrows$ Bit-search training (two steps alternate for 1 iteration each)	Bit-meta: Train FP weight and s_a with fixed α Bit-search: Train α with fixed FP weight and s_a
Weight training	epoch #3 - #140	MobileNet-v2 and v3	Fine-tuning 4-bit weight, s_a and s_w Fixed per-layer activation bit-width (max α)
	epoch #3 - #90	ResNet-18	

Table 1: Detailed training process (α: architectural parameters, s_a: step sizes of per-layer activations, s_w: step sizes of weights)

block structure of Figure 4. Given an activation x, its quantized activation $\bar{q}(x)$ is calculated as follows.

$$\bar{q}(x) = \sum_{i=1}^{B} \frac{\exp(\alpha_{b_i})}{\sum_{j=1}^{B} \exp(\alpha_{b_j})} \cdot q_i(x, b_i) \tag{2}$$

Each branch with b_i-bits is assigned an architectural parameter α_{b_i}. We first obtain the branch-specific quantized activation, $q_i(x, b_i)$ and then calculate $\bar{q}(x)$ by weighting the branch activation with its softmax as shown in Equation 2.

Note that the branches are differently utilized on bit-meta and bit-search training. As Figures 3 and 4 (left) show, in bit-meta training, for a bit-width, only the associated branch is utilized without architectural parameter α_{b_i} to build a single-precision network where all the layers have the same bit-width. However, in bit-search training, as Equation 2 and Figures 3 and 4 (right) show, each branch obtains its own quantized result $q_i(x, b_i)$ on the given input x. All the branches are utilized by weighting softmax of assigned α_{b_i}.

Note also that, in this step, we utilize the model with fixed full-precision weights, i.e., the fixed meta-state model. Thus, during back-propagation, we update only the architectural parameters to minimize the training loss without updating the network weights. Equation 3 shows the training loss.

$$\mathcal{L}(\alpha) = \mathcal{L}(w, \alpha) + \lambda_r \cdot r(\alpha) \tag{3}$$

where $\mathcal{L}(w, \alpha)$ is the task loss and $r(\alpha)$ is an L1 regularization term used to restrict the total number of bits or computation cost within a given budget. λ_r is a scale factor. Equation 4 shows $r(\alpha)$ when a computation cost constraint, i.e., a target number of bit operations *t_bops* is given.

$$r(\alpha) = \left| \sum_{i=1}^{N} op_i \cdot b^w \cdot \sum_{j=1}^{B} hs(\alpha_i)_j \cdot b_j - t\_bops \right| \tag{4}$$

where op_i is the number of operations on layer i and b^w is the weight bit-width.[5] Since the per-layer bit-width of activation is finally determined by the bit-width of the branch having the top softmax score in Equation 2, it is required to obtain the bit-width of the top performer branch while making gradients flow through all the branches (Jang, Gu, and Poole 2017) in order to learn architectural parameters α. To do this, we use straight-through-softmax trick on *hs* in Equation 4. As Equation 5 shows, *hs* outputs one-hot vector of architectural parameter vector α (i.e., the element having the largest softmax function is assigned to one while the others to zero) in forward pass of training. However, in backward pass, we approximate the onehot to softmax results thereby allowing gradients to propagate through all the branches.

$$\begin{aligned} hs(\alpha_i) &= onehot(\alpha_i) \\ \nabla_{\alpha_j} hs(\alpha_i) &\approx \nabla_{\alpha_j} softmax(\alpha_i) \end{aligned} \tag{5}$$

The mixed-precision quantization of MetaMix looks similar to that of DNAS (Wu et al. 2018). Our key difference is the different utilization of branches to mitigate negative effect of bit-width on activation instability. MetaMix forms a combination of single bit-width end-to-end networks (Figure 3 left) to train the shared weights in bit-meta training. Then the trained model, i.e., the meta-state model is fixed and only the architectural parameters of all bit-widths are trained in bit-search training (Figure 3 right). However, DNAS forms per-branch weights with assigned bit-width and architectural parameters which can maximize the activation instability on training weights. After bit search, DNAS samples diverse architectures and exhaustively trains many model candidates, while MetaMix directly selects one final architecture for fine-tuning (in the weight training phase), which offers faster bit search. As a result, MetaMix enables better mixed-precision model (71.94% in 32.86GBOPs vs. 70.6% in 35.17GBOPs on ResNet-18) and faster bit-width searching (GPU hours of 13.4h vs. 40h on ResNet-18) and re-training (fine-tuning in our case) compared to DNAS.

