Reducing the Side-Effects of Oscillations in Training of Quantized YOLO Networks

Kartik Gupta Akshay Asthana
Seeing Machines, Australia
{kartik.gupta,akshay.asthana}@seeingmachines.com

Abstract

*Quantized networks use less computational and memory resources and are suitable for deployment on edge devices. While quantization-aware training (*QAT*) is a well-studied approach to quantize the networks at low precision, most research focuses on over-parameterized networks for classification with limited studies on popular and edge device friendly single-shot object detection and semantic segmentation methods like YOLO. Moreover, majority of* QAT *methods rely on Straight Through Estimator (*STE*) approximation which suffers from an oscillation phenomenon resulting in sub-optimal network quantization. In this paper, we show that it is difficult to achieve extremely low precision (4-bit and lower) for efficient YOLO models even with SOTA* QAT *methods due to oscillation issue and existing methods to overcome this problem are not effective on these models. To mitigate the effect of oscillation, we first propose Exponentially Moving Average (*EMA*) based update to the* QAT *model. Further, we propose a simple* QAT *correction method, namely* QC*, that takes only a single epoch of training after standard Quantization-Aware Training (*QAT*) procedure to correct the error induced by oscillating weights and activations resulting in a more accurate quantized model. With extensive evaluation on COCO dataset using various YOLO5 and YOLO7 variants, we show that our correction method improves quantized YOLO networks consistently on both object detection and segmentation tasks at low-precision (4-bit and 3-bit).*

1. Introduction

Deep neural networks have achieved remarkable success in various applications, including image classification, object detection, and semantic segmentation. However, deploying these models on edge devices such as mobile phones, smart cameras, and drones poses a significant challenge due to their limited computational and memory resources. These devices typically have limited battery life, storage capacity, and processing power, making it challenging to execute complex neural networks. To overcome these challenges, researchers have developed techniques for optimizing neural networks to reduce their computational and memory requirements while maintaining their accuracy. One such line of research is QAT, which reduces the number of bits used to represent the network parameters, and activations resulting in smaller model sizes and faster inference times. Existing QAT [9, 16, 21, 23] methods have made remarkable progress in quantizing neural networks at ultra-low precision with the effectiveness of *Straight Through Estimator (*STE*)* approximation still being a point of study. Previous works [10, 32] have proposed smooth approximation of rounding function to avoid the use STE approximation but STE is still considered to be the de-facto method for approximating gradient of quantization function during propagation due to its simplicity. Furthermore, recent works [7, 25] have shown oscillation issue affects quantization performance of efficient network architecture at low-precision due to STE approximation in QAT.

Apart from that, the majority of QAT literature focuses on image classification, and quantization performance achieved on such classification tasks does not necessarily translate onto downstream tasks such as object detection, and semantic segmentation. In this paper, we focus on the more challenging task of quantizing the single-shot efficient detection networks such as YOLO5 [30] and YOLO7 [31] at low-precision (3-bits and 4-bits). Furthermore, we show that the oscillation issue is even more prevalent on these networks and the gap between full-precision and quantized performance is far from what is usually observed in QAT literature. We also show that apart from latent weights, learnable scale factors for both weights and activations are also affected by the oscillation issue in YOLO models and latent weights around quantization boundaries are sometimes closer to optimality than quantization levels. This indicates that per-tensor quantization worsens the issue of oscillation.

To deal with the issues of oscillations in YOLO, we propose *Exponential Moving Average (*EMA*)* in QAT, that smoothens out the effect of oscillations and *Quantization*

*Correction (*QC*)*, that corrects the error induced due to oscillation after each quantized layer as a post-hoc step after performing QAT. By mitigating side-effects of oscillations, these two methods in combination achieve state-of-the-art quantization results at 3-bit and 4-bit on YOLO5 and YOLO7 for both object detection and semantic segmentation on extremely challenging COCO dataset.

Below we summarize the contributions of this paper:

- We show that quantization on most recent efficient YOLO models such as YOLO5 and YOLO7 is extremely challenging even with state-of-the-art QAT methods due to oscillation issue.
- Our analysis finds that the oscillation phenomenon does not only affect latent weights but also affects the training of learnable scale factors for both weights and activations.
- We propose two simple methods namely EMA and QC, that can be used in combination with any QAT technique to reduce the side-effects of oscillations during QAT on efficient networks.
- With extensive experiments on COCO dataset for both object detection and semantic segmentation tasks, we show that our methods in combination consistently improve quantized YOLO5 and YOLO7 variants and establish a state-of-the-art at ultra-low precision (4-bits and 3-bits).

