MSQ: Memory-Efficient Bit Sparsification Quantization

Seokho Han[1*]   Seoyeon Yoon[1*]   Jinhee Kim[1]   Dongwei Wang[2]
Kang Eun Jeon[1✉]   Huanrui Yang[2✉]   Jong Hwan Ko[1✉]

[1]Department of Electrical and Computer Engineering, Sungkyunkwan University, Korea
[2]Department of Electrical and Computer Engineering, University of Arizona, USA

{beppa2396, syy000405, a2jinhee, kejeon, jhko}@skku.edu
{dongweiw, huanruiyang}@arizona.edu

Abstract

As deep neural networks (DNNs) see increased deployment on mobile and edge devices, optimizing model efficiency has become crucial. Mixed-precision quantization is widely favored, as it offers a superior balance between efficiency and accuracy compared to uniform quantization. However, finding the optimal precision for each layer is challenging. Recent studies utilizing bit-level sparsity have shown promise, yet they often introduce substantial training complexity and high GPU memory requirements. In this paper, we propose Memory-Efficient Bit Sparsification Quantization (MSQ), a novel approach that addresses these limitations. MSQ applies a round-clamp quantizer to enable differentiable computation of the least significant bits (LSBs) from model weights. It further employs regularization to induce sparsity in these LSBs, enabling effective precision reduction without explicit bit-level parameter splitting. Additionally, MSQ incorporates Hessian information, allowing the simultaneous pruning of multiple LSBs to further enhance training efficiency. Experimental results show that MSQ achieves up to 8.00× reduction in trainable parameters and up to 86% reduction in training time compared to previous bit-level quantization, while maintaining competitive accuracy and compression rates. This makes it a practical solution for training efficient DNNs on resource-constrained devices.

1. Introduction

With the increasing adoption of deep neural networks (DNNs) in mobile and edge devices [12, 20], optimizing their efficiency has become a major area of research due to their limited computational and memory resources [2]. Quantization - transforming model weights (and activa-

tions) from high-precision floating-point to low-precision fixed-point values [1, 3, 28, 30, 31] - has been widely adopted to address this challenge. This transformation not only reduces the storage requirements of models but also enables the use of fixed-point arithmetic units, significantly lowering energy and area costs while improving computational speed [10].

However, quantization introduces noises in model weights and activations due to the discrepancy between the original floating-point and low-precision values, potentially degrading model performance. When every layer of a model is quantized to the same low precision, the noise accumulated from some sensitive layers can lead to substantial accuracy drops. Mixed-precision quantization, which assigns different precision levels to each layer, has demonstrated improvements in achieving higher accuracies with smaller model sizes [24].

Despite its advantages in performance and compression rate, mixed-precision quantization is challenging to implement due to the vast search space required to determine the optimal bit scheme. Previous approaches have utilized reinforcement learning-based searches [23] or computed sensitivity statistics on pre-trained models [5, 6] to assign higher precision to sensitive layers and lower precision to less critical ones. However, these search-based methods are resource-intensive and fail to capture sensitivity changes during quantization-aware finetuning. To address this, dynamic mixed-precision quantization adjustment via bit-level structural sparsity has gained attention [26, 27]. These methods induce certain bits in the fixed-point weight representation to zeros, enabling simultaneous weight training and quantization scheme adjustment. However, previous bit-level training methods require instantiating a trainable variable for each individual bit, leading to extended training times and high GPU memory consumption.

This work aims to mitigate the memory and training cost challenges of the bit-level training to achieve mixed-precision quantization scheme more efficiently. We propose

---

*: Equal contributions
✉: Corresponding authors

Memory-Efficient Bit Sparsification Quantization (MSQ), a method to induce bit-level sparsity without explicitly introducing bit-level trainable parameters. MSQ derives LSB sparsity and aims to prune it. To achieve this, it employs a novel *round-clamp quantizer* provides a gradient direction for LSB sparsification. Furthermore, ℓ_1 regularization is applied to the computed LSB to effectively induce sparsity and, consequently, enable precision reduction. Additionally, MSQ incorporates Hessian information to account for layer sensitivity, enabling faster bit pruning rate on insensitive layers for greater training efficiency.The experiment examines training efficiency and the accuracy-compression tradeoff. Training with fewer parameters ,due to the absence of bit-level splitting, achieves significant memory savings and reduces training time by up to 86% compared to the traditional bit-level splitting approach, while experiments on ResNet and ViT models demonstrate a comparable accuracy-compression tradeoff.

Our contributions with MSQ are as follows:

- Significantly reducing bit-level training and sparsification cost by mitigating explicit bit splitting.
- Introducing a novel round-clamp quantizer for effective LSB computation and sparsity-inducing regularization.
- Extending bit-level quantization to both heterogeneous CNNs (e.g., MobileNetV3) and complex architectures such as Vision Transformers (ViTs).

