Stability-Driven Quantization-Aware Training for Low-Bit Models

Du Tran-Ngoc
Viettel Semiconductor Center
Hanoi, Vietnam
dutn1@viettel.com.vn

Quang Le-Hoang-Minh
Viettel Semiconductor Center
Hanoi, Vietnam
quanglhm@viettel.com.vn

Thang Nguyen-Minh*
Viettel Semiconductor Center
Hanoi, Vietnam
thangnm35@viettel.com.vn

Trung Dong*
Viettel Semiconductor Center
Hanoi, Vietnam
trungdq8@viettel.com.vn

***Abstract*—Quantization-Aware Training (QAT) has gained significant attention thanks to reduced memory usage and accelerated inference, which are achieved by transforming full-precision models to low-bit integer formats while preserving accuracy. However, applying QAT to ultra-low precision (e.g., 4 bits or 3 bits) can lead to severe accuracy degradation due to the improper weights and quantization parameters during training. In this paper, we present a comprehensive analysis of the root causes of accuracy degradation in low-bit QAT. Based on the analysis, we propose three novel techniques — Gradient Scaling-Aware Distance, EMA-Aware Distance, and Adaptive Fine-Tuning Weight that effectively mitigate quantization noise without adding inference overhead. Experimental results demonstrate that our method outperforms existing QAT approaches. In 4-bit quantization, our method achieves accuracy drop within 0.7% of the full-precision model on the ImageNet-1k classification task and 1.8% on MS COCO object detection task, while reducing model size by 87.5%.**

***Index Terms*—Quantization-aware training, Oscillation, Deep Neural Network**

I. Introduction

In recent years, the increasing demand to deploy deep neural networks (DNNs) on resource-constrained devices has gained attention into model compression techniques [1], [2]. Quantization, which reduces the precision of model weights and activations from full precision to fixed-point representations, has proven to be an effective approach for reducing memory usage and computational cost with minimal degradation in accuracy. Depending on the implementation strategy, quantization methods are categorized into post-training quantization (PTQ) and quantization-aware training (QAT). PTQ compresses a pretrained model from 32-bit floating point to low-bit representation without retraining, only requires a small calibration dataset to determine quantization parameters. However, in complex tasks such as image classification or object detection, PTQ often suffers from significant accuracy performance degradation.

To overcome the limitations of PTQ, QAT introduces virtual quantization layers during training, enabling the model to adapt to quantization-induced rounding errors. This approach employs the Straight-Through Estimator (STE) [3] to approximate the gradients of non-differentiable rounding operations, allowing end-to-end backpropagation. While QAT outperforms PTQ, it still suffers from a noticeable performance drop when applied to ultra low-bit settings, e.g. 3 bits, 4 bits. Recent studies [4], [5] highlight that using STE may face unstable optimization due to oscillation during training. When using STE, the weights fluctuate between neighboring quantization levels, resulting in optimization noise throughout the training process. Some works [6], [7] propose freeze-based solutions to reduce the oscillation during training. The authors of [8] [9] propose smooth approximations to replace STE for stable training. The work [5] introduces a regularized loss for controlling the weights in the non-oscillating state. In fact, the gradient of parameters has the potential to cause instability during training, leading to a drop in QAT accuracy. However, the analysis of its properties remains underexplored.

In this paper, we investigate the root causes of accuracy degradation in QAT using theoretical and empirical analysis. Our analysis highlights that unstable gradient of quantization parameters is a primary cause of training oscillations and accuracy degradation in ultra low-bit settings. Furthermore, we show that using the STE approximation in computing the gradients of the weights not only causes oscillations, but also leads to non-uniform updates. In particular, weights close to the threshold are more susceptible to update gradients to optimize target loss than the ones far away in the same quantization step. This problem hinders convergence to a better local minima. To this end, we propose three algorithms to improve the accuracy of QAT: Gradient Scaling Distance-Aware (ScaleGrad), EMA-Aware Distance for updating weights (EAD), and Adaptive Finetuning Weight (AFW).