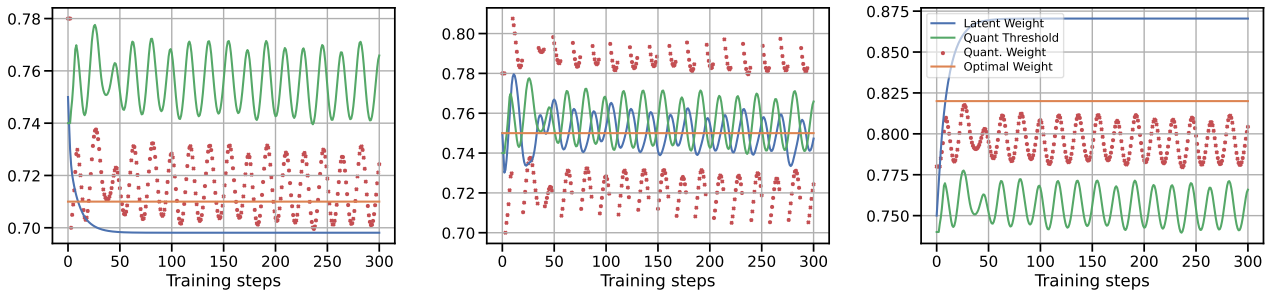
2. Related Work

Quantization-Aware Training. In recent years, model quantization has been a topic of great interest in the deep learning community due to neural networks continuously scaling exponentially in terms of compute. Neural network quantization approaches can be broadly categorized into: Post-training Quantization (PTQ) and Quantization-Aware Training (QAT). Though PTQ [2, 22, 23, 26] is faster and does not rely on whole training data, it yields significant performance degradation at low-bit precision. QAT is the focus of our work, and it has been well-studied in literature [6, 9, 10, 14, 21, 33].

The STE is a de-facto method for backpropagating through the non-differentiable rounding function in QAT. The effectiveness of STE has been the point of argument in recent literature. Ajanthan *et al*. [1] proposed a mirror descent formulation for neural network quantization and established the connection between STE approximation and mirror descent framework for constrained optimization. Lee *et al*. [14] showed that the STE leads to bias in gradients and proposed gradient scaling by the distance of latent weights from quantization boundaries. Gong *et al*. [10] attempt to mitigate the issues caused by STE for low-bit quantization by using differentiable hyperbolic tangent functions to simulate the rounding function in the backward pass. Similar to that, Yang *et al*. [32] approximate the rounding function using smooth sigmoid functions to address the gradient bias in STE.

Recent works [7, 25] have identified oscillation as a side-effect of STE approximation during QAT. Defossez *et al*. [7] proposed additive Gaussian noise to mimic the quantization noise and replace it as quantization operation during QAT to prevent weight oscillations and biased gradients resulting from STE. Nagel *et al*. [25] also showed that weight oscillation seriously impacts QAT performance, specifically on efficient networks comprising depth-wise convolutions due to STE approximation of rounding function. They propose to constrain the latent weights to avoid oscillation by either regularizing them to their quantized states or by freezing them. Recently, Liu *et al*. [20] studied the issue of oscillations on vision transformers and proposed fixed scale factors for weight quantization and query-key re-parameterization to mitigate the negative influence of oscillation. However, their proposed approach is specifically targeted at solving oscillation phenomenon on transformers architecture.

Quantization of Object Detectors. The majority of existing neural network quantization literature focuses on the image classification task rather than real-world downstream tasks such as object detection or semantic segmentation. Some recent works do explore the quantization of object detectors to improve the efficiency of these models. Jacob *et al*. [15] proposed a quantization scheme using integer-only arithmetic and performing object detection on COCO dataset but the approach is only effective for 8-bit quantization. Li et al. [17] observed training instability during the quantized fine-tuning of RetinaNet and propose mutliple solutions specific to RetinaNet architecture. These solutions are not applicable on more efficient object detectors like YOLO. Ding *et al*. [8] proposed ADMM based weight quantization framework for YOLO3 but do not address the issue of oscillations and activation quantization. Due to the discrete nature of quantization functions, gradient estimates are known to be noisy, which affects Stochastic Gradient Descent (SGD) updates. To overcome that, Zhuang *et al*. [34] proposed to utilize a full precision auxiliary module to enable stable training of quantized object detectors. Furthermore, Zhuang *et al*. [5] proposed multi-level Batch Normalization (BN) to accurately calculate batch statistics for each pyramid level in RetinaNet [18] and FCOS [29]. The proposed approach is specific to pyramid-level architecture in RetinaNet and FCOS that share BN layers at different levels of the pyramid, but not applicable on more recent efficient SOTA YOLO models. In this paper, we identify the gap in the recent literature on quantized object detection and introduce state-of-the-art quantized object detectors using YOLO5 [30] and YOLO7 [31] variants.



(a) (b) (c)

Figure 1. *A toy 3D regression problem to demonstrate the oscillation issue in weight and activation quantization. (a),(b) and (c) Trajectory of different weights during the optimization. Here, it can be seen that oscillation not only affects the latent weights but also the learnable scale factors. Here, "quantization threshold" refers to the quantization boundary in the latent space.*

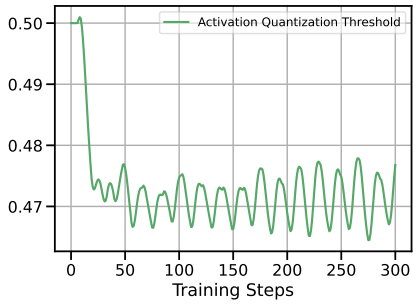


Figure 2. *Trajectory of activation quantization threshold during training for same toy example as in Fig. 1. Even the scale factors for activation quantization oscillate during the optimization.*

3. Preliminaries

Here we provide a brief background on the quantization-aware training (QAT) and introduce the issue of oscillations in QAT using a small toy example.