2. Related Works

Quantization and quantization-aware training. Quantization techniques transform floating-point weight parameters into integer representations with reduced numerical precision. Though post-training quantization, which directly applies quantization to a pre-trained model, has undergone tremendous improvements [8, 16, 18, 25], finetuning is still required under ultra-low precision scenarios to prevent substantial model performance degradation. Consequently, quantization-aware training (QAT) techniques have been investigated to finetune quantized models under low-precisions. Since quantized weights assume discrete values, traditional gradient-based optimization methods, designed for continuous parameter spaces, are not directly applicable to training quantized models [3, 5, 27, 31]. To address this challenge, algorithms such as DoReFa-Net [31] employ the straight-through estimator (STE) to allow continuous gradient accumulation in the floating-point format. The forward and backward computation of the quantization-aware training process can be expressed as:

$$\mathbf{W_n} = \frac{1}{2^n - 1} \text{Round}[(2^n - 1)\mathbf{W}], \quad (1)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{W_n}}, \quad (2)$$
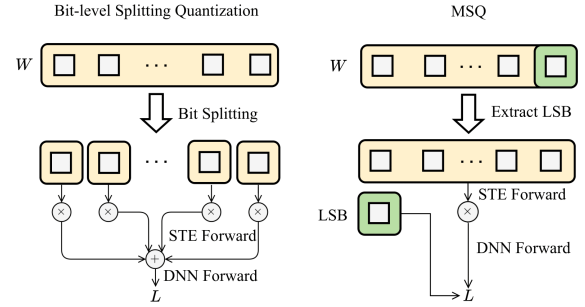

Figure 1. Work flow of bit-level quantization vs. MSQ.

where a normalized floating-point weight $\mathbf{W}$ is kept throughout the entire training. Along the forward pass, the STE will quantize $\mathbf{W}$ to an n-bit fixed-point representation $\mathbf{W_n}$, which will be used to compute the model output and loss $\mathcal{L}$. During the backward pass, the STE will directly pass the gradient w.r.t. $\mathbf{W_n}$ onto $\mathbf{W}$, which enables $\mathbf{W}$ to be updated with the standard gradient-based optimizer.

Mixed-precision quantization is explored following the observation that different layers in the model contributes differently to the model loss [5, 23]. A central challenge in mixed-precision quantization research is designing an optimal set of bit schemes to balance performance and model size. Initial approaches relied on manual heuristics, such as setting higher precision for the first and last layers. Search-based methods, like HAQ [23], use reinforcement learning to determine the quantization scheme. This process can be costly, especially for deeper models with a large search space. Other methods attempt to measure each layer's sensitivity directly, using metrics such as Hessian eigenvalue or trace [5, 6]. However, these approaches only account for the sensitivity of a fully trained, high-precision model and do not consider sensitivity changes that occur during quantization or updates in quantization-aware training.

Bit-level quantization. Bit-level sparsity quantization (BSQ) [27] was the first method to simultaneously apply a mixed-precision quantization scheme and train a quantized DNN model within a single training phase. In BSQ, each bit in the quantized model is treated as an independent trainable variable, where bit-level structural sparsity is induced in training to achieve mixed-precision quantization schemes. Although BSQ has shown strong empirical results, it requires straight-through gradient estimation for the bit variables in the rounded bit-level representation, which can reduce accuracy. Additionally, hard precision adjustments applied through bit pruning during training can compromise convergence stability. To mitigate this instability, Continuous Sparsification Quantization (CSQ) [26] was introduced, which smooths both bit-level training and precision adjustments using continuous sparsification [21].
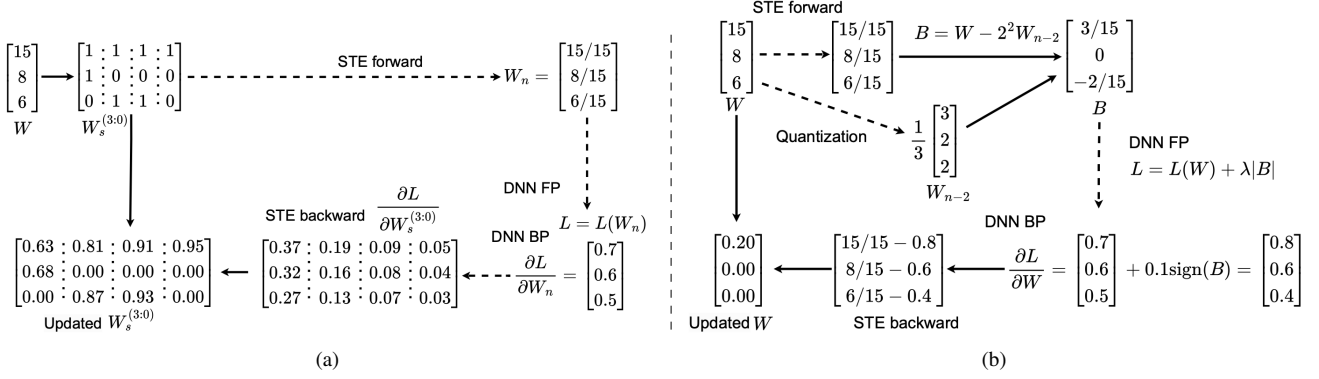
Figure 2. Comparison of bit-level splitting quantization and MSQ. (a) Training process of the bit-level model weight with STE.(b) Training process of the MSQ with LSB regularization.

While bit-level quantization allows for simultaneous mixed-precision quantization scheme discovery and parameter training, it demands substantial resources, including increased time and GPU memory, due to the large number of trainable parameters required for bit-level treatment as shown in Fig. 1. In this work, we reduce these resource demands by eliminating the need for multiplied parameters and instead applying a direct LSB computation and regularization from original trainable parameters, resulting in lower training costs compared to prior bit-level quantization methods.