The contributions of this paper are summarized as follows:

- We provide theoretical and empirical analyses for quantization-aware training.
- We propose ScaleGrad and EAD to address the problem of instability in the QAT training process.
- To fine-tune the oscillating weights during training, we propose a post-processing method called AFW.
- Extensive experiments demonstrate that our approach achieves outstanding performance, outperforming existing QAT methods in ultra low-bit quantization settings.

II. Related Work

A. Quantization Aware Training

Quantization-aware training has been extensively studied for its ability to minimize accuracy degradation when converting

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

models to low-precision formats by incorporating quantization into the training process. Techniques such as Element-Wise Gradient Scaling (EWGS) [10] and Learned Step Size Quantization (LSQ) [11] propose to enhance training stability and mitigate the impact of quantization. QAT typically relies on STE to facilitate gradient-based optimization due to the non-differentiability of quantization operations. However, STE approximates the gradient of the rounding function to 1 at every input value, which causes suboptimality. To address its limitations, alternative approaches such as mirror descent and gradient bias correction in [10], smooth approximations [8], [9] have been introduced. Nevertheless, in the ultra lower-bit settings on complicated datasets, these methods suffer from a drop in accuracy compared with the full precision model.

B. Oscillations in QAT

One critical challenge of STE-based QAT is the oscillation in weight. Recent studies [5], [12] show that the approximation error introduced by STE can lead to instability during training. To address the issue, the approach in [12] replaces the quantization operation with an additive Gaussian noise to mimic quantization effects while avoiding oscillatory behavior. Similarly, the authors of [5] find that oscillations worsen in lightweight architectures employing depth-wise convolutions. They propose to constrain the latent weights either by regularizing them toward quantized values or by freezing them entirely. More recently, the work [7] investigates the oscillation in vision transformers and introduces fixed scaling factors along with a reparameterization of the query-key mechanism to mitigate these effects. Although QAT improves accuracy under low-bit settings, the causes of oscillation remain insufficiently analyzed.

III. Analysis of Oscillation in Quantization-aware Training

This section investigates oscillations in QAT through theoretical analysis and empirical observations.

A. Quantization Aware Training Analysis

We formulate the Quantization function with latent value x - input and output $\hat{x}$ as follows:

$$\bar{x} = clip\left(\left\lceil \frac{x}{s} \right\rfloor, \alpha, \beta\right) \tag{1}$$

$$\hat{x} = s \times \bar{x} \tag{2}$$

where s is the scaling factor, α and β denote the lower and upper bounds based on bitwidth, respectively, $\lceil \cdot \rfloor$ denotes the rounding-to-nearest operation whose derivative is approximated by 1 in the backward pass by STE, with the following formula:

$$\frac{\partial L}{\partial x} = \begin{cases} \frac{\partial L}{\partial \hat{x}} \text{ if } \frac{x}{s} \in [\alpha, \beta] \\ 0 \end{cases} \tag{3}$$

where L is the target loss. During training, weights and activations are quantized in the forward pass by (1), considering the effects of quantization errors. Scaling factor s can be either precomputed or jointly optimized with the model parameters [11] [13]. Unfortunately, STE degrades the accuracy of QAT at lower-bit formats.

Next, we mathematically analyze the high probability causes of the degradation and propose methods to improve performance in the following sections. Firstly, we analyze the gradient concerning the scaling factor of the loss function, as outlined below:

$$\frac{\partial L}{\partial s} = \sum_{i}^{K} \frac{\partial L}{\partial \hat{x}_i} \frac{\partial \hat{x}_i}{\partial s} \tag{4}$$

with:

$$\frac{\partial \hat{x}_i}{\partial s} = \begin{cases} \alpha \text{ if } \frac{x_i}{s} \leq \alpha \\ \left(\left\lceil \frac{x_i}{s} \right\rfloor - \frac{x_i}{s}\right) \text{ if } \frac{x_i}{s} \in [\alpha, \beta] \\ \beta \text{ if } \frac{x_i}{s} \geq \beta \end{cases} \tag{5}$$

From (5), it can be seen that term $\left(\left\lceil \frac{x_i}{s} \right\rfloor - \frac{x_i}{s}\right)$ is a discontinuous function. Specifically, when $\frac{x_i}{s}$ comes close to $k+0.5$, which is a transition point, the term experiences a surge with maximum amplitude.