3.1. Quantization-aware Training (QAT)

Quantization-aware training can be achieved by simulating the quantized computational operations during the training of the neural networks. The forward pass of the neural network is encapsulated with a quantization function $q(\cdot)$ that converts full-precision weights and activations into quantized weights and activations. It takes input vector $\mathbf{w}$ and returns quantized output $\widehat{\mathbf{w}}$ given by:

$$\widehat{\mathbf{w}} = q(\mathbf{w}; s, u, v) = s \cdot \text{clip}\left(\left\lfloor \frac{\mathbf{w}}{s} \right\rceil, u, v\right), \qquad (1)$$

where $\lfloor \cdot \rceil$ is the *round-to-nearest* operator, $\text{clip}(\cdot, b, c)$ is a clipping function with lower and upper bounds b and c, respectively, s is a quantization scale factor. Here, u and v denote the minimum and maximum range after the quantization. The scale factor s can be learned [4, 9] during quantization-aware training through backpropagation by approximating the gradient of the rounding operator with STE. The original full-precision weights $\mathbf{w}$ are commonly referred to as *latent weights* and gradient descent is performed only on the latent weights for the update. During inference, the quantized weights $\widehat{\mathbf{w}}$ are used to compute the convolutional or dense layer output.

Due to the non-differentiability of the quantization function, it is non-trivial to backpropagate through the neural network embedded with such an operation. To this purpose, a commonly used technique for alleviating this issue involves using the straight-through estimator (STE) [3, 12]. The basic idea of STE is to approximate the gradient of the rounding operator as 1 within the quantization limits.

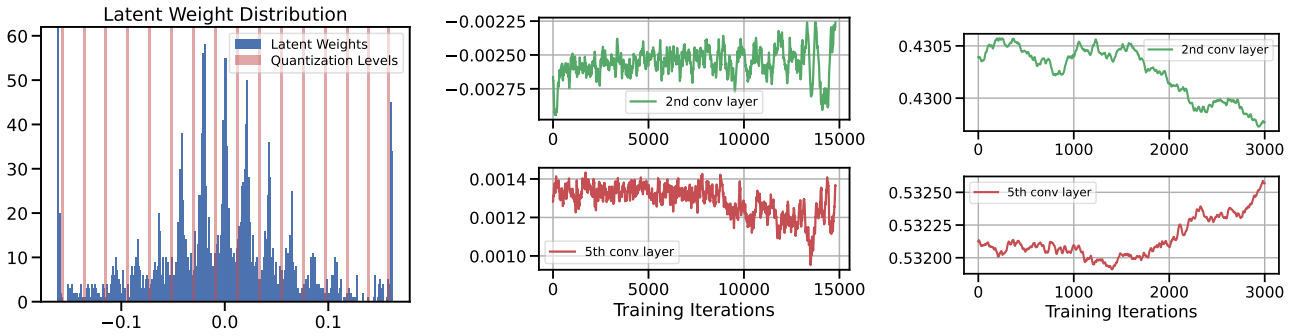
3.2. Oscillations in QAT

Recent works [7, 25] observed the oscillation phenomenon as a side-effect of quantization-aware training with STE approximation. Due to STE approximation that passes the gradient through the quantization function, the latent weights oscillate around the quantization threshold.

We illustrate the oscillation phenomenon in STE based QAT using a 3D toy regression problem with both weights and input quantization at 1-bit. Here, we optimize a latent weight vector $\mathbf{w}$ and scale factors $s_{\mathbf{w}}, s_{\mathbf{x}}$ for weights and activations respectively with the following objective:

$$\arg\min_{\mathbf{w}, s_{\mathbf{w}}, s_{\mathbf{x}}} \mathcal{L}(\mathbf{w}, s_{\mathbf{w}}, s_{\mathbf{x}}) = \mathbb{E}_{\mathbf{x}\sim U} \left\| \mathbf{x}\mathbf{w}_* - q(\mathbf{x}, s_{\mathbf{x}}) q(\mathbf{w}, s_{\mathbf{w}}) \right\| \qquad (2)$$

Here, $\mathbf{w}_*$ refers to the optimal ground truth value and $q(\cdot)$ is the quantization function defined in Eq. (1). We randomly sample data vector $\mathbf{x}$ from a uniform distribution U within range $[0, 1)$. The oscillation behavior during the optimization is shown in Fig. 1. Note that the quantized weights oscillate around the quantization threshold instead of converging close to the optimal value in Fig. 1a, Fig. 1b, and Fig. 1c. Though the underlying reason for oscillation of quantized weights in Fig. 1b is the oscillation of latent weights, oscillation of quantized weights in Fig. 1a and Fig. 1c happens due to oscillation of learnable scale factors. Furthermore, this oscillation behavior is not just restricted to quantized weights but also impacts the quantized



(a) (b) (c)

Figure 3. *Oscillation issue in YOLO5-n variant trained on COCO dataset at 4-bit precision using* LSQ *[9]. (a) Latent weight distributions of 5^{th} convolution layer in YOLO5-n, and Scale factors for (b) weight quantization during the last 15K iterations, (c) Scale factors for activation quantization during the last 3K iterations of QAT in YOLOv5-n. Here, it can be clearly observed that latent weights peak exactly at quantization thresholds. Also, both quantization scale factors of weights and activations are not stable even until the end of training.*

activations as can be observed in Fig. 2. In the case of activation quantization, scale factors for activations oscillate as well and this can lead to further performance degradation of quantized models due to sub-optimal minima. Here, we would like to point out that previous work [25] only accounts for the issue of oscillation of latent weights, whereas we show oscillations in learnable scale factors can lead to performance degradation during QAT.

4. Side-effects of Oscillations in YOLO

The issue of oscillating weights and activations in quantization-aware training is not just restricted to a small toy problem but occurs in practice on YOLO networks [30,31] trained for the task of object detection and semantic segmentation. This leads to a significant loss in the accuracy of quantized YOLO models. In this section, we show the evidence of oscillation issue prevalent in weights and activations of quantized YOLO networks and how learning a single scale factor for each tensor is the underlying cause for sub-optimal latent weights.

