RUPQ: Improving low-bit quantization by equalizing relative updates of quantization parameters

Valentin Buchnev
buchnev.valentin@huawei.com
Jiao He
hejiao4@huawei.com
Fengyu Sun
sunfengyu@huawei.com
Ivan Koryakovskiy
koryakovskiy.ivan1@huawei.com

Huawei Technologies Co. Ltd.

Abstract

Neural network quantization is a model compression technique using low-bit width representation of floating-point weights and activations. Although quantization can significantly reduce power consumption and inference time, it often leads to worse target metrics, such as accuracy or peak signal-to-noise ratio. To improve upon the target metrics, particularly at low-bit width computations, we propose a new Relative Update-Preserving Quantizer. This quantizer stabilizes relative updates for all parameters during training. Our experimental results show that the proposed quantizer performs consistently better than LSQ+ and, compared to full-precision models, significantly reduces the gap in quality for low-bit SRResNet, EDSR and YOLO-v3 models. The method is easy to implement and use in practice. The code is available at https://github.com/Valentin-Buchnev/RUPQ.

1 Introduction

Deep neural networks successfully solve a wide range of tasks. However, large models often do not fit well into resource-constrained edge devices. To satisfy the power, memory, and inference time constraints, neural networks can be compressed by different techniques, including pruning [8, 9], knowledge distillation (KD) [11], neural architecture search (NAS) [6] and quantization [7, 27]. In this work, we focus on quantization, which is an effective approach for model compression for various deep learning tasks: image classification [7], segmentation [32], object detection [20], video classification [36] and super-resolution (SR) [17]. The quality of quantized models depends on the chosen bit width: lower bit width provides better compression but leads to quality degradation [4, 16]. Therefore, the task of improving the quality of low-bit models is of great interest to researchers.

The work of You [34] shows that good quality is achieved in networks where the ratio of average update magnitude to average parameter magnitude is approximately the same for all parameters. To keep this ratio close to each other for all parameters, LSQ+ [3] proposes to use an additional gradient scaling factor applied to step gradients. In this work, we analyse the behavior of relative updates for quantization steps in LSQ+. We show that this quantity is not stable during training. To make it stable, we propose a novel Relative Update-Preserving Quantization (RUPQ) method. The proposed method uses the Adam [15] optimizer for training quantization steps normalized by the standard deviation of the quantized tensor. A similar normalization technique was introduced in previous works on quantization [12, 19], but to the best of our knowledge, we are the first to explain the importance of such a normalization in terms of relative update preservation. We also lower a weight step learning rate for the networks where the majority of quantized layers are followed by Batch Normalization (BN) layers. We use theoretical derivation to prove the lower learning rate.

To sum up, our contribution is threefold.

- We provide the analysis of relative updates for the current state-of-the-art (SOTA) quantization method, LSQ+.
- We propose a new RUPQ method and show that relative updates are more stable during training compared to LSQ+.
- We achieve new SOTA results with the proposed quantizer for image classification (ResNet-18 [10] and MobileNet-v2 [30]), SR (SRResNet [17] and EDSR [22]) and object detection (YOLO-v3 [28]) networks.

2 Related Work

For uniform quantization, the quantization function can be configured by a single step parameter, the distance between each two adjacent quantized values. Early works on quantization [5, 37] restricted the weights to the $[0, 1]$ range with some transformation function and used a constant quantization step during training. Later works [3, 7, 14] proposed to quantize weights and input activations with a trainable step and did not apply any transformations, but some works [3, 7] additionally used a gradient scaling factor for better training of quantization steps. This approach has proven its efficiency, and the LSQ+ algorithm has remained a SOTA method for the past several years. Recent works on quantization [12, 16] proposed to first normalize the data on its standard deviation and then quantize it with a trainable quantization step. This normalization does not change the family of quantization functions which can be learned with a trainable step parameter. These works did not explain how data normalization helps to achieve better model quality.

Recent studies focused more on task- and model-specific quantization algorithms [18, 24, 25, 33]. Some works also incorporated additional losses to make the model more robust to quantization [26, 31]. In this study, we consider a generic approach orthogonal to ideas proposed in these articles.