Experiments

Training Details

Table 1 shows the details of training in our proposed method. As the table shows, in the first epoch, we perform bit-meta training which learns full-precision weights and the step sizes of activations. Specifically, on each branch of bit-width (in Figure 4), the activation is quantized to its associated bit-width while its associated step size is being trained. In the second epoch, we iterate bit-meta and bit-search training. In bit-search training, we fix the full-precision weights and step sizes obtained in the bit-meta training and learn only the architectural parameters for per-layer bit-width probabilities. After the per-layer bit-width is obtained, in the weight training phase, we fine-tune both network weights and the step sizes of weights and activations.

[5]Note that we utilize a single bit-width of weight in the entire network. Refer to supplementary for weight mixed-precision.

Method	Bits (A/W)	1st 8-bit A	1st 8-bit W	last 8-bit A	last 8-bit W	Top-1 (%)
Baseline	FP					71.9
PACT	4/4					61.40
DSQ	4/4					64.80
LLSQ	4/4					67.37
LSQ	4/4					69.5
LSQ+BR	4/4					70.4
OOQ	4/4	✓	✓	✓	✓	70.6
LCQ	4/4		✓		✓	70.8
SAT	4/4		✓	✓	✓	70.8
APoT	4/4		✓		✓	71.0
PROFIT	4/4		✓	✓	✓	71.56
PROFIT†	4/4	✓	✓	✓	✓	70.93
N2UQ	4/4					72.1
N2UQ†	4/4		✓	✓	✓	62.83
N2UQ†	4/4	✓	✓	✓	✓	52.12
MetaMix	3.98/4	✓	✓	✓	✓	**72.60**

Table 2: ImageNet comparison with single-precision quantization results on MobileNet-v2 (PACT (Choi et al. 2018), DSQ (Gong et al. 2018), LLSQ (Zhao et al. 2020), LSQ (Esser et al. 2020), LSQ+BR (Han et al. 2021), OOQ (Nagel et al. 2022), LCQ (Yamamoto 2021), SAT (Jin, Yang, and Liao 2019), APoT (Li, Dong, and Wang 2020), PROFIT (Park and Yoo 2020), N2UQ (Liu et al. 2022)). In column 'Bits', 'A' is for activation bit-width and 'W' is for weight bit-width. '1st 8-bit' and 'last 8-bit' columns show whether the activation (A) or weight (W) of the 1st or last layer is quantized to 8-bit or not. PROFIT† and N2UQ† are re-implemented version.

We evaluate the proposed method on ImageNet-1K (Deng et al. 2009). In MobileNet-v2 and v3, we use 8-bit, 4-bit, and 3-bit as candidate bit-widths of activations. In ResNet-18, we use 8-bit, 4-bit, and 2-bit as candidates. All weights are quantized to 4-bit. We also quantize 1st and last layers to 8-bit for both weights and activations, which offers fully quantized networks. Further details are in supplementary.

Comparison on Single-Precision Quantization

In this section, we compare single-precision quantization methods (PACT (Choi et al. 2018), LQ-Net (Zhang et al. 2019), DSQ (Gong et al. 2018), LLSQ (Zhao et al. 2020), QIL (Jung et al. 2018), LSQ (Esser et al. 2020), LSQ+BR (Han et al. 2021), OOQ (Nagel et al. 2022), LCQ (Yamamoto 2021), SAT (Jin, Yang, and Liao 2019), APoT (Li, Dong, and Wang 2020), PROFIT & DUQ (Park and Yoo 2020), N2UQ (Liu et al. 2022)) and ours using mixed-precision obtained with the target bit-width of 4-bit.[6]

Table 2 compares the accuracy of 4-bit level MobileNet-v2. MetaMix (72.60%) gives better accuracy than PROFIT (71.56%) and N2UQ (72.1%). When the 1st layer input is quantized in PROFIT†, MetaMix outperforms it by a larger margin (72.60% vs. 70.93%). N2UQ† shows large accuracy drop with quantized inputs and weights of 1st and last layers.