4.1. Oscillation Issue in YOLO networks

We demonstrate the oscillation issue in YOLO networks using the latent weight distribution of 5^{th} layer in YOLO5-n variant [30] trained with 4-bit quantization on COCO dataset [19] using Learnable Step-size Quantization (LSQ) [9] with STE approximation in Fig. 3a. Most of the latent weights lie in between the quantization levels and the peaks of distribution lie on quantization thresholds rather than the quantization levels itself. Since most of the latent weights lie around quantization thresholds, they tend to keep switching their quantization state even at the end of the training as also shown in [25]. Further, we plot the learnable scale factors used to quantize weights as well as activations in Fig. 3b and Fig. 3c respectively. The quantization scale factors remain unstable even until the end of quantization-aware training. The oscillation issue does not only affect the latent weights but also affects the scale factors of both weights and activations. This leads to a sub-optimal quantization state of both weights and activations of the final QAT model. Here, we would like to highlight that previous work [25] observed the issue of oscillations in latent weights but here we show that oscillation also affects quantization scale factors corresponding to weights and activations.

Table 1. *Comparison between hard-rounding and soft-rounding (k=0.45). Here, soft-rounding outperforms hard-rounding function which is actually even used during* QAT.

Method	#-bits	YOLO5-n	YOLO5-s
Full precision	FP	28.0	37.4
Hard Rounding	4-bit	20.6	32.5
	3-bit	15.2	27.2
Soft Rounding	4-bit	**21.2**	**32.6**
	3-bit	**16.2**	**27.7**

4.2. Analysis using threshold-based Soft-Rounding

Oscillation dampening [25] was introduced as one of the techniques to reduce the side-effect of oscillations. This method in essence regularizes the latent weights such that the distribution of latent weights and quantization weights overlap each other. We step away from that and look at the optimality of latent weights in their un-quantized state. For this analysis, we modify the original quantization to deduce a soft-rounding function that allows quantizing the weights closer to quantization levels and leave the latent weights around the quantization threshold in their latent state. We describe our soft-rounding function q^* that can be used to softly round weights or activations using a threshold k as

$$\mathbf{w}^* = q^*(\mathbf{w}; s, u, v) = s \cdot \text{clip}\left(\text{softround}\left(\frac{\mathbf{w}}{s}\right), u, v\right), \tag{3}$$

$$\text{where} \quad \text{softround}\,(x) = \begin{cases} \lfloor x \rceil & \text{if } |x - \lfloor x \rceil| \leq k \\ x & \text{otherwise.} \end{cases}$$

This soft-rounding function can be used on both weights and activations to evaluate whether latent weights and activations stuck at the quantization thresholds (see Fig. 3a) are already closer to their optimal state. We replace the quantization function described in Eq. (1) with this soft-rounding function (at $k = 0.45$) in all the quantized layers of pre-trained quantized YOLO models and evaluate the performance on COCO dataset and the results are presented in Table 1. Surprisingly, we found that the weights or activations in their latent state produce equivalent or better performance than the ones in the quantized state. This indicates that latent weights oscillate around the quantization boundaries partly because not all weights or activations within a tensor can be quantized with the same single scale factor as in the case of per-tensor quantization. If a single scale factor per-tensor is chosen, some weights can never reach optimality due to the limitation of per-tensor quantization.

5. Approach

In this section, we first provide the notations and formulate the quantization-aware training optimization problem with learnable scale factors. Then, we introduce our two simple methods to deal with side-effects of error induced due to oscillation in network parameters during QAT.

Problem setup. For notational convenience, we consider a fully-connected neural network with weights $\mathbf{W}^l \in \mathbb{R}^{\mathrm{N}_l \times \mathrm{N}_{l-1}}$, biases $\mathbf{b}^l \in \mathbb{R}^{\mathrm{N}_{l-1}}$, pre-activations $\mathbf{h}^l \in \mathbb{R}^{\mathrm{N}_l}$, and post-activations $\mathbf{a}^l \in \mathbb{R}^{\mathrm{N}_l}$, for $l \in \{1, \ldots, \ell\}$ up to K layers. Then, the feed-forward dynamics of the neural network with simulated quantization can be formulated as:

$$\mathbf{a}^l = \phi(\text{BatchNorm}(\mathbf{h}^l)), \qquad \mathbf{h}^l = \widehat{\mathbf{W}}^l \widehat{\mathbf{a}}^{l-1} + \mathbf{b}^l, \tag{4}$$

$$\text{where} \qquad \widehat{\mathbf{W}}^l = q(\mathbf{W}^l, s^l_{\mathbf{W}}), \qquad \widehat{\mathbf{a}}^{l-1} = q(\mathbf{a}^{l-1}, s^{l-1}_{\mathbf{a}}).$$