3 Background

In this work, we use the quantization method LSQ+ proposed in [3]. It is defined as follows:

$$\hat{\mathbf{v}} = \left\lfloor \text{clamp}\left(\frac{\mathbf{v}-z}{s}, Q_N, Q_P\right)\right\rceil s, \tag{1}$$

where $\text{clamp}(\cdot, Q_N, Q_P)$ is a clamping function with lower and upper bounds Q_N and Q_P, respectively, and $\lfloor\cdot\rceil$ is a round-to-nearest function. s and z correspond to the quantization step and offset, respectively, and $\hat{\mathbf{v}}$ denotes the quantized value of tensor $\mathbf{v}$. We use $\mathbf{v}$ as a placeholder for a specific tensor of weights $\mathbf{w}$ or input activations $\mathbf{x}$ whenever it needs a clarification. For unsigned data (*e.g.*, rectified input activations $\mathbf{x}$ to the layer), $Q_N = 0$, $Q_P = 2^b - 1$, and for signed data (*e.g.*, layer's weights $\mathbf{w}$), $Q_N = -2^{b-1}$, $Q_P = 2^{b-1} - 1$. Asymmetric quantization ($z \neq 0$) is applied only to input activation quantization since it has been shown [3] that asymmetric quantization for weights incorporates additional overhead operations at inference time. The quantization steps for weights and input activations are denoted by s_w and s_x, respectively. To approximate the gradient of a round function, we use the straight-through estimator (STE) technique [2].

Let us consider a quantizable layer, which is defined as a tensor multiplication of weights $\mathbf{w}$ and input activations $\mathbf{x}$ shifted by a bias b:

$$\mathbf{y} = \mathbf{w}\mathbf{x} + b. \tag{2}$$

We define the relative update of a trainable parameter $\mathbf{v}$ in a particular optimization step as the ratio between a l_2-norm of the parameter update $\Delta\mathbf{v}$ of a gradient descent and a l_2-norm of the parameter itself:

$$\delta(\mathbf{v}) = \frac{\|\Delta\mathbf{v}\|_2}{\|\mathbf{v}\|_2}. \tag{3}$$

The LARS [34] optimization method improves training stability by making each parameter update $\Delta\mathbf{v}_{\text{LARS}}$ proportional to the magnitude of an updated parameter by the rule

$$\|\Delta\mathbf{v}_{\text{LARS}}\|_2 = \left\|\eta\frac{\|\mathbf{v}\|_2}{\|\nabla_{\mathbf{v}}L\|_2}\nabla_{\mathbf{v}}L\right\|_2 = \eta\,\|\mathbf{v}\|_2, \tag{4}$$

where η is the learning rate.

We define relative updates divided by learning rate η for the layer's weights, input activations and quantization steps using the following equations:

$$r_{\mathbf{w}} = \frac{\delta(\mathbf{w})}{\eta}, \quad r_{\mathbf{x}} = \frac{\delta(\mathbf{x})}{\eta}, \tag{5}$$

$$\rho_{\mathbf{w}} = \frac{\delta(s_{\mathbf{w}})}{\eta}, \quad \rho_{\mathbf{x}} = \frac{\delta(s_{\mathbf{x}})}{\eta}, \tag{6}$$

We define the value $r_{\mathbf{x}}$ as if an input activation $\mathbf{x}$ would be trainable with the same optimizer as for $s_{\mathbf{x}}$.

As shown in [34], for a better training, the relative updates for all trainable parameters of neural network should be similar in magnitude. This is equivalent to the following condition:

$$r_{\mathbf{w}} \approx \rho_{\mathbf{w}} \approx \rho_{\mathbf{x}}. \tag{7}$$

The work of Esser [2] proposes a gradient scaling factor for quantization steps to equalize relative updates $r_{\mathbf{w}} \approx \rho_{\mathbf{w}}$ and $r_{\mathbf{x}} \approx \rho_{\mathbf{x}}$:

$$\rho_{\mathbf{v}} \; / = \sqrt{n_{\mathbf{v}} Q_P}. \tag{8}$$

However, this scaling factor does not guarantee the fulfilment of condition (7) since the quantities $r_{\mathbf{w}}$ and $r_{\mathbf{x}}$ can be very different (see Fig. 1).