3. Method

In this work, we propose MSQ, an efficient training algorithm to achieve bit-level sparsity for mixed-precision quantization scheme. MSQ mitigates the burden of having independent bit-level trainable variables as in BSQ [27] and CSQ [26] by deriving the sparsity of LSBs and prune them. To achieve this, we introduce a novel RoundClamp quantizer that supports bipartite bit-slicing for LSB computation and regularization, and regularization in Sec. 3.1 We then propose the Hessian-aware aggressive pruning technique to control the layer-wise bit reduction speed in Sec. 3.2 to reach the targeted model compression ratio with less training epochs. Finally, the full procedure and details of the proposed MSQ algorithm is discussed in Sec. 3.3. The training process of MSQ is illustrated in Fig. 2.

3.1. Bipartite Bit Slicing

RoundClamp quantizer design. In previous work of BSQ and CSQ, we observe that most precision reduction happens when LSBs are removed. This comes from the fact that LSBs has the minimal impact on the quantized weight values, therefore are induced to zeros more easily by the sparsity-inducing regularizer.

To this end, we first propose a bipartite bit-slicing method that can directly compute the values of the LSBs

from a model weight element. Specifically, consider a weight element $\mathbf{W}$ scaled to $[0, 1]$. We want to quantize $\mathbf{W}$ to a n-bit fixed point number $\mathbf{W}_n$, and compute the value represented by the k LSBs of $\mathbf{W}_n$, denoted as $\mathbf{B}_k$. Note that the quantized weight is equivalent to the summation of $\mathbf{B}_k$ and the top $(n-k)$ MSBs of $\mathbf{W}_n$ shifted left by k bits. We therefore propose to apply a quantizer where the top $(n-k)$ MSBs of $\mathbf{W}_n$ is exactly the $(n-k)$-bit quantized $\mathbf{W}_{n-k}$, so that $\mathbf{B}_k$ can be easily computed from $\mathbf{W}$ as

$$\mathbf{B}_k = \mathbf{W}_n - 2^k \mathbf{W}_{n-k}. \tag{3}$$

The DoReFa [31] quantizer, as discussed in Eq. (1), is the most common linear quantizer used in previous work. However, when observing the $\mathbf{B}_k$ computation results under the DoReFa quantizer, as illustrated in Fig. 3(a), we notice two issues hindering its usage in inducing bit-level. Firstly, the direction of inducing $\mathbf{B}_k$ to be zero is pointing towards the negative direction for most of the values of $\mathbf{W}$. This will induce the value of $\mathbf{W}$ to be constantly smaller as the training proceed, ultimately deviating away from its optimal value. Secondly, due to the use of a scaling factor of $(2^n - 1)$ in the rounding function, the quantization bin boundaries under different quantization precision are not well aligned. For example, some values quantized to "110" under 3-bit are mapped to "10" instead of "11" under 2-bit, leading to an error in the LSB computation.

To address this limitation, we introduce the RoundClamp quantizer, a novel quantization scheme designed to effectively compute LSBs for bit-level sparsity exploration. We adjust the scaling factor in the rounding function to be 2^n instead of $(2^n - 1)$, which leads to the formulation of the quantizer as

$$\mathbf{W}_n = q_r(\mathbf{W}; n) = \frac{1}{2^n - 1} \min(\lceil 2^n \mathbf{W} \rfloor, 2^n - 1), \quad (4)$$

where $\lceil \cdot \rfloor$ denotes the rounding function. Note that a clamping is needed to ensure the quantized value stays in the valid range of $[0, 1]$.
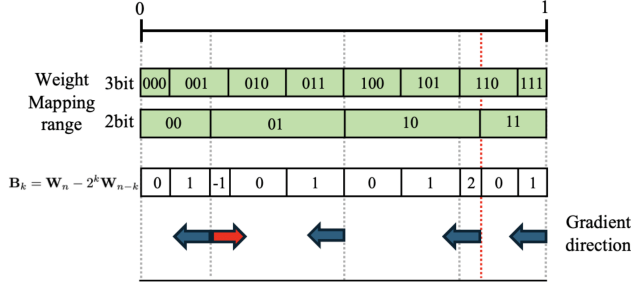
(a) Except for 001, gradient directions of other LSB-nonzero values do not work properly, as they only have one decreasing direction. It even has a gradient for 110, which should not exist.
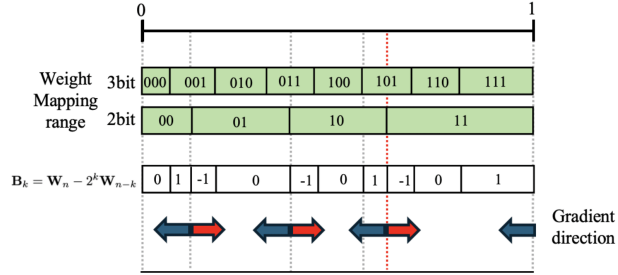
(b) All gradient directions of LSB-nonzero values work well.

Figure 3. Three-bit and two-bit quantized weight mapping range of (a) DoReFa quantizer and (b) round-clamp quantizer.

As illustrated in Fig. 3(b), RoundClamp adjusts the quantization bin boundaries of $(n-1)$-bit quantized weights to align with the midpoint of the quantization bins in n-bit quantization. This ensures that for those weights where the quantized value having nonzero LSBs, they have the chances to be round both up or down to the nearest bin with zero LSBs. When sparsity-inducing regularizer is applied to the computed LSBs to induce sparsity through training, we observe that the gradient directed by the RoundClamp quantizer can lead to zero LSBs and precision reduction more effectively.