Here $\phi : \mathbb{R} \rightarrow \mathbb{R}$ is an elementwise nonlinearity, and the input is denoted by $\mathbf{a}^0 = \mathbf{x}^0 \in \mathbb{R}^{\mathrm{N}}$. We denote quantized weights and activations by $\widehat{\mathbf{W}}^l$ and $\widehat{\mathbf{a}}^l$ respectively and use $s^l_{\mathbf{W}}$, $s^l_{\mathbf{a}}$ to represent their respective quantization scale factors. For simplicity of notation, we further also express network weight parameters corresponding to all the layers as $\mathcal{W} = \{\mathbf{W}^i\}_{i=1}^{\ell}$. Similarly, we represent the vector corresponding to all scale factors for quantization of various weight tensors as $\mathbf{s_W} = \{s^i_{\mathbf{W}}\}_{i=1}^{\ell}$ and scale factors for quantization of activation tensors as $\mathbf{s_a} = \{s^i_{\mathbf{a}}\}_{i=1}^{\ell}$. Given dataset $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{n}$, the typical neural network optimization problem for quantization-aware training with learnable scale factors can then be formulated as:

$$\underset{\mathcal{W}, \mathbf{s_w}, \mathbf{s_a}}{\arg\min}\, L(\mathcal{W}, \mathbf{s_w}, \mathbf{s_a}; \mathcal{D}). \tag{5}$$

We now explain our two simple techniques to overcome the issue of oscillating weights and scale factors and compensate for sub-optimal latent weights stuck at the quantization boundaries.

5.1. Exponential Moving Average (EMA) to Smoothen effect of Oscillations

Weight averaging of multiple local minima by using multiple model checkpoints attained at cyclic learning rates with restarts, has been shown [13, 28] to lead to better generalization and wider minima. The idea of using weight averaging for final model inference was earliest suggested in [27]. Later, the semi-supervised learning method [28] and self-supervised learning methods [11] utilized exponential moving average of weights to learn in a knowledge distillation manner.

To overcome the oscillating weights and quantization scale factors due to STE approximation, we propose exponential moving average of latent weights and scale factors for both weights and activations during the optimization. STE approximation approach leads to latent weights moving around the quantization boundary which leads to constantly changing latent weight states. Exponential moving average can take into account model weights at the last several steps of training and smoothen out the oscillation behavior and come up with the best possible latent state for oscillating weights. The final quantized state inference can be done using EMA weights and quantization parameters instead of the latest update.

For trainable network weight parameters $\mathbf{W}^l_{(t)}$ at layer l, we can compute the corresponding exponentially moving average weights $\mathbf{W'}^l_{(t)}$ at t iteration as:

$$\mathbf{W'}^l_{(t)} = \alpha \cdot \mathbf{W'}^l_{(t-1)} + (1 - \alpha) \cdot \mathbf{W}^l_{(t)}. \tag{6}$$

Similarly, we can also calculate the exponential moving average scale factors for both weights and activations as:

$$\mathbf{s'}_{\mathbf{W}\,(t)} = \alpha \cdot \mathbf{s'}_{\mathbf{W}\,(t-1)} + (1 - \alpha) \cdot \mathbf{s}_{\mathbf{W}\,(t)} \tag{7}$$

$$\mathbf{s'}_{\mathbf{a}\,(t)} = \alpha \cdot \mathbf{s'}_{\mathbf{a}\,(t-1)} + (1 - \alpha) \cdot \mathbf{s}_{\mathbf{a}\,(t)}. \tag{8}$$

Here, α is used as a decay parameter that can be tuned to account for approximately $1/(1-\alpha)$ last SGD updates to achieve the EMA model. We keep the decay parameter α to be 0 at the start of the QAT procedure to enable larger updates at the start of the training. The EMA parameters are updated after end of every iteration of the training update and thus do not require backpropagation. It is important to note here that oscillation dampening and iterative freezing proposed in [25], are only applicable on weights but oscillation issue due to scale factors is even present in activation quantization as shown in Sec. 2. To overcome that issue, EMA on scale factors of activations can tackle it more appropriately. Furthermore, we would also highlight here that

other non-trainable parameters such Batch Normalization (BN) statistics in deep neural networks already utilize exponential moving average can improve the unstable BN statistics if the momentum value is chosen appropriately. Recent methods [25], suggested that BN Statistics re-estimation enables improvement in the corrupted BN statistics occurring due to oscillation of latent weights. We would like to mention that corrupted BN statistics is not the only reason for performance degradation resulting from oscillations in QAT. Nevertheless, our EMA models yield stable updates to both latent weight, activations, and their respective scale factors.

5.2. Post-hoc Quantization Correction (QC)

In this section, we propose a simple correction step that can be performed in a post-hoc manner after quantization-aware training to overcome the error induced by oscillating latent weights and scale factors. As shown empirically in the Sec. 4, oscillation results in the majority of latent weights hanging at the quantization boundaries. We have further shown in Sec. 4.2 that latent weights hanging at quantization boundaries are already closer to optimality than their nearest quantization levels as the soft-rounded (based on Eq. (3)) YOLO models yield better performance than quantized ones. The oscillation of scale factors mainly happens due to ineffective quantization with a single quantization scale factor per-tensor apart from the bias of STE approximation. Intuitively, different regions in the tensor might require different scale factors for an accurate quantized approximation.