4 Method

4.1 Relative updates for trainable parameters

We define the relative updates for SGD and Adam optimizers as follows:

$$\delta_{\text{SGD}}(\mathbf{v}) = \eta \frac{\|\hat{g}_{\mathbf{v}}\|_2}{\|\mathbf{v}\|_2}, \tag{9}$$

$$\delta_{\text{Adam}}(\mathbf{v}) = \eta \frac{\left\| \frac{\hat{g}_{\mathbf{v}}}{\sqrt{\hat{u}_{\mathbf{v}}}+\varepsilon} \right\|_2}{\|\mathbf{v}\|_2} \approx \eta \frac{\sqrt{n_{\mathbf{v}}}}{\|\mathbf{v}\|_2}, \tag{10}$$

where $\hat{g}_{\mathbf{v}}$ is the exponential moving average (EMA) of the gradient $\nabla_{\mathbf{v}} L$, $\hat{u}_{\mathbf{v}}$ is the EMA of the squared gradient $(\nabla_{\mathbf{v}} L)^2$, and $n_{\mathbf{v}}$ is the size of tensor $\mathbf{v}$. The final formula of the relative update depends on the optimizer choise and is defined as

$$\delta(\mathbf{v}) = \begin{cases} \eta \frac{\|\hat{g}_{\mathbf{v}}\|_2}{\|\mathbf{v}\|_2}, & \text{if optimizer is SGD} \\ \eta \frac{\sqrt{n_{\mathbf{v}}}}{\|\mathbf{v}\|_2}, & \text{if optimizer is Adam} \end{cases} \tag{11}$$

Note that l_2-norm is proportional to $\sqrt{n_{\mathbf{v}}}$, and the condition (7) is true only if all trainable parameters optimized by Adam have the same scale.

4.2 Comparison of relative updates for quantization steps $\rho_{\mathbf{w}}$ and $\rho_{\mathbf{x}}$

We begin our analysis by comparing relative updates (5) and (6) for trainable parameters for LSQ+. Figure 1 shows that $\rho_{\mathbf{w}}$ and $\rho_{\mathbf{x}}$ have similar magnitudes as the corresponding counterparts for quantized data, $r_{\mathbf{w}}$ and $r_{\mathbf{x}}$, respectively, but differ a lot compared to each other. To mitigate this discrepancy, we propose using the Adam optimizer for quantization steps training since Adam normalizes gradients during training and makes relative update independent from the gradient scale.

4.3 Variability of relative updates for quantization steps

The changes of the term $\|s_{\mathbf{v}}\|_2$ during training affect relative update (6) and can violate conditions (7). As shown in Figure 2, quantization steps $s_{\mathbf{w}}$ and $s_{\mathbf{x}}$ are changing during training, and these changes correlate with changes in the standard deviation of quantized tensors $\mathbf{w}$ and $\mathbf{x}$. As a result, the steps must be adjusted during training. To remove this dependency from the scale of quantized data, we propose scaling data on its standard deviation before quantization and rescaling it back after:

$$\hat{\mathbf{v}} = \left\lfloor \text{clamp}\left(\frac{\mathbf{v}-z}{s\sigma_{\mathbf{v}}}, Q_N, Q_P \right) \right\rceil s\sigma_{\mathbf{v}}. \tag{12}$$

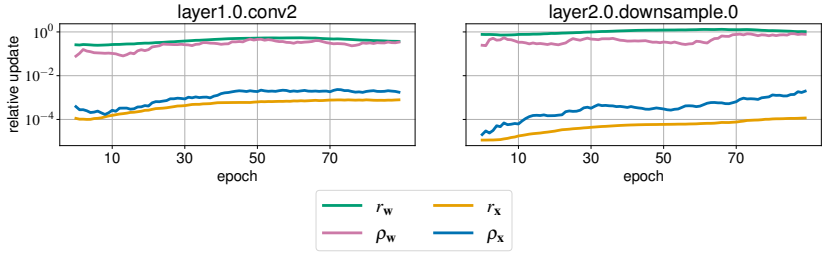


Figure 1: Relative updates (5) and (6) for W2A2 ResNet-18 on ImageNet dataset and LSQ+ quantization method. Theoretical conclusions from [7] show that $\rho_{\mathbf{w}}$ and $\rho_{\mathbf{x}}$ have magnitudes similar to $r_{\mathbf{w}}$ and $r_{\mathbf{x}}$, respectively. In this experiment, we see that $\rho_{\mathbf{w}}$ is indeed similar to $r_{\mathbf{w}}$, while $\rho_{\mathbf{x}}$ is slightly larger than $r_{\mathbf{x}}$. However, article [34] shows that all three relative updates – $r_{\mathbf{w}}$, $\rho_{\mathbf{w}}$, $\rho_{\mathbf{x}}$ – should be similar for better training.