[6]We use two decimal digits except some values having a single decimal digit in the original papers.

Method	Bits (A/W)	1st 8-bit A	1st 8-bit W	last 8-bit A	last 8-bit W	Top-1 (%)
Baseline	FP					75.3
PACT	4/4					70.16
DUQ	4/4		✓	✓	✓	71.01
PROFIT	4/4		✓	✓	✓	73.81
PROFIT†	4/4	✓	✓	✓	✓	71.57
MetaMix	3.83/4	✓	✓	✓	✓	**74.24**

Table 3: ImageNet comparison with single-precision quantization results on MobileNet-v3 (large) (PACT (Choi et al. 2018), DUQ & PROFIT (Park and Yoo 2020)).

Method	Bits (A/W)	1st 8-bit A	1st 8-bit W	last 8-bit A	last 8-bit W	Top-1 (%)
Baseline	FP					70.5
PACT	4/4					69.2
LQ-Net	4/4					69.3
DSQ	4/4					69.56
LLSQ	4/4					69.84
QIL	4/4					70.1
APoT	4/4		✓		✓	70.7
LSQ	4/4					71.1
LCQ	4/4		✓		✓	71.5
N2UQ†	4/4					71.91
N2UQ†	4/4		✓	✓	✓	70.60
N2UQ†	4/4	✓	✓	✓	✓	NC
MetaMix	3.85/4	✓	✓	✓	✓	**71.94**
MetaMix	3.85/3	✓	✓	✓	✓	**70.69**
MetaMix	3.85/2	✓	✓	✓	✓	**69.45**

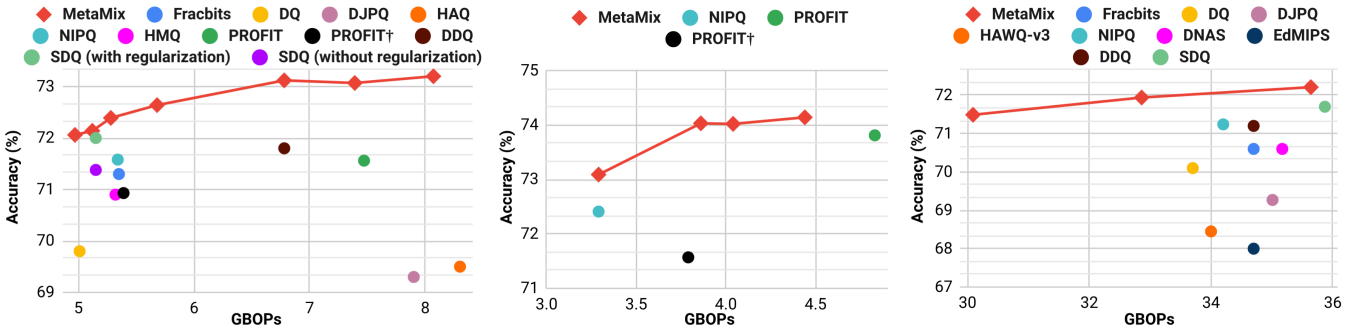
Table 4: ImageNet comparison with single-precision quantization results on ResNet-18 (PACT (Choi et al. 2018), LQ-Net (Zhang et al. 2019), DSQ (Gong et al. 2018), LLSQ (Zhao et al. 2020), QIL (Jung et al. 2018), APoT (Li, Dong, and Wang 2020), LSQ (Esser et al. 2020), LCQ (Yamamoto 2021), N2UQ (Liu et al. 2022)). 'NC' means not converged.

Table 3 compares the accuracy of 4-bit level MobileNet-v3. MetaMix outperforms PROFIT (74.24% vs. 73.81%) and, especially, PROFIT† with 8-bit input by a large margin (74.24% vs. 71.57%).