For example, when $\frac{\partial L}{\partial \hat{x}^t} > 0$ and approximately constant with value ε, and $x < 0$ over an interval of T_0 iterations $[t, t+T_0]$. At iteration t, assume that $\frac{x_i^t}{s^t}$ is approaching the left side of a transition point, i.e.,

$$\frac{x_i^t}{s^t} \to k + 0.5^- \tag{6}$$

In this situation, the gradient with respect to the scaling factor becomes $\frac{\partial L}{\partial s^t} \approx -0.5\varepsilon < 0$

. After the gradient descent update, the new value s^{t+1} is: $s^{t+1} = s^t - lr.\frac{\partial L}{\partial s^t} > s^t$. So, at iteration $t+1$, we have given as:

$$\frac{x_i^t}{s^t} < \frac{x_i^{t+1}}{s^{t+1}} \to k + 0.5^+ \tag{7}$$

$$\frac{\partial L}{\partial s^{t+1}} \approx 0.5\varepsilon > 0 \tag{8}$$

Here, $\to$ indicates approaching a value, while $k+0.5^+$ and $k+0.5^-$ denote right-sided and left-sided neighborhoods of $k+0.5$. Equations (6), (7), and (8) show that the gradient of loss with respect to the scaling factor can fluctuate during training. Since x_i includes both weights and activations, such instability leads to oscillations, thus degrading model performance.

The latent weight oscillation during QAT is also determined from the nature of its gradients. Due to the use of the STE approximation, the gradient depends on the occurrence of a quantization level transition rather than the latent weight's value. Indeed as shown in (3), the gradient of the latent weight depends only on the gradient of the quantize weight, ignoring the gradient component of the quantization function.

So, when $\left|\left\lceil \frac{x_i}{s} \right\rfloor - \frac{x_i}{s}\right|$ is small, gradient updates may not be large enough to change the latent weight. Thus, such updates fail to reduce total loss and optimize the model parameters effectively.

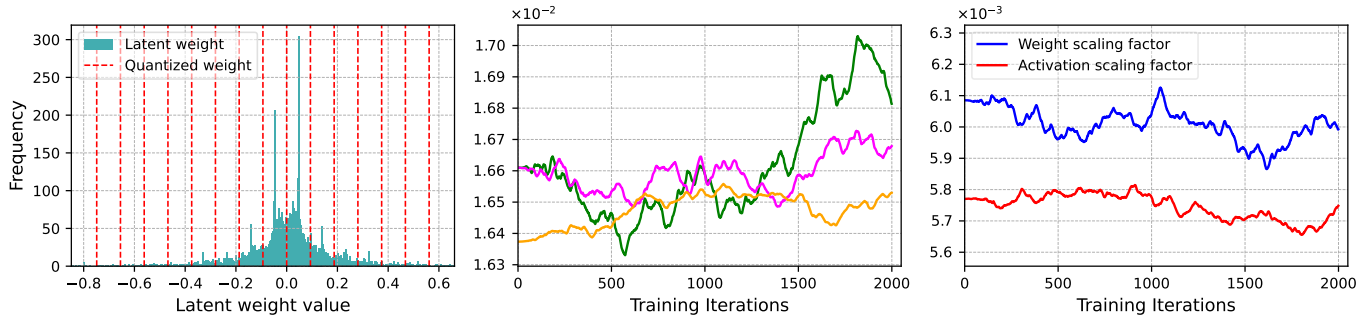

Fig. 1: The oscillation of the weights of the YOLOv8-n model in the last 2000 iterations: (a) latent weight histogram on the 2^{nd} layer, (b) weight fluctuations in the 2^{nd} layer's first channel, and (c) scaling factor dynamics in the 3^{rd} layer.

TABLE I: YOLOv8-n mAP performance under LSQ training and training with frozen weights and learnable scaling factors.