While the standard deviation for the tensor of weights $\sigma_{\mathbf{w}}$ can be determined, the same variable for input activations $\sigma_{\mathbf{x}}$ is stochastic and should be somehow approximated during training. We propose using EMA since this is a popular way for building a robust estimation of a stochastic variable. The estimation $\sigma_{\mathbf{x}}^t$ is updated on each training step t by the following formula:

$$\sigma_{\mathbf{x}}^t = m \cdot \sigma_{\mathbf{x}}^{t-1} + (1-m) \cdot \sigma_{\mathbf{x}}, \tag{13}$$

where m is the momentum for EMA.

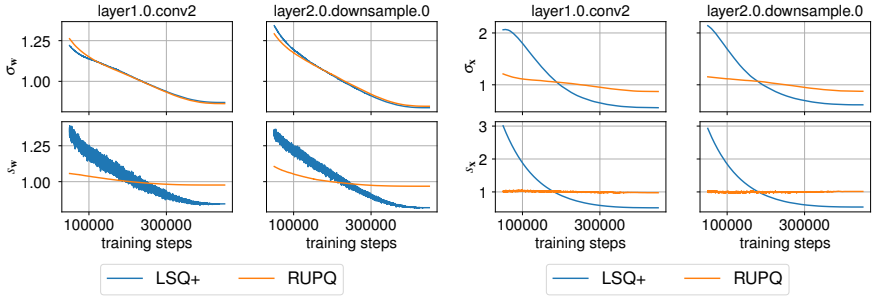
The work of Esser [7] demonstrates that quantization step minimizing quantization error $s_{\text{error}} = \arg\min_s \|\mathbf{v} - \hat{\mathbf{v}}(s)\|$ is not equal to the one which minimizes a model task loss $s_{\text{loss}} = \arg\min_s E_{\mathbf{x}} L(\mathbf{W}, \mathbf{x})$. Despite this, we still can assume that these two steps are similar to each other. If we suppose that the data to quantize follows a distribution $f(v)$ parametrized by a scale parameter σ, the step s_{error} equals to $\arg\min_s \int_{-\infty}^{\infty} (v - \hat{v}(s))^2 f(v) dv = c\sigma$, where c is some constant value. Figure 3 shows that data normalized by its standard deviation makes both weights and input activations steps independent of the actual scale of the data, so in RUPQ the steps do not change as much as in LSQ+. The relative updates for quantization steps $\rho_{\mathbf{w}}$ and $\rho_{\mathbf{x}}$ are not stable during training with LSQ+, whereas for our proposed method, RUPQ, the same relative updates are changing much less.

4.4 Weight step training in the network with batch normalization layers

Consider the model where the quantized layer is followed by a BN [13] layer:

$$\mathbf{y} = \text{BN}(\hat{\mathbf{w}}\hat{\mathbf{x}} + b) = \left(\frac{\left\lfloor \text{clamp}\left(\frac{\mathbf{w}}{s_{\mathbf{w}}}, Q_N, Q_P\right) \right\rceil \hat{\mathbf{x}}}{\sqrt{\sigma^2 + \varepsilon}} \cdot \gamma s_{\mathbf{w}} \right) + \left(\frac{b - \mu}{\sqrt{\sigma^2 + \varepsilon}} \gamma + \beta \right). \tag{14}$$

Here, μ and σ^2 denote BN statistics, running mean and variance, respectively. Variables γ and β denote trainable BN parameters.



(a) Standard deviation $\sigma_{\mathbf{w}}$ and quantization step $s_{\mathbf{w}}$ for weights

(b) Standard deviation $\sigma_{\mathbf{x}}$ and quantization step $s_{\mathbf{x}}$ for input activations

Figure 2: Standard deviation and quantization steps for the weights (a) and input activations (b) for W2A2 ResNet-18 on ImageNet dataset. All quantities are rescaled to have an average value equal to one for a better comparison. As shown, the values of quantization steps strongly correlate with the standard deviation of quantized data. Pearson correlation coefficitients for s and σ are in the range $0.98 - 1.00$ for different layers for LSQ+.