Table 4 gives a comparison of 4-bit level ResNet-18. The state-of-the-art method, N2UQ†, did not converged with quantized inputs and weights of 1st and last layers. MetaMix (71.94%) offers better accuracy than N2UQ† (70.60%) despite the fact that N2UQ† adopts non-uniform quantization while not quantizing the input activation of 1st layer.

Comparison on Mixed-Precision Quantization

Figure 5 shows the accuracy of mixed-precision quantization methods in diverse BOPs (bit operations) budget. Note that MetaMix uses only a restricted set of bit-widths (8-bit, 4-bit,



(a) MobileNet-v2 (b) MobileNet-v3 (c) ResNet-18

Figure 5: ImageNet top-1 accuracy vs. BOPs on (a) MobileNet-v2, (b) MobileNet-v3 (large), (c) ResNet-18 (HMQ (Habi, Jennings, and Netzer 2020), DQ (Uhlich et al. 2020), DJPQ (Wang, Lu, and Blankevoort 2020), HAQ (Wang et al. 2019), NIPQ (Shin et al. 2023), Fracbits (Yang and Jin 2021), DDQ (Zhang et al. 2021), SDQ (Huang et al. 2022), DNAS (Wu et al. 2018), HAWQ-v3 (Yao et al. 2021), EdMIPS (Cai and Vasconcelos 2020) and state-of-the-art single-precision quantization PROFIT (Park and Yoo 2020)). PROFIT† quantizes, in 8-bits, the input activation of 1st layer.

3-bit in MobileNet-v2 and v3, 8-bit, 4-bit, 2-bit in ResNet-18) in search space while others use all the integer bits from 2-bit to 8-bit (HMQ, HAQ, Fracbits, DDQ, SDQ) or 2-bit to 10-bit (DQ, DJPQ, NIPQ). The figure shows that MetaMix offers new state-of-the-art results while pushing the boundary of mixed-precision quantization.

In MobileNet-v2, MetaMix (72.14% in 5.12GBOPs) shows comparable result to the state-of-the-art SDQ (72.0% in 5.15GBOPs) and NIPQ (71.58% in 5.34GBOPs) which uses more choices of integer bit. The figure also shows a comparison with the single-precision state-of-the-art PROFIT. MetaMix shows larger gain, i.e., 72.39% in 5.28GBOPs (MetaMix) vs. 70.93% in 5.39GBOPs (PROFIT†), with the quantized 1st layer input.

As reported in PROFIT (Park and Yoo 2020), the depth-wise layers, near the input and output of the network, exhibit large activation instability due to weight quantization when they are quantized to 4-bit. PROFIT shows early stopping of those layers' training at the end of training improves final batch norm parameters. We also tried to apply PROFIT to MetaMix and could not obtain improvements. It is because those sensitive layers tend to be assigned high precision in mixed-precision quantization (as in Figure 7 (b)) thereby reducing their negative effect on activation instability.

In MobileNet-v3, MetaMix shows better accuracy (73.09% in 3.29GBOPs) than mixed-precision NIPQ (72.41% in 3.29GBOPs). Compared to PROFIT (73.81% in 4.83GBOPs) and PROFIT† (71.57% in 3.79GBOPs), MetaMix shows superior results (74.14% in 4.44GBOPs and 73.09% in 3.29GBOPs, respectively).

In ResNet-18, MetaMix pushes the boundary of mixed-precision quantization towards smaller BOPs. Overall, MetaMix offers by more than 0.5% better accuracy than other methods including SDQ (71.7% in 35.87GBOPs) and NIPQ (71.24% in 34.2GBOPs).