Our post-hoc correction quantization step simply transforms the pre-activations $\mathbf{h}^l \in \mathbb{R}^{\mathrm{N}_l}$ of all l layers using an affine function, to compensate for error induced in matrix multiplications due to oscillations during the quantization-aware training. We can now formulate the modified feed-forward dynamics of the quantized neural network for the post-hoc error correction step as:

$$\widetilde{\mathbf{h}}^l = \boldsymbol{\gamma}^l \cdot \mathbf{h}^l + \boldsymbol{\beta}^l \,, \quad \mathbf{h}^l = \widehat{\mathbf{W}}^l \widehat{\mathbf{a}}^{l-1} + \mathbf{b}^l \,, \tag{9}$$

$$\mathbf{a}^l = \phi(\mathrm{BatchNorm}(\widetilde{\mathbf{h}}^l)) \tag{10}$$

Here, $\widetilde{\mathbf{h}}^l$ denotes the modified pre-activations after the affine transformation in layer l. Also, for layer l we represent the affine function with scale correction parameters $\boldsymbol{\gamma}^l \in \mathbb{R}^{\mathrm{N}_l}$ and shift correction parameters $\boldsymbol{\beta}^l \in \mathbb{R}^{\mathrm{N}_l}$. For simplicity of notation, we further express a set of all correction scale parameters with $\mathcal{G} = \{\boldsymbol{\gamma}^i\}_{i=1}^{\ell}$ and correction shift parameters with $\mathcal{B} = \{\boldsymbol{\beta}^i\}_{i=1}^{\ell}$. We initialize these correction parameters as identity transformation. We then optimize for these correction parameters via backpropagation starting from a pre-trained QAT model with the following objective:

$$\arg\min_{\mathcal{G},\mathcal{B}} L(\mathcal{W}, \mathcal{G}, \mathcal{B}, \mathbf{s_w}, \mathbf{s_a}; \mathcal{D}_c). \tag{11}$$

We train these correction parameters on a small calibration set $\mathcal{D}_c$, which is also part of the training set. Notice that, for a typical convolutional layer these correction factors will have dimensions the same as the number of output channels after the convolution operation. We would like to highlight that these extra set of correction parameters can be absorbed in Batch Normalization (BN) trainable parameters generally succeeding a convolution layer and do not result in an extra computational load on the hardware. It is important to note that our correction step is different from the BN re-estimation step [25] where BN statistics are re-estimated on the dataset after QAT. BN re-estimation cannot recover from quantization error accumulated in forward propagation of quantized neural network, unlike our post-hoc correction step. In fact, re-estimating BN statistics is not required since the exponential moving average in BN statistics can enable a stable state of statistics if the momentum value is chosen appropriately. Furthermore, these correction parameters can also be stored as quantization scale factors by converting per-tensor quantization to per-channel quantization. The conversion from per-tensor quantization to per-channel quantization is a natural outcome of batch normalization folding [24] where batch normalization parameters are folded into the quantization scale factors of weights to get rid of BN at inference. Previous work [25] on weight oscillations in QAT only evaluate their quantization-aware training mechanism on per-tensor quantization but still keep the BN layers intact in train mode during the training. Their main motivation for oscillation avoidance is to get rid of corrupted BN statistics during the training. However, general practise [23] to support per-tensor quantization is to fold BN parameters before QAT.

6. Experiments

In this section, we evaluate the effectiveness of our proposed EMA and QC mechanisms to deal with side-effects of oscillations during QAT on various YOLO5 and YOLO7 variants on COCO dataset [19]. In all the experiments, we perform both weights and activation quantization. We present state-of-the-art results for low-precision (4-bit and 3-bit) on all YOLO5 and YOLO7 variants for object detection. We also compare our method against standard baselines such as LSQ [9] and Oscillation dampening [25]. We also perform some ablation studies to reflect the improvement of our method in comparison to per-channel quantization. Finally, we establish a state-of-the-art quantized YOLO5 model on the task of semantic segmentation using COCO dataset. In summary, our results establish new state-of-the-art for quantized YOLO5 and YOLO7 at low-bit precision while outperforming comparable baselines.

Experimental Setup. Similar to [25], we apply LSQ [9] based weight and activation quantization. Since object de-

Table 2. *Our quantization-aware training performance using mAP metric for object detection task on the COCO dataset. * denotes first and last layers are trained at 4-bit quantization.*

Network	# Params	FP	Ours (EMA)			Ours (EMA+ QC)		
			4-bit	3-bit	4-bit*	4-bit	3-bit	4-bit*
YOLO5-n	1.87M	28.0	22.1	16.3	16.5	**23.8**	**18.2**	**20.4**
YOLO5-s	7.23M	37.4	33.1	28.5	25.6	**34.0**	**30.2**	**32.0**
YOLO5-m	21.2M	45.2	42.1	38.5	38.5	**42.8**	**40.0**	**40.1**
YOLO5-l	46.6M	49.0	45.9	43.1	38.0	**46.6**	**44.0**	**43.6**
YOLO5-x	86.7M	50.7	47.8	45.9	40.6	**47.9**	**46.8**	**45.2**
YOLO7-tiny	6.23M	37.5	34.6	30.3	32.8	**35.2**	**31.0**	**34.3**
YOLO7	37.6M	51.2	48.7	46.2	46.3	**48.9**	**46.8**	**47.6**

tection is a complex downstream task and quantization can be very challenging, following the practice of existing literature [9], we quantize the first and last layer with 8-bit. During QAT, we use per-tensor quantization [16] and learn the quantization scaling factor using backpropagation [9] with a learning rate of 0.0001 in ADAM optimizer. Our QAT starts from a pre-trained full-precision network and is performed for 100 epochs. For all our QAT experiments, we use EMA decay rate of 0.9999. In QC, we train using ADAM optimizer with a learning rate of 0.0001 to learn the correction scale factors and shift factors. We train correction factors for a single epoch while keeping the Batch Normalization (BN) statistics fixed. Rest of hyperparameters are used as default based on official YOLO5[1] and YOLO7[2] implementations. Since, both LSQ and Oscillation dampening perform experiments only on ImageNet, we reimplemented their methods for the task of object detection on YOLO. All our results are reported with standard object detection or semantic segmentation metric, namely mAP. Our code is in PyTorch and the experiments are performed on NVIDIA A-40 GPUs.