The quantization step $s_{\mathbf{w}}$ takes part in formula (14) in two places. First, the weights are divided by a quantization step $s_{\mathbf{w}}$. These weights are trainable and therefore can adapt to the given step during the training process. Second, the quantized weights are multiplied by $s_{\mathbf{w}}$ back to return it to initial scale. Note that these quantized weights are also multiplied by a trainiable parameter γ. We observe that for the cases where the majority of quantized layers are followed by BN layers, reducing a learning rate for weight steps helps achieving better quality. We think this happens due to the multiplication of the trainable weight step by another trainable parameters. Positively, lower learning rate reduces the stochasticity of the optimization process.

4.5 RUPQ method

Summing up, RUPQ has the following differences compared to the original LSQ+:

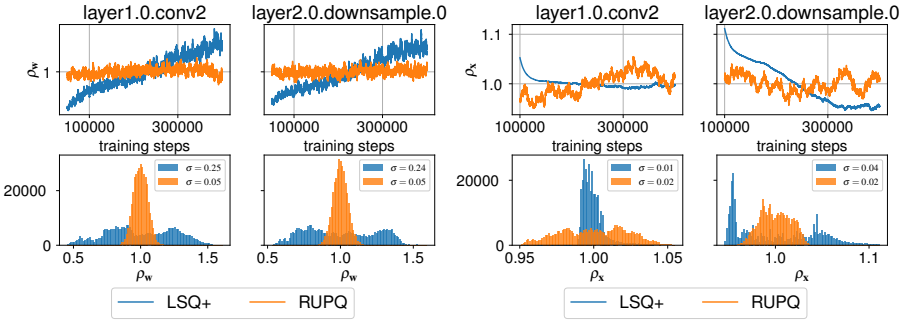
1. Weights are scaled on $\sigma_{\mathbf{w}}$ before quantization and rescaled back after.
2. Input activations are scaled on $\sigma_{\mathbf{x}}$ before quantization and rescaled back after. The EMA is applied to approximate $\sigma_{\mathbf{x}}$ during training.
3. The Adam optimizer is applied to optimize quantization steps.
4. The lower learning rate is applied to train weight steps in the networks with BN layers.

For clarity, we provide the pseudocode of our quantization algorithm in Algorithm 1.

5 Experiments

5.1 Experimental setup

We evaluate the effectiveness of our method by quantizing the image classification networks, ResNet-18 and MobileNet-v2, SR networks, SRResNet (scale x2) and EDSR (scale x4), and



(a) Relative update for weight step $\rho_{\mathbf{w}}$

(b) Relative update for input step $\rho_{\mathbf{x}}$

Figure 3: The comparison of relative updates for quantization steps $\rho_{\mathbf{w}}$ (left) and $\rho_{\mathbf{x}}$ (right) on W2A2 ResNet-18 for two quantization methods: LSQ+ and RUPQ. For a better comparison, all quantities are smoothed out by EMA and then rescaled to have an average value equal to one. As shown, when the RUPQ method is applied, the relative updates are changing less during training. The corresponding standard deviation σ for relative updates $\rho_{\mathbf{w}}$ and $\rho_{\mathbf{x}}$ for LSQ+ and RUPQ is shown in the legend.

object detection YOLO-v3 network. Note that the only network without BN layers is EDSR, and the quantization settings for this model differ from the others. We focus on low-bit quantization: 4, 3, and 2 bit widths. The models for image classification task are trained and evaluated on ImageNet [29], the models for SR task are trained on the DIV2K [1] dataset and evaluated on the Set14 dataset [35], and object detection YOLO-v3 model is trained and evaluated on COCO [23] dataset.

In all the experiments, the weights of the quantized models are initialized with the pre-trained full-precision (FP) weights, and dataset preprocessing is kept the same as in the original papers. FP models are trained with the same configuration as in the original papers with batch size and epoch number listed in Table 1, and the weights of quantized models are optimized with the same configuration as in the FP model training. The learning rate is annealed by a cosine decay in all the experiments.

The training hyperparameters are provided in Table 1. We select a learning rate for quantization steps that gives the best achievable quality for quantized model. We choose the EMA momentum $m = 0.9999$ for all the experiments. For the networks with BN layers, we find the learning rate for input steps equal to 10^{-3} and the lower learning rate for weight steps to be optimal. For the networks without BN layers (EDSR), we find that the learning rate for quantization steps equal to 10^{-4}, the same as for the weights, is optimal for training. The results of hyperparameters searching for SRResNet are reported in Sec. 5.3.