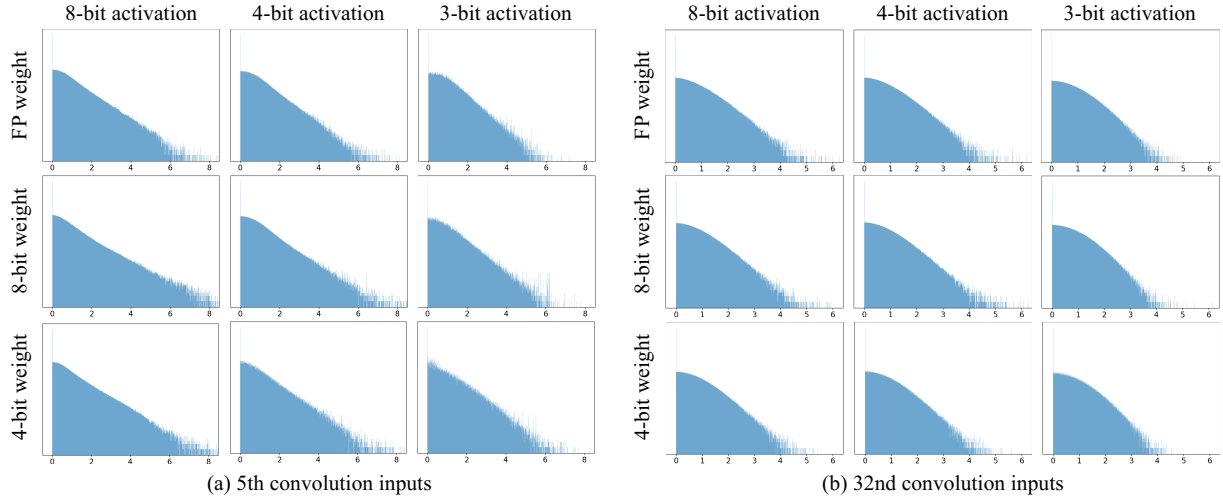
Comparison with SOTA mixed-precision quantization SDQ applies strong regularization technique (i.e., 1.5% improvement on ResNet-18). In order to compare the sole effect of mixed-precision method itself, we compare MetaMix with SDQ without applying strong regularization. In Figure 5 (a), on MobileNet-v2, SDQ without strong regularization (71.38% on 5.15G) shows dropped accuracy from SDQ with strong regularization (72.0% on 5.15G). However, MetaMix shows better accuracy (72.14% on 5.12G) without any regularization. Note that we also have far better accuracy in ResNet-18 (72.21% in 35.64GBOPs vs. 71.7% in 35.87GBOPs), even compared with strong regularization.

Method	Searching		Re-training
	GPU hours	Epochs	Epochs
DNAS	40	60	120
SPOS	312	120	240
EdMIPS	36	25	95
MetaMix	13.4	2	88

Table 5: Training cost comparison with NAS-based methods (DNAS (Wu et al. 2018), SPOS (Guo et al. 2020), EdMIPS (Cai and Vasconcelos 2020)) on ResNet-18 (Note that DNAS uses a small subset of ImageNet in searching).

Training Cost

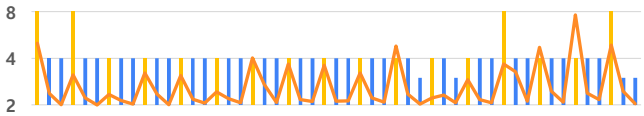
Table 5 compares training cost with NAS based mixed-precision quantization methods. The table shows MetaMix offers faster bit search (>3×) and re-training (weight training in MetaMix) than existing methods. The fast training advantages of MetaMix mostly come from the usage of meta-state in the bit selection phase. Mixed-precision-aware initialization of weights in the weight training phase also contributes to faster training than the re-training of NAS-based methods which discard the weights obtained in bit search and re-train from scratch with the selected bit-width.



(a) 5th convolution inputs (b) 32nd convolution inputs

Figure 6: Input activation distribution before quantizer with {8, 4, 3}-bit bit-meta training of MobileNet-v2. Formats are the same as in Figure 1. All distributions show similar variances across different bit-widths of activations and weights in contrast to Figure 1. This demonstrates that bit-meta training helps mitigate activation instability due to bit selection.

(a) MetaMix without fixed meta-state (71.93%, 5.33GBOPs)



(b) MetaMix (72.39%, 5.28GBOPs)

Figure 7: Per-layer bit-width (yellow for depth-wise convolution layers and blue for the others) of MetaMix obtained (a) without and (b) with fixed meta-state on MobileNet-v2. (b) also shows the sum of hessian trace divided by the number of per-layer operations (orange line).