The difference in the behavior of the two quantizers is evident in the resulting weight distributions after training, as shown in Fig. 4. In the case of DoReFa [31], depicted in Fig. 4(a), the weight distribution exhibits pronounced spikes at zero. This outcome arises due to the dominance of the negative gradient in the process of sparsifying the nonzero LSBs. In contrast, the RoundClamp quantizer produces a weight distribution characterized by higher densities at LSB-zero positions and lower densities at LSB-nonzero positions. This result aligns with the intended design, effectively driving the weights toward near-zero LSB values, as desired.

LSBs Computation and Regularization. With the Round-Clamp quantizer, we can compute the LSB of floating-point
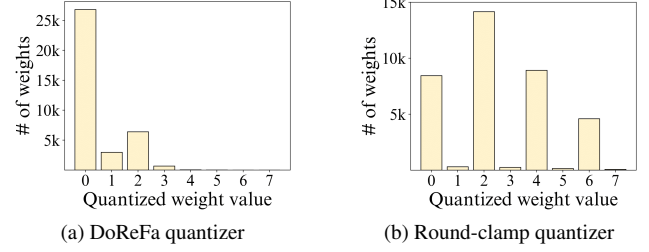

(a) DoReFa quantizer

(b) Round-clamp quantizer

Figure 4. Weight distribution of training with (a) DoReFa and (b) round-clamp quantizer. Both weight distributions are the results of 179 epochs, right before pruning.

weight $\mathbf{W}$ under n-bit quantization as

$$\mathbf{B}_k = \mathbf{W} - 2^k q_r(\mathbf{W}; n - k). \tag{5}$$

$\mathbf{B}_k$ is a continuous function with respect to the weight $\mathbf{W}$. Through this process, the LSB can be extracted without bit-level splitting.

To enforce LSB sparsity and precision reduction, we apply the ℓ_1 regularization to the computed LSB across all layers during training. The regularization term is expressed as:

$$\mathrm{R}(\mathbf{B}_k) = \sum_{\forall l} |\mathbf{B}_k^{(l)}|, \tag{6}$$

where $\mathbf{B}_k^{(l)}$ denotes the LSBs of the weights for layer l, and $|\cdot|$ represents the absolute value function. The gradient of the regularization loss with respect to $\mathbf{W}$ is given by:

$$\frac{\partial \mathrm{R}(\mathbf{B_k})}{\partial \mathbf{W}} = \mathrm{sign}(\mathbf{B_k}), \tag{7}$$

which will guide the weight towards the nearest low-precision value following the RoundClamp quantizer design.

The overall quantization-aware training objective with the ℓ_1 regularization is defined as:

$$\min_{\mathbf{W}} \ \mathcal{L}(\mathbf{W}_n) + \lambda \sum_{\forall l} |\mathbf{B}_k^{(l)}|, \tag{8}$$

where $\mathcal{L}(\cdot)$ is the original training loss, $\mathbf{W}_n$ is the n-bit quantized weight with the RoundClamp quantizer, and λ is the regularization strength trading off model compression rate and model performance. The objective can be optimized via gradient-based optimizer with the help of the straight-through estimator.

3.2. Hessian-aware Aggressive Pruning

The bipartite bit slicing introduces a new hyperparameter, k, to the MSQ process. As the MSQ training objective in Eq. (8) aims to sparsify the k LSBs in each layer, k decides each layer's precision reduction speed. Intuitively, if

(a) Omega values during the first pruning step.

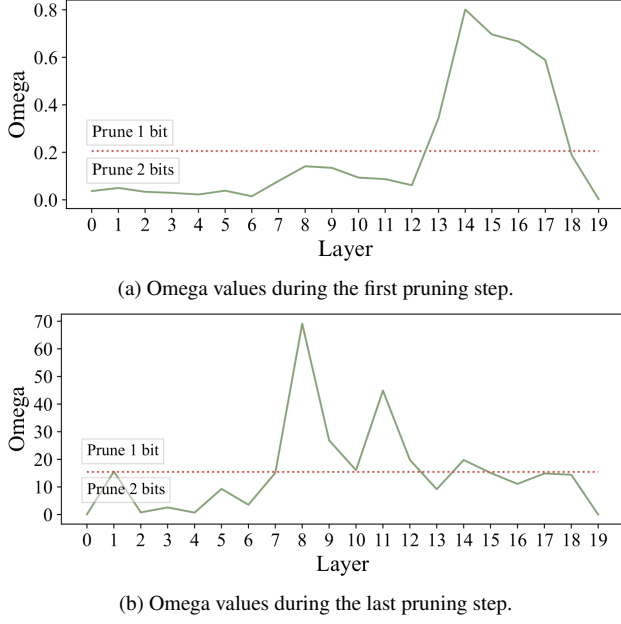(b) Omega values during the last pruning step.

Figure 5. Comparison of Omega values for each layer of ResNet-20 between the first and last pruning steps. Layers with omega values exceeding the average (red dotted line) are pruned by 1 bit, while those below are pruned by 2 bits.

a layer's weight can tolerate more perturbations without impacting the final model loss, i.e. less sensitive, it's precision can be reduced at a higher pace. On the other hand, precision reduction on a sensitive layer should be cautious to avoid catastrophically hurting the model performance.

To incorporate sensitivity information into the pruning process, we propose Hessian-aware Aggressive Pruning, where we measure the sensitivity of each layer using their Hessian statistics and assign a larger k to those layers that are less sensitive. Specifically, we follow the methodology proposed in HAWQ-V2 [6], which identifies the Hessian trace can be used as a reliable sensitivity metric for quantized models. The sensitivity of each layer, Ω_l, is calculated as:

$$\Omega_l = Tr(H_l)\|\mathbf{W}_n^{(l)} - \mathbf{W}^{(l)}\|^2, \qquad (9)$$

where l refers to the lth layer, $Tr(H_i)$ denotes Hessian trace with respect to the lth layer weight, $\mathbf{W}_n$ represents the quantized weight under the current precision, and $\|\cdot\|$ the L_2 norm.