6.1. Results on YOLO based object detection

We evaluate both of our EMA and QC techniques using LSQ [9] on YOLO5 and YOLO7 variants on COCO dataset for object detection task. We present results at different levels of precision *i.e*. 3-bit, 4-bit, and 4-bit with even the first and last layer quantized to 4-bit. The object detection of quantized YOLO5 and YOLO7 networks obtained by our proposed methods and their full precision (FP32) training are reported in Table 2.

Our QC method just by performing a post-hoc correction step consistently improves the EMA significantly for all different network architectures at 4-bit and 3-bit quantization. The improvement are especially significant on most efficient variants namely YOLO5-n and YOLO5-s at 3-bit ($\approx 4-6\%$) and 4-bit ($\approx 2\%$). Furthermore, even in the case of full quantization where even the first and last layers are quantized, our 4-bit quantization results using QC consistently improve on our QAT models trained with EMA. This clearly shows that latent weights that are stuck along the quantization thresholds can still be very useful if the error induced by those weights can be corrected using correction scale factors and shift factors learnt in our QC method. Overall, our combined EMA and QC method can reduce the gap between full precision models and 4-bit quantized models for all YOLO5 and YOLO7 variants with a margin of around $\leq 2.5\%$.

[1]YOLO5: https://github.com/ultralytics/yolov5
[2]YOLO7: https://github.com/WongKinYiu/yolov7

Table 3. *Comparison between LSQ [9], Oscillation dampening [25], and our proposed method for quantization-aware training using mAP metric for object detection on COCO dataset.*

Method	#-bit	YOLO5-n	YOLO5-s	YOLO7-tiny
Full-Precision	32-bit	28.0	37.4	37.5
LSQ [9]	4-bit	20.6	32.4	32.9
Osc. Damp. [25]		21.5	32.9	33.5
Ours (EMA)		22.1	33.1	34.6
Ours (EMA+QC)		**23.8**	**34.0**	**35.2**
LSQ [9]	3-bit	15.2	27.2	28.4
Osc. Damp. [25]		16.4	27.5	29.2
Ours (EMA)		16.4	28.5	30.3
Ours (EMA+QC)		**18.2**	**30.2**	**31.0**

6.2. Comparison against baselines

We also perform evaluation comparisons of our EMA and QC techniques using LSQ [9] on YOLO5 and YOLO7 variants on COCO dataset for object detection task against baseline methods namely, LSQ [9] and Oscillation dampening [25] at 4-bit and 3-bit quantization using YOLO5 and YOLO7 on COCO dataset. Both LSQ [9] and Oscillation dampening do not perform experiments on object detection and YOLO networks, so we re-implement their methods to create the baselines following their papers. The comparisons are done using the mAP metric and the results are reported in Table 3. Both our EMA and QC methods outperform LSQ consistently and the gap between LSQ and QC is significant

Table 4. *Comparison between per-channel quantization against our* QC *method. We train using* EMA *and* LSQ *at different bit-width on COCO dataset. Note, * denotes first and last layers are also trained at 4-bit quantization.*

Network	# Params	#-bits	LSQ + Ours (EMA)		
			Per-tensor	Per-channel	Ours (QC)
YOLO5-n	1.87M	4-bit	22.1	22.1	**23.8**
		3-bit	16.3	14.4	**18.2**
		4-bit*	16.5	19.4	**20.4**
YOLO5-s	7.23M	4-bit	33.1	32.6	**34.0**
		3-bit	28.5	27.3	**30.2**
		4-bit*	25.6	31.2	**32.0**
YOLO7-tiny	6.23M	4-bit	34.6	32.3	**35.2**
		3-bit	30.3	27.3	**31.0**
		4-bit*	32.8	30.3	**34.3**

Table 5. *Comparison with baseline method* i.e.*,* LSQ *[9] against our proposed methods using mAP metric for semantic segmentation task on the COCO dataset. * denotes first and last layers are trained at 4-bit quantization. Our methods consistently outperforms baseline methods on both YOLO5-n,s variants at 3-bit and 4-bit quantization.*

Method	#-bit	Mask (mAP)		Box (mAP)	
		YOLO5-n	YOLO5-s	YOLO5-n	YOLO5-s
Full-Precision	32-bit	23.4	31.7	27.6	37.6
Baseline	4-bit	16.5	27.8	18.5	32.1
Ours (EMA)		17.7	28.3	19.8	32.6
Ours (EMA+QC)		**19.5**	**29.4**	**22.3**	**33.7**
Baseline	3-bit	12.5	23.9	14.1	27.1
Ours (EMA)		13.9	24.7	15.9	27.8
Ours (EMA+QC)		**15.8**	**25.5**	**18.1**	**29.6**
Baseline	4-bit*	15.7	26.0	17.1	30.0
Ours (EMA)		16.5	26.7	17.9	30.4
Ours (EMA+QC)		**18.3**	**27.2**	**20.8**	**31.5**

on both YOLO5 and YOLO7 architectures with a margin of $\approx 2-3\%$ consistently. Our EMA models are either comparable or sometimes even better than Oscillation dampening, reflecting the efficacy of EMA in reducing the effect of oscillation, especially resulting from activation quantization as oscillation dampening does not account for oscillation issue in activation quantization. Furthermore, our QC method increases the gap ($\approx 2-3\%$) even further between our method and Oscillation dampening across both YOLO5 and YOLO7 variants at 3-bit and 4-bit quantization.