Ablation Study

Effects of Meta-State Model

Figure 6 shows the effects of bit-meta training on activations in MobileNet-v2. The activation distributions exhibit much better consistency across different bit-widths than those in Figure 1. The consistency enables stable batch norm statistics when changing bit-width as in Figure 2 (bottom). As such, the meta-state, which is the outcome of bit-meta training, offers consistent activation distributions across different activation bit-widths, which helps the subsequent bit-search training make high-quality bit selections by reducing the negative effect of activation instability due to bit selection.

As explained before, we use the fixed full-precision weights (i.e., meta-state) in bit-search training. Figure 6 also demonstrates the benefit of full-precision weights since, in the full-precision weight cases, activation distributions exhibit much stronger consistency across different activation bit-widths than in the cases of low bit-width weights.

Effects of Fixed Meta-State Model

In the second epoch of bit selection phase (in Table 1), we iterate bit-meta training and bit-search training. The bit-search training utilizes the fixed full-precision weights after the previous bit-meta training. Thus, given the same mini-batch, there is no activation instability due to both bit selection and weight quantization in the subsequent bit-search training, which enables the bit-search training to benefit from the stable distribution of activation (as in the bottom of Figure 2 and Figure 6) enabled by the bit-meta training.

Figure 7 illustrates the effects of fixed meta-state model in bit-search training on MobileNet-v2. The figure compares the per-layer bit-width results of two cases: (a) without and (b) with the fixed meta-state. We set the same budget of computation cost (5.3GBOPs) in both cases. When we do not fix but train the network weights (after initializing them with the meta-state) during bit-search training, as Figure 7 (a) shows, MetaMix tends to select 4-bit in most of layers. However, as Figure 7 (b) shows, when the fixed meta-state is used, MetaMix tends to select higher bit-width for early and late depth-wise convolution layers which are known to be difficult to quantize in 4-bit while lowering the bit-widths of some intermediate layers, in return, finally meeting the budget. As a result, MetaMix with the fixed meta-state in bit-search training gives better bit selections (72.39%) than without the fixed meta-state (71.93%). Figure 7 (b) also shows the relationship between the selected per-layer bit-width, Hessian, and OPs. For details, refer to supplementary.

Conclusion

In this paper, we presented a novel training method, MetaMix, to address activation instability in mixed-precision quantization. It consists of bit selection and weight

training phases. In the bit selection phase, we determine the per-layer bit-width of activation by iterating the bit-meta training and the bit-search training, which reduce activation instability due to both bit selection and weight quantization thereby enabling fast and high-quality bit selection. The subsequent weight training phase offers fast fine-tuning of network weights and step sizes since they are initialized in a mixed-precision-aware manner in the previous bit selection phase. Our experiments on ImageNet show that MetaMix outperforms, in terms of accuracy vs. cost, single- and mixed-precision SOTA methods on efficient and hard-to-quantize models, i.e., MobileNet-v2 & v3 and ResNet-18.

Acknowledgments

We appreciate valuable comments from Dr. Jiyang Kang, Wonpyo Park, and Yicheng Fan at Google. This work was supported in part by IITP grant funded by the Korean government (MSIT, 2021-0-00105 Development of Model Compression Framework for Scalable On-Device AI Computing on Edge Applications).

References

Cai, H.; Zhu, L.; and Han, S. 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. *International Conference on Learning Representations (ICLR)*.

Cai, Z.; and Vasconcelos, N. 2020. Rethinking Differentiable Search for Mixed-Precision Neural Networks. *Computer Vision and Pattern Recognition (CVPR)*.

Choi, J.; Wang, Z.; Venkataramani, S.; Chuang, P. I.-J.; Srinivasan, V.; and Gopalakrishnan, K. 2018. PACT: Parameterized Clipping Activation for Quantized Neural Networks. *arXiv:1805.06085*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition (CVPR)*.