MSQ uses a heuristic-based thresholding method to decide the precision reduction speed k for each layer. We set the threshold to be the averaged sensitivity of all layers. If the sensitivity of a layer is larger than the threshold, we set $k = 1$, where as layers with sensitivity lower than the threshold use $k = 2$, as shown in Fig. 5. Additionally, as the training and LSB pruning step progresses, the sensitivity metric Ω_l for each layer will be recomputed

---

Algorithm 1 Overall training algorithm

Input: Data $\mathbf{X}$, label $\mathbf{Y}$
Output: mixed-precision model G

1: Initialize: p, q in G
2: Initialize: regularization strength λ, pruning interval I, pruning threshold α, target compression Γ
3: Initialize: LSB-nonzero rate β, compression γ
4: **for** epoch = 1, ..., T **do**
5:     **for** batch from $\mathbf{X}$, $\mathbf{Y}$ **do**
6:         Compute forward pass of G
7:         *# LSB L_1 regularization*
8:         Update parameters with $L = L_{ce} + \lambda\mathrm{R}(\mathbf{B})$
9:     **end for**
10:     **if** epoch % I == 0 **and** $\gamma > \Gamma$ **then**
11:         Calculate Hessian trace $Tr(H)$
12:         Initialize Ω
13:         *# Omega and LSB-nonzero rate calculation*
14:         **for** Quantized layer l in G **do**
15:             $\Omega_l \leftarrow Tr(H_l)\|\mathbf{W}_{q_l}^{(l)} - \mathbf{W}^{(l)}\|^2$
16:             $\beta_l \leftarrow \mathrm{sum}(\mathbf{B} > 2^{p_l - 1})$
17:         **end for**
18:         *# Pruning*
19:         Ascending sort β
20:         **for** Quantized layer l in G **do**
21:             **if** $\beta_l < \alpha$ **then**
22:                 $q_l = q_l - p_l$
23:             **end if**
24:             **if** $\gamma > \Gamma$ **then**
25:                 break
26:             **end if**
27:         **end for**
28:         *# Hessian aware changing pruning bit*
29:         **for** Quantized layer l in G **do**
30:             **if** $\Omega_l < mean(\Omega)$ **then**
31:                 $p_l = 2$
32:             **else**
33:                 $p_l = 1$
34:             **end if**
35:         **end for**
36:     **end if**
37: **end for**

---

to capture its change, which can be observed in the transition from Fig. 5a to Fig. 5b. This adaptive strategy allows for accelerated training by pruning multiple bits at once in low-sensitivity layers while preserving accuracy in high-sensitivity layers. Moreover, it ensures that Hessian information is effectively utilized to optimize the pruning process, resulting in both computational efficiency and improved performance in quantized networks.
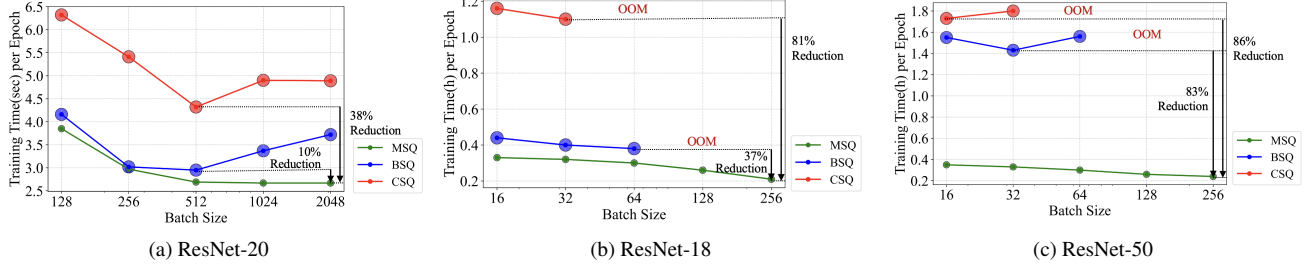
Figure 6. Comparison of training time per epoch for different methods across varying batch sizes. Results are reported up to the maximum batch size that does not cause out-of-memory (OOM). Circle sizes represent the number of trainable parameters.

3.3. Overall Training Algorithm

The complete training workflow is summarized in Algorithm 1. The model is trained using the objective defined in Eq. (8), enabling simultaneous weight optimization and LSB sparsification through L_1 regularization. At every pruning interval, the sensitivity metric Ω is calculated for all layers, guiding the adjustment of the pruning bit amount p for the subsequent pruning step.

The pruning strength is controlled by two key parameters: the regularization strength λ and the pruning threshold α, which are applied uniformly across all layers. Pruning occurs when the LSB nonzero rate β for a given layer falls below the pruning threshold α. The pruning continues until the target compression ratio Γ is achieved. In the final round of pruning, we sort β to prioritize pruning layers with the lowest LSB nonzero rates until the target model compression ratio is reached. Once the Γ is reached, regularization and pruning are completely stopped, and training continues as a standard QAT process to improve model performance under the finalized quantization scheme.