Method		YOLOv8-n
Origin Model	32bit	37.3
Baseline	LSQ	32.1
Freeze	Freeze weight	31.7

B. Observing Oscillations in QAT

In this section, we examine the oscillation during quantization-aware training of DNNs using YOLOv8n model [14]. We perform QAT on YOLOv8n using learnable scaling factors, following [11], until the model's mAP shows minimal further improvement.

Fig. 1(a) shows the histogram of latent weights in the 2nd layer. A large proportion of weights are concentrated near the quantization thresholds, which indicates a high probability of oscillations between adjacent quantization levels. This phenomenon is further illustrated in Fig. 1(b), where the latent weights have chaotic oscillation.

To observe the oscillation of the scaling factor, the model weights are frozen after training, and the scaling factors are updated via the backpropagation algorithm.

Fig. 1(c) shows that, despite a small learning rate, the scaling factor of weight and activation fluctuates around the initial value. Such an instability is more severe in early layers, where gradient fluctuations accumulate due to the chain rule. The results in Table I show a noticeable drop in the model's mAP compared to the baseline, emphasizing the issue.

IV. Methodology

This section presents proposed methods to reduce oscillations and improve QAT stability. The first two methods are applied during training, and the third one is a post-processing step for fine-tuning performance.

A. Gradient Scaling Distance-aware (ScaleGrad)

We propose a function to solve the discontinuous behavior as discussed in Sec.III-A, called *PulseSmooth*, as depicted in Fig. 2. *PulseSmooth* allows for smoother gradient transitions.

We mathematically model the proposed method as follows. Let $X \in \mathbb{R}^*$ denote the latent weight or activation tensor, and

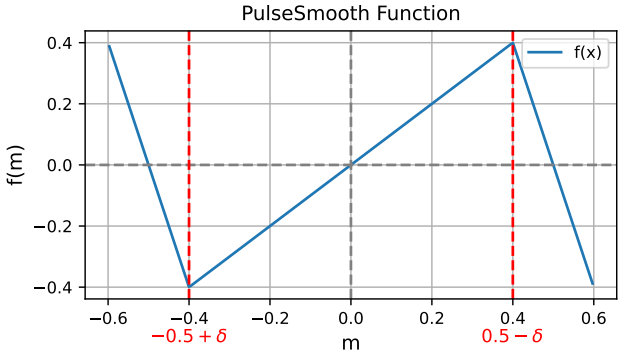

Fig. 2: The proposed PulseSmooth function, the x-axis represents the value of m in the range from -0.5 to 0.5. Note that when m is in the range (-0.5+δ, 0.5-δ) the PulseSmooth function is an identity function.

$s \in \mathbb{R}$ be the scaling factor. We define that $m = \left(\left\lceil \frac{X}{s} \right\rfloor - \frac{X}{s}\right)$ and $m \in (-0.5, 0.5)$, and let δ be the continuous control parameter. *PulseSmooth* function, denoted as F_{ps}, is formulated as follows:

$$F_{ps}(m) = \begin{cases} -\dfrac{0.5-\delta}{\delta}(m+0.5) & \text{if } m \in (-0.5, -0.5+\delta] \\ m & \text{if } m \in [-0.5+\delta, 0.5-\delta] \\ -\dfrac{0.5-\delta}{\delta}(m-0.5) & \text{if } m \in [0.5-\delta, 0.5) \end{cases}$$

Our idea is that m has a smoother transition using δ. When m approaches $0.5-\delta$ or $-0.5+\delta$, it has a gradual change in amplitude, while the sign remains unchanged. Therefore, the PulseSmooth function reduces the abrupt changes in the gradient of the scaling factor s, while still preserving the directional characteristics of the original discontinuous function (5). The gradient for the loss function is given by:

$$\frac{\partial L}{\partial s} = \sum_{i}^{N_x} \frac{\partial L}{\partial \hat{x}_i} F_{ps}\left(\frac{\partial \hat{x}_i}{\partial s}\right).$$