Dong, Z.; Yao, Z.; Cai, Y.; Arfeen, D.; Gholami, A.; Mahoney, M. W.; and Keutzer, K. 2020. HAWQ-V2: Hessian Aware trace-Weighted Quantization of Neural Networks. *Advances in Neural Information Processing Systems (NeurIPS)*.

Dong, Z.; Yao, Z.; Gholami, A.; Mahoney, M.; and Keutzer, K. 2019. HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision. *International Conference on Computer Vision (ICCV)*.

Défossez, A.; Adi, Y.; and Synnaeve, G. 2022. Differentiable Model Compression via Pseudo Quantization Noise. *Transactions on Machine Learning Research (TMLR)*.

Elthakeb, A. T.; Pilligundla, P.; Mireshghallah, F.; Yazdanbakhsh, A.; and Esmaeilzadeh, H. 2018. ReLeQ: A Reinforcement Learning Approach for Deep Quantization of Neural Networks. *Advances in Neural Information Processing Systems (NeurIPS) Workshop on ML for Systems*.

Esser, S. K.; McKinstry, J. L.; Bablani, D.; Appuswamy, R.; and Modha, D. S. 2020. Learned Step Size Quantization. *International Conference on Learning Representations (ICLR)*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *International Conference on Machine Learning (ICML)*.

Gong, R.; Liu, X.; Jiang, S.; Li, T.; Hu, P.; Lin, J.; Yu, F.; and Yan, J. 2018. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks. *International Conference on Computer Vision (ICCV)*.

Guo, Z.; Zhang, X.; Mu, H.; Heng, W.; Liu, Z.; Wei, Y.; and Sun, J. 2020. Single Path One-Shot Neural Architecture Search with Uniform Sampling. *European Conference on Computer Vision (ECCV)*.

Habi, H. V.; Jennings, R. H.; and Netzer, A. 2020. HMQ: Hardware Friendly Mixed Precision Quantization Block for CNNs. *European Conference on Computer Vision (ECCV)*.

Han, T.; Li, D.; Liu, J.; Tian, L.; and Shan, Y. 2021. Improving Low-Precision Network Quantization via Bin Regularization. *International Conference on Computer Vision (ICCV)*.

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; Le, Q. V.; and Adam, H. 2019. Searching for MobileNetV3. *International Conference on Computer Vision (ICCV)*.

Howard, A.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861*.

Hu, S.; Xie, S.; Zheng, H.; Liu, C.; Shi, J.; Liu, X.; and Lin, D. 2020. DSNAS: Direct Neural Architecture Search without Parameter Retraining. *Computer Vision and Pattern Recognition (CVPR)*.

Huang, X.; Shen, Z.; Li, S.; Liu, Z.; Hu, X.; Wicaksana, J.; Xing, E.; and Cheng, K.-T. 2022. SDQ: Stochastic Differentiable Quantization with Mixed Precision. *International Conference on Machine Learning (ICML)*.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. *International Conference on Learning Representations (ICLR)*.

Jin, Q.; Yang, L.; and Liao, Z. 2019. Towards Efficient Training for Neural Network Quantization. *arXiv:1912.10207*.

Jung, S.; Son, C.; Lee, S.; Son, J.; Kwak, Y.; Han, J.-J.; Hwang, S. J.; and Choi, C. 2018. Learning to Quantize Deep Networks by Optimizing Quantization Intervals with Task Loss. *Computer Vision and Pattern Recognition (CVPR)*.

Kim, H.-B.; Park, E.; and Yoo, S. 2022. BASQ: Branch-wise Activation-clipping Search Quantization for Sub-4-bit Neural Networks. *European Conference on Computer Vision (ECCV)*.

Li, R.; Wang, Y.; Liang, F.; Qin, H.; Yan, J.; and Fan, R. 2019. Fully Quantized Network for Object Detection. *Computer Vision and Pattern Recognition (CVPR)*.

Li, Y.; Dong, X.; and Wang, W. 2020. Additive Powers-of-Two Quantization: An Efficient Non-uniform Discretization

for Neural Networks. *International Conference on Learning Representations (ICLR)*.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. *International Conference on Learning Representations (ICLR)*.