4. Evaluation

We evaluate MSQ on a variety of neural network architectures to demonstrate its generality and effectiveness. For convolutional networks, we test MSQ on ResNet-20 [9] using the CIFAR-10 dataset [13], and on ResNet-18 and ResNet-50 [9] using the ImageNet dataset [4]. To further assess generalization to heterogeneous CNN architectures, we also evaluate MSQ on MobileNetV3-Large [11], which include depthwise convolutions and squeeze-and-excitation blocks. For transformer-based architectures, we assess MSQ on ImageNet using DeiT [22] and Swin-T [17], representing both lightweight and hierarchical Vision Transformers (ViTs). For CNNs, we compare MSQ with existing uniform quantization methods [3, 31] as well as mixed-precision quantization approaches [6, 26, 27, 29]. For ViTs, we evaluate MSQ against various ViT-specific quantization techniques [7, 14, 19, 22].

4.1. Experimental Setup

We employ a consistent set of hyperparameters for experiments conducted on the same model. Models trained on CIFAR-10 use the SGD optimizer with an initial learning rate of 0.1 and a warm-start cosine annealing scheduler. For ImageNet, we use the SGD optimizer with an initial learning rate of 0.01, also adopting a warm-start cosine annealing schedule. Experiments on CIFAR-10 are trained from scratch and for 400 epochs, whereas on ImageNet, we use floating-point pretrained models and train for 100 epochs. These training durations are comparable to the total epochs used in prior methods, such as BSQ [27] and CSQ [26]. For activation quantization, we use uniform quantization and report the bit precision in the "A-Bits" column. In ImageNet experiments, all activations remain at full precision, while in ViT experiments, activations are quantized to 8-bit.

4.2. Experimental Results

We compare MSQ with previous baseline quantization methods. In all tables, "FP" refers to the full-precision model, and "MP" denotes mixed-precision weight quantization. The weight compression ratio, "Comp" is calculated relative to the full-precision model. The target compression ratios of 16.00 and 10.67 correspond approximately to average weight bit-widths of 2 and 3 bits, respectively.

Table 1. Training resource usage for each quantization method.

Network		BSQ	CSQ	**MSQ**
ResNet-20	Epochs	350	600	400
	Train Batch Size	256	256	1024
	Total Time (h)	0.36	0.90	**0.33**
	Params (M)	2.16	2.16	**0.27**
	Peak Memory(GB)	2.21	2.40	**2.77**
ResNet-18	Epochs	90	200	100
	Train Batch Size	64	32	256
	Total Time (h)	63.92	220.32	**21.11**
	Params (M)	93.52	93.52	**11.69**
	Peak Memory(GB)	10.13	10.41	**11.13**
ResNet-50	Epochs	90	200	100
	Train Batch Size	32	16	256
	Total Time (h)	128.76	346.12	**24.30**
	Params (M)	204.8	204.8	**25.6**
	Peak Memory(GB)	11.54	12.82	**12.22**

Table 2. Quantization results of ResNet-20 models on the CIFAR-10 dataset.

A-Bits		32			3			2	
Method	W-Bit	Comp	Acc	W-Bit	Comp	Acc	W-Bit	Comp	Acc
FP	32	1.00	92.62	–	–	–	–	–	–
LQ-Nets	3	10.67	92.00	3	10.67	91.60	2	16.00	90.20
PACT	–	–	–	3	10.67	91.60	2	16.00	89.70
DoReFa	–	–	–	3	10.67	89.90	2	16.00	88.20
BSQ	MP	19.24	91.87	MP	11.04	92.16	MP	18.85	90.19
CSQ T2	MP	16.00	92.68	MP	16.93	92.14	MP	16.41	90.33
MSQ	MP	**16.13**	**92.17**	MP	**17.43**	**92.00**	MP	**19.13**	**90.22**
MSQ	MP	**20.13**	**92.15**	–	–	–	MP	**16.43**	**90.60**

Table 3. Quantization results of ResNet-18 and ResNet-50 models on the ImageNet dataset.

	ResNet-18			ResNet-50		
Method	W-Bits	Comp(x)	Acc(%)	W-Bits	Comp(x)	Acc(%)
FP	32	1.00	69.76	32	1.00	76.13
DoReFa [31]	4	6.40	68.4	3	10.67	69.90
PACT [3]	4	8.00	69.2	3	10.67	75.30
LQ-Nets [30]	3	10.67	69.30	3	10.67	74.20
HAWQ-V3 [29]	4	8.00	68.45	4	8.00	74.24
HAQ [23]	-	-	-	MP	10.57	75.30
BSQ [27]	-	-	-	MP	13.90	75.16
CSQ T3 [26]	MP	10.67	69.73	MP	10.67	75.47
MSQ	**MP**	**11.84**	**69.74**	**MP**	**10.89**	**75.32**

Table 4. Comparison of quantization methods on DeiT-T/S and Swin-T with compression ratio and Top-1 accuracy on ImageNet.

		DeiT-T		DeiT-S		Swin-T	
Method	W-Bits	Comp(×)	Acc(%)	Comp(×)	Acc(%)	Comp(×)	Acc(%)
LSQ	3	10.67	68.09	10.67	77.76	10.67	78.96
Mix-QViT	3	10.67	69.62	10.67	78.08	10.67	79.45
QViT	3	10.67	67.12	10.67	78.45	10.67	80.06
OFQ	3	10.67	72.72	10.67	79.57	10.67	79.57
OFQ	4	8.00	75.46	8.00	81.10	8.00	81.88
MSQ	MP	**10.54**	**74.74**	**9.58**	**80.64**	**9.14**	**81.38**

MSQ even improves accuracy over CSQ. On ImageNet (Table 3), MSQ outperforms prior methods on ResNet-18 and achieves comparable performance to CSQ on ResNet-50.