We note that, when the gradient force is weak, the latent weight stays unchanged at a quantization level. So, we propose a method of scaling the gradients of the latent weight accord-

ing to the distance-aware factor for more stable convergence. The proposed scaling function is given as:

$$\begin{aligned} \text{Fscale}(w) &= e^{-k\cdot\Delta} \\ \Delta &= \left| \left\lceil \frac{w}{s} \right\rfloor - \frac{w}{s} \right| \in [0, 0.5) \end{aligned} \tag{9}$$

where k controls the scaling function. The gradient of the latent weight after scaling is as follows:

$$\frac{\partial L}{\partial w} = \begin{cases} \frac{\partial L}{\partial \hat{w}} \cdot \text{Fscale(w)} \text{ if } \frac{w}{s} \in [\alpha, \beta] \\ 0 \end{cases} \tag{10}$$

Equations (9) and (10) show that when a latent weight is far from the quantization threshold, Fscale has a large value to push the update process. Meanwhile, for the latent weights near the quantization threshold, a smaller value of Fscale does not affect their update process; it is non-trivial to switch to a new state to change the overall loss. As a result, this leads to a more balanced and stable learning.

B. EMA-Aware Distance for updating weight (EAD)

To further mitigate the oscillations of weights and scaling factors during the QAT process, we propose a distance-aware Exponential Moving Average (EMA) method. As the training process approaches convergence, weights with quantized values far from the latent weight highly oscillate between two adjacent quantization levels. Therefore, those weights can have a larger EMA smoothing factor to stabilize the update process. In contrast, weights far from the quantization threshold have a lower chance of changing quantization levels. So, a smaller smoothing factor can use the latest updates from target loss, which benefits overall performance. The update formula for the latent weight W of l^{th} layer at t^{th} iteration is as follows:

$$\begin{aligned} W_t^l &= \alpha W_{t-1}^l + (1-\alpha) W_t^l \\ \alpha &= \begin{cases} k \cdot |W_t^l - \hat{W}_t^l| & \text{if } t > \tau \\ 0 & \text{else} \end{cases} \end{aligned} \tag{11}$$

The the scaling factor of activation s_a and weight s_W in the l^{th} layer is updated during training is as follows:

$$\begin{aligned} s_{W(t)}^l &= (1-\mu) s_{W(t-1)}^l + \mu s_{W(t)}^l \\ s_{a(t)}^l &= (1-\mu) s_{a(t-1)}^l + \mu s_{a(t)}^l \\ \mu &= (1 - \frac{t-\tau}{epoch-\tau}) \mu_{init} \end{aligned} \tag{12}$$

Where τ and k are hyperparameters representing the iteration milestone at which the EMA update process is activated and the corresponding scale factor, respectively. μ_{init} is initial smoothing coefficient used for updating of the scaling factor.

C. Adaptive Finetuning Weight (AFW)

In this section, we propose a post-processing for the quantized model to fine-tune weights that exhibit strong oscillations after training, aiming to help the model reach a better performance. To identify which weights should be adjusted during this stage, we track their state and sign transitions throughout training using the frequency tracking metric [5]. Weights that switch between adjacent quantization levels are frozen during training and later fine-tuned in a post-processing step. We formulate the following optimization problem as follows:

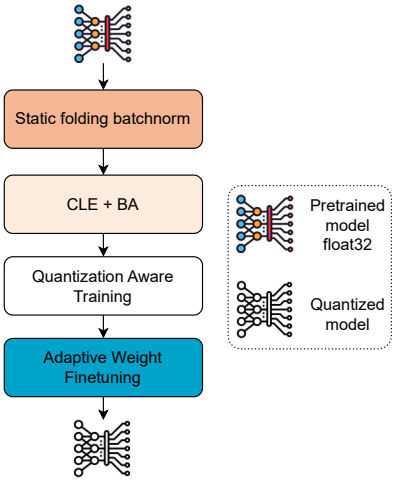

Fig. 3: The training flow of the proposed method.