6.3. Comparison against per-channel quantization

As mentioned in Sec. 5.2, QC scale and shift factors can be folded either in the succeeding Batch Normalization (BN) layer after the convolution layer or into quantization scale factors by converting per-tensor quantization to per-channel quantization. It has been noted previously that per-channel quantization tends to be more unstable [24] for efficient networks with depth-wise convolutions due to a single scale factor being learnt for a depth-wise convolution filter (for eg. with size 3×3). Therefore, we further also provide experimental comparisons of our QC method against per-channel quantization to reflect the efficacy of our method in improving the stability of per-channel QAT by choosing QC as a post-hoc correction step after QAT. For this evaluation, we perform QAT with EMA using per-tensor and per-channel quantization. We perform QC only in case of per-tensor quantization and report the results in Table 4. First of all, as previous studies also noted, it can be observed that per-channel quantization with depth-wise convolutions can sometimes be inferior to per-tensor quantization. Furthermore, our QC method on per-tensor quantization consistently produces better performance on both YOLO5 and YOLO7 at 3-bit as well as 4-bit quantization with a margin of $\approx 3-4\%$ on YOLO7 and $\approx 2-4\%$ on YOLO5 variants.

6.4. Results on YOLO based semantic segmentation

We further also evaluate our methods to quantize YOLO5 variants at 3-bit and 4-bit on semantic segmentation task. We perform these experiments using COCO dataset and present results with mAP metric for the box and segmentation mask in Table 5. Similar to the observations in the object detection task, our QC method consistently improves the EMA method with a margin of $\approx 1-3\%$ across YOLO5-n and YOLO5-s variants quantized at 3-bit and 4-bit. Furthermore, QC in combination with EMA reduces the gap between full precision counterparts consistently on both detection box and segmentation mask metrics.

7. Discussion

In this work, we perform the first study for QAT on efficient real-time YOLO5, YOLO7 detectors and show that these networks suffer from oscillation issue. We further show that the oscillation issue does not only affect weight quantization but also activation quantization on YOLO models. To mitigate side-effects of oscillations due to STE approximation of rounding function and per-tensor quantization, we introduce two simple techniques, namely EMA and QC. Our proposed QAT pipeline combining EMA and QC produces a new state-of-the-art quantized YOLO models at low-bit precision (3-bits and 4-bits). In future work, we believe QC scale and shift factors can be generalized by estimating correction factors that are weighted for specific regions in the tensors that could potentially lead to even further performance gains.

References

[1] Thalaiyasingam Ajanthan, Kartik Gupta, Philip Torr, Richard Hartley, and Puneet Dokania. Mirror descent view for neural network quantization. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 2809–2817. PMLR, 13–15 Apr 2021. 2

[2] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Neural Information Processing Systems (NeuRIPS)*, 2019. 2

[3] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 3

[4] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun 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 (CVPR) Workshops*, June 2020. 3

[5] Peng Chen, Jing Liu, Bohan Zhuang, Mingkui Tan, and Chunhua Shen. Aqd: Towards accurate quantized object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 104–113, 2021. 2

[6] 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:805.06085*, 2018. 2

[7] Alexandre Défossez, Yossi Adi, and Gabriel Synnaeve. Differentiable model compression via pseudo quantization noise. *Transactions on Machine Learning Research*, 2022. 1, 2, 3

[8] Caiwen Ding, Shuo Wang, Ning Liu, Kaidi Xu, Yanzhi Wang, and Yun Liang. Req-yolo: A resource-aware, efficient quantization framework for object detection on fpgas. In *proceedings of the 2019 ACM/SIGDA international symposium on field-programmable gate arrays*, pages 33–42, 2019. 2

[9] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. *International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 3, 4, 6, 7, 8

[10] Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4851–4860, 2019. 1, 2

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 5

[12] Geoffrey Hinton. Neural networks for machine learning, lectures 15b. 2012. 3

[13] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018. 5

[14] B. Ham J. Lee, D. Kim. Network quantization with element-wise gradient scaling. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[15] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[16] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 1, 7

[17] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2810–2819, 2019. 2

[18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 4, 6

[20] Shih-Yang Liu, Zechun Liu, and Kwang-Ting Cheng. Oscillation-free quantization for low-bit vision transformers. *arXiv preprint arXiv:2302.02210*, 2023. 2

[21] Zechun Liu, Kwang-Ting Cheng, Dong Huang, Eric P Xing, and Zhiqiang Shen. Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4942–4952, 2022. 1, 2

[22] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 7197–7206. PMLR, 2020. 2

[23] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. 1, 2, 6

[24] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. 6, 8

[25] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *International Conference on*

Machine Learning, pages 16318–16330. PMLR, 2022. 1, 2, 3, 4, 5, 6, 7

[26] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. *International Conference on Computer Vision (ICCV)*, 2019. 2

[27] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 5

[28] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017. 5

[29] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: Fully convolutional one-stage object detection. In *Proc. Int. Conf. Computer Vision (ICCV)*, 2019. 2

[30] Ultralytics. YOLOv5: A state-of-the-art real-time object detection system. https://docs.ultralytics.com, 2021. 1, 2, 4

[31] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 4

[32] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2

[33] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017. 2

[34] Bohan Zhuang, Lingqiao Liu, Mingkui Tan, Chunhua Shen, and Ian Reid. Training quantized neural networks with a full-precision auxiliary module. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1488–1497, 2020. 2