MobileNetV3 Results. To assess the generalization capability of MSQ, we evaluate it on MobileNetV3-Large using the ImageNet dataset. This architecture includes diverse components such as depthwise separable convolutions and squeeze-and-excitation (SE) blocks, which present additional challenges for quantization. As shown in Table 5, MSQ achieves higher compression ratios compared to prior methods while preserving model accuracy.

ViT Results. Table 4 presents the MSQ results for Vision Transformer models on ImageNet datasets, including DeiT [22] and Swin [17]. In these experiments, we use the 4-bit quantized checkpoint from the OFQ [15] repository as the pretrained model and fine-tune it using using MSQ. MSQ achieves a better efficiency-accuracy trade-off compared to previous methods, delivering comparable performance at higher compression ratios. These results demonstrate that MSQ is effective for transformer-based models, extending its applicability beyond CNN architectures.

4.3. Ablation Study

In this section, we delve into the critical design aspects of the MSQ, particularly examining the influence of the Hessian information in determining the bit scheme for mixed-precision quantization. The experiments used in this section are performed using ResNet-20 models [9], leveraging 3-bit activation quantization, and are evaluated on the CIFAR-10 dataset [13].

Training Efficiency Comparison. As shown in Fig.6, This comparison presents the training time for each batch size across different methods (BSQ, CSQ, and MSQ). The batch size for each method was increased until it reached the out-of-memory limit. The specific details are summarized in Table 1, where the training resource usage is reported. MSQ achieves up to 8× fewer trainable parameters compared to BSQ and CSQ, primarily due to its structural design that avoids explicit bit-level splitting during training. Consequently, MSQ allows for larger batch sizes before reaching the out-of-memory limit, ultimately reducing training time. To ensure a fair comparison, we also measured the total training time using batch sizes adjusted to yield similar peak memory usage for each method on an RTX GeForce 4080 Super. As the number of parameters increased, MSQ's efficiency advantage became more significant. In particular, on ResNet-50, MSQ achieved a 5.3× speedup over BSQ and a 14.2× speedup over CSQ. This acceleration stems from MSQ's architectural efficiency, which enhances parallelism and minimizes computational overhead by eliminating bit-level splitting during training.

ResNet Results. We evaluate MSQ on ResNet-20 with CIFAR-10, while ImageNet is used for ResNet-18 and ResNet-50. As shown in Table 2, MSQ achieves higher compression ratios than CSQ under 32-bit and 3-bit activation settings, with minimal accuracy drop. In the 2-bit case,

Table 5. Quantization results of MobileNetV3-L on the ImageNet dataset.

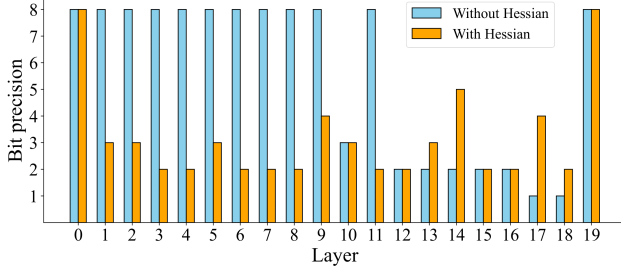| Method	W-Bits	Comp(×)	Acc(%)
FP	32	1.00	75.27
DoReFa [31]	8	4.00	74.44
MSQ	MP	**5.36**	**74.29**
DoReFa [31]	4	8.00	72.92
MSQ	MP	**10.30**	**73.58**

Figure 7. Final bit scheme selected during the training process. The bit scheme is determined at epoch 210 for the case without Hessian (accuracy: 91.23%), and at epoch 150 for the case with Hessian (accuracy: 91.93%).
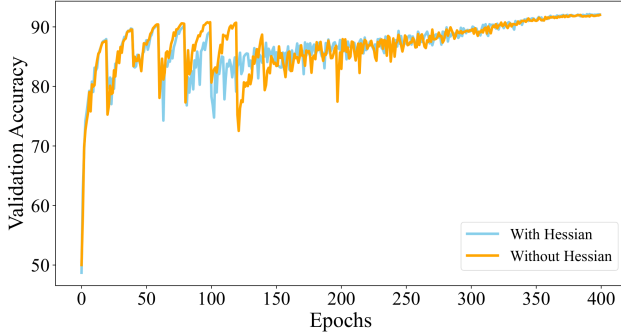

Figure 8. The change of validation accuracy with and without Hessian during training. Accuracy drop is observed when LSB pruning happens.

Effectiveness of Hessian. As discussed in Sec. 3.2, MSQ leverages Hessian information to account for layer sensitivity, enhancing the efficiency of the bit-pruning process. Fig. 7 compares the final bit schemes after training with and without Hessian guidance. Without the Hessian guidance, some layers are constantly pruned while others are less changed, leading to the need of using more pruning epochs to reach the target compression rate and poorer final accuracy. On the other hand, the proposed Hessian-aware aggressive pruning scheme reduces the pruning speed for overly-pruned layers, which becomes more sensitive as they are pruned, while promoting faster pruning of other less sensitive layers. This leads to a more well-behaving quantization scheme. As we achieve faster pruning speed with the Hessian guidance, the accuracy drop suffered by the model from each pruning step is smaller as shown in Fig. 8, which helps achieving a better compression-accuracy tradeoff with less training.