$$\theta = \underset{\theta}{\text{argmin}}\ L(F(W,\theta)) + \lambda R(\theta) \tag{13}$$

where $F(W,\theta)$ is the quantization function applied to the weights selected for finetuning, parameterized by the learnable variable θ, and is defined as follows:

$$\begin{aligned} F(W,\theta) &= s.clip\left(\left\lceil \frac{W}{s} \right\rfloor + g(\theta), \alpha, \beta\right) \\ g(\theta) &= clip(\gamma \cdot \tanh(\theta), -1, 1) \end{aligned} \tag{14}$$

where γ is a scaling hyperparameter, and λ is a loss-balancing parameter. $R(\theta)$ is a regularization term that constrains the quantity $g(\theta)$ to take values only in $-1, 0, 1$, formulated as follows:

$$R(\theta) = g(\theta)^2.((1-g(\theta)^2)^2 \tag{15}$$

Unlike Adaround [21], which applies floor operations uniformly to all W/s terms and may disrupt the model state, our method preserves quantized weights and refines them using a learnable offset $g(\theta)$, defined as a clipped tanh function bounded in $[-1,1]$ (Eq. 14). As shown in Table II and Table V, our method outperforms both weight-freezing and Adaround.

V. Experiment

To evaluate the effectiveness of the proposed methods, we conduct experiments focusing on lightweight models, such as YOLO, MobileNet, EfficientNet, and ResNet18, which are commonly deployed on resource-constrained hardware. To highlight the capability of our methods under low-bit quantization, we apply 3-bit and 4-bit quantization to both weights and activations.

Experiment setup: Full flow of the proposed quantization method is illustrated in Fig. 3. Following [2], we first fold

TABLE II: The comparative results between the proposed method and other QAT approaches on both object detection and object classification tasks.

Task	Model	FP	LSQ [11]		PerChannel [2]		Damp [5]		RedYolo [4]		Ours	
			4a4w	3a3w	4a4w	3a3w	4a4w	3a3w	4a4w	3a3w	4a4w	3a3w
OD	YOLOX-T [15]	32.8	28.3	25.3	29.2	26.1	29.5	26.1	29.1	25.6	**30.5**	**28.2**
	YOLOv8-n [14]	37.3	33.1	29.9	33.4	30.5	33.9	31.3	33.3	30.9	**35.2**	**33.1**
	YOLOv9-t [16]	38.3	34.8	31.8	34.6	31.7	35.4	31.9	35.6	32.6	**36.5**	**34.7**
	YOLOv9-s [16]	46.8	43.8	40.1	43.8	41.3	44.1	41.8	44.6	40.3	**45.1**	**43.6**
CLS	MobileNetV1 [17]	70.6	67.6	64.7	67.7	65.2	68.2	65.7	68.7	66.1	**69.2**	**66.6**
	MobileNetV2 [18]	72.0	69.5	65.3	70.5	67.9	70.6	67.8	70.4	67.6	**71.1**	**68.2**
	EffictionNet-B0 [19]	77.1	75.9	73.2	76.3	73.4	76.1	73.5	75.6	73.1	**76.4**	**74.1**
	MobileNeXt-x1.4 [20]	76.1	74.3	71.8	74.2	71.8	74.9	72.0	74.7	72.3	**75.1**	**72.8**

TABLE III: The experimental results of the proposed method on various lightweight classification models evaluated on the ImageNet-1K dataset.

Model	Param (M)	Acc.	Method	
			4A4W	3A3W
MobileNetV1 [17]	4.2M	70.6	69.2	66.6
MobileNetV2 [18]	3.4M	72.0	71.1	68.2
MobileNetV3-L [23]	5.4M	75.3	73.7	72.0
MobileNeXt-x1.4 [20]	6.1M	76.2	75.1	72.8
EffictionNet-B0 [19]	5.3 M	77.1	76.4	74.1
ShuffleNetV2-1.0 [24]	2.3M	69.5	67.9	65.8
Resnet18 [25]	11.7M	69.8	68.6	66.4
SE-ResNet-50 [26]	25.6M	76.7	76.0	73.7
VGG-16 [27]	138M	72.9	72.5	71.2

TABLE IV: Experimental results of the proposed method on different lightweight object detection models are evaluated on the MS COCO dataset.