Liu, Z.; Cheng, K.-T.; Huang, D.; Xing, E.; and Shen, Z. 2022. Nonuniform-to-Uniform Quantization: Towards Accurate Quantization via Generalized Straight-Through Estimation. *Computer Vision and Pattern Recognition (CVPR)*.

Nagel, M.; Fournarakis, M.; Bondarenko, Y.; and Blankevoort, T. 2022. Overcoming Oscillations in Quantization-Aware Training. *International Conference on Machine Learning (ICML)*.

NVIDIA. 2018. NVIDIA turing architecture white paper. https://images.nvidia.com/aem-dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf. Accessed: 2023-08-15.

Park, E.; and Yoo, S. 2020. PROFIT: A Novel Training Method for sub-4-bit MobileNet Models. *European Conference on Computer Vision (ECCV)*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized Evolution for Image Classifier Architecture Search. *AAAI Conference on Artificial Intelligence*.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *Computer Vision and Pattern Recognition (CVPR)*.

Sharma, H.; Park, J.; Suda, N.; Lai, L.; Chau, B.; Kim, J. K.; Chandra, V.; and Esmaeilzadeh, H. 2018. Bit Fusion: Bit-Level Dynamically Composable Architecture for Accelerating Deep Neural Networks. *International Symposium on Computer Architecture (ISCA)*.

Shin, J.; So, J.; Park, S.; Kang, S.; Yoo, S.; and Park, E. 2023. NIPQ: Noise proxy-based Integrated Pseudo-Quantization. *Computer Vision and Pattern Recognition (CVPR)*.

Uhlich, S.; Mauch, L.; Cardinaux, F.; Yoshiyama, K.; Garcia, J. A.; Tiedemann, S.; Kemp, T.; and Nakamura, A. 2020. Mixed Precision DNNs: All you need is a good parametrization. *International Conference on Learning Representations (ICLR)*.

Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; and Han, S. 2019. HAQ: Hardware-Aware Automated Quantization with Mixed Precision. *Computer Vision and Pattern Recognition (CVPR)*.

Wang, Y.; Lu, Y.; and Blankevoort, T. 2020. Differentiable Joint Pruning and Quantization for Hardware Efficiency. *European Conference on Computer Vision (ECCV)*.

Wu, B.; Wang, Y.; Zhang, P.; Tian, Y.; Vajda, P.; and Keutzer, K. 2018. Mixed Precision Quantization of ConvNets via Differentiable Neural Architecture Search. *arXiv:1812.00090*.

Yamamoto, K. 2021. Learnable Companding Quantization for Accurate Low-bit Neural Networks. *Computer Vision and Pattern Recognition (CVPR)*.

Yang, L.; and Jin, Q. 2021. FracBits: Mixed Precision Quantization via Fractional Bit-Widths. *AAAI Conference on Artificial Intelligence*.

Yao, Z.; Dong, Z.; Zheng, Z.; Gholami, A.; Yu, J.; Tan, E.; Wang, L.; Huang, Q.; Wang, Y.; Mahoney, M. W.; and Keutzer, K. 2021. HAWQV3: Dyadic Neural Network Quantization. *International Conference on Machine Learning (ICML)*.

Yu, H.; Han, Q.; Li, J.; Shi, J.; Cheng, G.; and Fan, B. 2020. Search What You Want: Barrier Panelty NAS for Mixed Precision Quantization. *European Conference on Computer Vision (ECCV)*.

Zhang, D.; Yang, J.; Ye, D.; and Hua, G. 2019. Differentiable Soft Quantization: Bridging Full-Precision and Low-Bit Neural Networks. *European Conference on Computer Vision (ECCV)*.

Zhang, Z.; Shao, W.; Gu, J.; Wang, X.; and Luo, P. 2021. Differentiable Dynamic Quantization with Mixed Precision and Adaptive Resolution. *International Conference on Machine Learning (ICML)*.

Zhao, X.; Wang, Y.; Cai, X.; Liu, C.; and Zhang, L. 2020. Linear Symmetric Quantization of Neural Networks for Low-precision Integer Hardware. *International Conference on Learning Representations (ICLR)*.

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