Quantization Scheme Comparison with BSQ. We showcase the quantization scheme achieved by MSQ and BSQ on the ResNet-20 model in Fig. 9. We noticed that the sparsity observed by BSQ is more concentrated on a few layers, whereas that achieved by MSQ appears more even. We believe that this difference is induced by the overly aggressive bit-level training and regularization introduced by
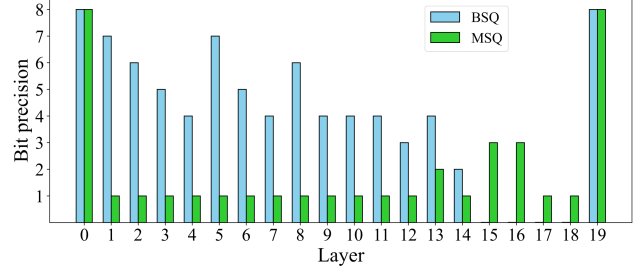

Figure 9. Comparison of Final Bit Schemes Between MSQ and BSQ During Training: The BSQ's compression ratio is 19.24x with accuracy 91.87%, while MSQ's compression ratio is 20.13x with accuracy 92.15%.

BSQ, where each bit is trained and regularized separately. In BSQ, strong regularization may reduce some layers to 0-bit precision, effectively removing all weights and allowing the layer to be skipped. MSQ, on the other hand, single out only the LSBs with the guidance of Hessian information, which leads to a better quantization scheme with improved accuracy and compression rate.

5. Conclusion

In this work, we introduce MSQ, a novel memory-efficient approach for bit-level mixed-precision quantization DNN training. MSQ computes and regularizes LSBs directly from the trainable parameters, leading to bit-level sparsity and precision reduction without explicit bit splitting. Hessian information is incorporated to guide proper precision reduction rate across the layers. Experimental results demonstrate the effectiveness of MSQ, achieving accuracy on par with existing mixed-precision methods while significantly reducing computational resources, including training time and GPU memory usage. This efficiency makes it feasible to train large-scale networks, such as Vision Transformers (ViTs), using a bit-level training approach.

Acknowledgments

References

[1] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019. 1

[2] Alessio Burrello, Angelo Garofalo, Nazareno Bruschi, Giuseppe Tagliavini, Davide Rossi, and Francesco Conti. Dory: Automatic end-to-end deployment of real-world dnns on low-cost iot mcus. *IEEE Transactions on Computers*, 70 (8):1253–1268, 2021. 1

[3] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 1, 2, 6, 7

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[5] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 293–302, 2019. 1, 2

[6] Zhen Dong, Zhewei Yao, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *Advances in neural information processing systems*, 33:18518–18529, 2020. 1, 2, 5, 6

[7] Steven K. Esser, John L. McKinstry, Dhruv Bablani, Raja Appuswamy, and Dharmendra S. Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 6

[8] Cong Guo, Yuxian Qiu, Jingwen Leng, Xiaotian Gao, Chen Zhang, Yunxin Liu, Fan Yang, Yuhao Zhu, and Minyi Guo. Squant: On-the-fly data-free quantization via diagonal hessian approximation. *arXiv preprint arXiv:2202.07471*, 2022. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6, 7

[10] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, pages 10–14. IEEE, 2014. 1

[11] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *arXiv preprint arXiv:1905.02244*, 2019. 6

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 1

[13] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6, 7

[14] Yanjing Li, Shuang Xu, Bowen Zhang, Xiangtai Cao, Peng Gao, and Guodong Guo. Q-vit: Accurate and fully quantized low-bit vision transformer. In *Advances in Neural Information Processing Systems*, pages 34451–34463, 2022. 6

[15] Shih-Yang Liu, Zechun Liu, and Kwang-Ting Cheng. Oscillation-free quantization for low-bit vision transformers. In *International Conference on Machine Learning*, pages 21813–21824. PMLR, 2023. 7

[16] Yijiang Liu, Huanrui Yang, Zhen Dong, Kurt Keutzer, Li Du, and Shanghang Zhang. Noisyquant: Noisy bias-enhanced post-training activation quantization for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20321–20330, 2023. 2

[17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 6, 7

[18] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International conference on machine learning*, pages 7197–7206. PMLR, 2020. 2

[19] Andreas Savakis Navin Ranjan. Mix-qvit: Mixed-precision vision transformer quantization driven by layer importance and quantization sensitivity. *arXiv preprint arXiv:2501.06357*, 2025. 6

[20] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1

[21] Pedro Savarese, Hugo Silva, and Michael Maire. Winning the lottery with continuous sparsification. *Advances in neural information processing systems*, 33:11380–11390, 2020. 2

[22] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 6, 7

[23] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8612–8620, 2019. 1, 2, 7

[24] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10734–10742, 2019. 1

[25] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023. 2

[26] Lirui Xiao, Huanrui Yang, Zhen Dong, Kurt Keutzer, Li Du, and Shanghang Zhang. Csq: Growing mixed-precision

quantization scheme with bi-level continuous sparsification. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023. 1, 2, 3, 6, 7

[27] Huanrui Yang, Lin Duan, Yiran Chen, and Hai Li. Bsq: Exploring bit-level sparsity for mixed-precision neural network quantization. *arXiv preprint arXiv:2102.10462*, 2021. 1, 2, 3, 6, 7

[28] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7308–7316, 2019. 1

[29] Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, et al. Hawq-v3: Dyadic neural network quantization. In *International Conference on Machine Learning*, pages 11875–11886. PMLR, 2021. 6, 7

[30] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018. 1, 7

[31] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 1, 2, 3, 4, 6, 7