Model	Param (M)	mAP	Method	
			4A4W	3A3W
MobileNet SSD V2 [28]	3.4M	25.7	23.1	20.9
YOLOX-Nano [15]	0.9M	26.1	22.3	20.3
YOLOX-Tiny [15]	5.1M	32.8	30.5	28.2
YOLOv8-n [14]	3.2M	37.3	35.2	33.1
YOLOv8-s [14]	11.2M	44.9	43.5	41.8
YOLOv8-m [14]	25.9M	50.2	49.1	47.9
YOLOv9-t [16]	2.0M	38.3	36.5	34.7
YOLOv9-s [16]	7.1M	46.8	45.1	43.6

TABLE V: Table of ablation impact of each proposed method on the overall results. The symbols "-" and ✓ indicate that the method was not used and was used, respectively.

Method				Bit	Model		
ScaleGrad	EAD	AFW	AdaRound	a/w	YOLOv8n	Yolov9t	MBNetV2
-	-	-	-		33.1	34.8	69.5
✓	-	-	-		34.2	36.0	70.4
✓	✓	-	-	4	35.0	35.6	70.8
✓	✓	-	✓		34.8	36.1	70.5
✓	✓	✓	-		**35.2**	**36.5**	**71.1**
-	-	-	-		29.9	31.8	65.3
✓	-	-	-		31.7	33.3	67.3
✓	✓	-	-	3	32.6	34.5	67.6
✓	✓	-	✓		32.6	34.3	67.4
✓	✓	✓	-		**33.1**	**34.7**	**68.2**

Conv-BatchNorm-ReLU layers in the pretrained model. Then, we apply LSQ per-tensor quantization to both weights and activations. To improve quantization performance, we use Cross-Layer Equalization [1] and Bias Correction [1], and initialize activation scales with the Moving Average Min-Max method [22] on the validation set. We perform QAT for 80 epochs using ScaleGrad, 20 epochs for EAD with $\alpha_{init} = 0.1$, and 10 epochs for AFM, employing the Stochastic Gradient Descent optimizer with a learning rate of 0.001, momentum of 0.9, weight decay of 1e-5, batch size of 128, γ and λ set to 1.1 and 0.1, respectively and a StepLR learning rate scheduler. All experiments use PyTorch 11.12 on an NVIDIA V100 GPU.

A. Experiment result

1) Quantity result of proposed method: We conduct experiments to evaluate the proposed method on two benchmark datasets: MS COCO and ImageNet-1K, using a diverse set of CNN architectures.

As shown in Table III, on the image classification task (CLS), our proposed quantization method drives the performance of quantized models to approach that of full-precision models. Under 4-bit quantization, the accuracy gap between quantized and original models keeps below 1.6%, and within 4% for 3 bits. Larger models like VGG16 and SE-ResNet-50 show less than 0.7% drop, confirming the method's effectiveness on deeper networks.

For object detection task (OD) (Table IV), our method shows a high mAP score on lightweight detectors like YOLO and SSD. For instance, YOLOv8-s and YOLOv9-s maintain under 1.8% loss at 4 bits and around 3% at 3 bits. This presents a favorable trade-off for edge deployment, offering up to 8× memory savings with minimal accuracy loss.

2) Comparison of results with other methods: We compare our method with existing QAT approaches LSQ, Per-Channel Quantization, Damping Loss, and RedYolo—on both object detection (using lightweight YOLO models) and classification (using MobileNet and EfficientNet variants). As shown in Table II, our method outperforms all baselines in 4 bits and 3 bits settings. Compared to LSQ, it achieves more than 1.5% improvement in most models, including a 2.2% mAP gain in YOLOX-T and a 1.6% top-1 accuracy boost in MobileNetV1. It also outperforms Damping Loss by over 1% at 4 bits and 1.7% at 3 bits. Unlike Per-Channel Quantization and RedYolo, our method avoids extra channel-wise parameters, maintaining compatibility with hardware accelerators while achieving better overall performance.

B. Ablation Study

We conduct ablation studies on lightweight models to assess the impact of each proposed algorithm on quantization performance. As shown in Table V, all methods improve upon baseline LSQ (first row). ScaleGrad consistently boosts performance by over 1.5% by smoothing gradients. Adding EAD yields further gains, for instance, Yolov9t witnesses a 1.2% accuracy increase. AFW provides the highest improvements, with MobileNetv2 gaining 1.6% at 4-bit and 2.9% at 3-bit quantization. Notably, AFW outperforms AdaRound, which in some cases slightly degrades performance (e.g., YOLOv8-n drops 0.2% mAP). These results confirm that combining AFW with ScaleGrad and EAD yields the best low-bit quantization results.

VI. Conclusion

In this paper, we conduct a comprehensive analysis of oscillation phenomena in low-bit QAT, combining both theoretical insights and empirical observations. We find that the key causes of instability and suboptimal convergence of QAT due to the abrupt fluctuations in the gradient of scaling factors and the dependency of latent weight gradients on quantization transitions rather than their values. To address these issues, we propose three techniques: Gradient Scaling Aware Distance, EMA Aware Distance for Update Weight, and Adaptive Finetuning Weight. Our methods stabilize training dynamics and improve performance for classification and object detection tasks with 4 bits and 3 bits of quantization. They do not introduce extra computational overhead during inference, making them practical for resource-constrained deployment. The future work aims to develop the AFW method to support end-to-end integration during training and further enhance quantization performance.

References

[1] M. Nagel, M. v. Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1325–1334, 2019.

[2] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. Van Baalen, and T. Blankevoort, "A white paper on neural network quantization," *arXiv preprint arXiv:2106.08295*, 2021.

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

[4] K. Gupta and A. Asthana, "Reducing the side-effects of oscillations in training of quantized yolo networks," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024.

[5] M. Nagel, M. Fournarakis, Y. Bondarenko, and T. Blankevoort, "Overcoming oscillations in quantization-aware training," in *International Conference on Machine Learning*, pp. 16318–16330, PMLR, 2022.

[6] J. Wenshøj, B. Pepin, and R. Selvan, "Oscillations make neural networks robust to quantization," *arXiv preprint arXiv:2502.00490*, 2025.

[7] S.-Y. Liu, Z. Liu, and K.-T. Cheng, "Oscillation-free quantization for low-bit vision transformers," in *International conference on machine learning*, pp. 21813–21824, PMLR, 2023.

[8] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4852–4861, 2019.

[9] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X.-s. Hua, "Quantization networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

[10] J. Lee, D. Kim, and B. Ham, "Network quantization with element-wise gradient scaling," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6448–6457, 2021.

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

[12] A. Défossez, Y. Adi, and G. Synnaeve, "Differentiable model compression via pseudo quantization noise," *Preprint arXiv:2104.09987*, 2021.

[13] 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*, pp. 696–697, 2020.

[14] M. Sohan, T. Sai Ram, and C. V. Rami Reddy, "A review on yolov8 and its advancements," in *International Conference on Data Intelligence and Cognitive Informatics*, pp. 529–545, Springer, 2024.

[15] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[16] C.-Y. Wang, I.-H. Yeh, and H.-Y. Mark Liao, "Yolov9: Learning what you want to learn using programmable gradient information," in *European conference on computer vision*, pp. 1–21, Springer, 2024.

[17] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[18] 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*, pp. 4510–4520, 2018.

[19] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*, pp. 6105–6114, PMLR, 2019.

[20] D. Zhou, Q. Hou, Y. Chen, J. Feng, and S. Yan, "Rethinking bottleneck structure for efficient mobile network design," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pp. 680–697, Springer, 2020.

[21] M. Nagel, R. A. Amjad, M. Van Baalen, C. Louizos, and T. Blankevoort, "Up or down? adaptive rounding for post-training quantization," in *International conference on machine learning*, PMLR, 2020.

[22] M. Fournarakis and M. Nagel, "In-hindsight quantization range estimation for quantized training," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3063–3070, 2021.

[23] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

[24] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[26] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[28] G. Ramesh, Y. Jeswin, R. R. Divith, S. BR, K. Kiran Raj, *et al.*, "Real time object detection and tracking using ssd mobilenetv2 on jetbot gpu," in *2024 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, pp. 255–260, IEEE, 2024.