In all the experiments, we keep the first layer of the networks in the 8-bit, which is standard practice, since this layer is very sensitive to quantization. For SR networks, we additionally keep the last layer in the 8-bit since this layer is producing output images. We use per-channel quantization, as it is known that this improves the quality of the quantized model [12, 21]. Bit width setting is denoted in the WaAb format, where a and b refer to the numbers of bits used for weights and input activations quantization, respectively.

For MobileNet-v2, the BatchNorm re-estimation technique proposed in [24] is applied at the end of the training since quantization causes weight oscillations during training, which is corrupting the estimated BN statistics [24].

Algorithm 1 RUPQ

procedure QUANTIZE($\mathbf{v}, s, Q_N, Q_P$)
  Inputs: Tensor $\mathbf{v}$, quantization step s, quantization thresholds Q_N and Q_P
  $\mathbf{v} = \mathbf{v}/s$
  $\mathbf{v} = \text{clip}(\mathbf{v}, Q_N, Q_P)$
  $\mathbf{v} = \text{detach}(\text{round}(\mathbf{v}) - \mathbf{v}) + \mathbf{v}$ ▷ Apply STE
  $\mathbf{v} = \mathbf{v} \times s$
  return $\mathbf{v}$
end procedure

procedure QUANTIZELAYER($\mathbf{x}, \mathbf{w}$)
  Inputs: Input activations $\mathbf{x}$, layer's weights $\mathbf{w}$
  Init: $\sigma_{\mathbf{x}}^0 = \sigma_{\mathbf{x}}$ on first batch from dataset
  $\sigma_{\mathbf{x}}^t = m \cdot \sigma_{\mathbf{x}}^{t-1} + (1-m) \cdot \sigma_{\mathbf{x}}$ ▷ Update $\sigma_{\mathbf{x}}^t$
  $\hat{\mathbf{x}} = \text{QUANTIZE}(\mathbf{x}, s_{\mathbf{x}} \cdot \sigma_{\mathbf{x}}^t, Q_N^x, Q_P^x)$ ▷ Quantize input activations
  $\hat{\mathbf{w}} = \text{QUANTIZE}(\mathbf{w}, s_{\mathbf{w}} \cdot \sigma_{\mathbf{w}}, Q_N^w, Q_P^w)$ ▷ Quantize weights
  return $\hat{\mathbf{x}}, \hat{\mathbf{w}}$
end procedure

Model	Method	Optimizer		Initial learning rate			Weight decay		Batch size	Epochs
		W	s_w, s_x, z_x	**W**	s_w*	s_x*, z_x*	**W**	s_w, s_x, z_x		
ResNet-18	LSQ+	SGD	SGD	10^{-2}	10^{-2}	10^{-2}	10^{-4}	0.0	512	90
	RUPQ		Adam		10^{-5}	10^{-3}				
MobileNet-v2	LSQ+	SGD	SGD	10^{-2}	10^{-2}	10^{-2}	5×10^{-4}	0.0	256	90
	RUPQ		Adam		10^{-4}	10^{-3}				
SRResNet	LSQ+	Adam	Adam	10^{-3}	10^{-3}	10^{-3}	0.0		32	7200
	RUPQ				10^{-6}	10^{-3}				
EDSR	LSQ+	Adam	Adam	10^{-4}	10^{-4}	10^{-4}	0.0		16	6000
	RUPQ				10^{-4}	10^{-4}				
YOLO-v3	LSQ+	SGD	SGD	10^{-2}	10^{-2}	10^{-2}	5×10^{-4}	0.0	64	300
	RUPQ		Adam		10^{-5}	10^{-3}				

Table 1: Hyperparameters for LSQ+ and RUPQ algorithms. Symbol * refers to the hyperparameters selected specifically for RUPQ.

5.2 Results

We evaluate the effectiveness of our method by comparing the quality of the image classification, SR and object detection models quantized with LSQ+ and RUPQ in W4A4, W3A3, and W2A2 bit widths. Results are reported in Table 2. For image classification networks, the improvement is marginal, except for W2A2 MobileNet-v2, where the results are better by 0.9 % compared to LSQ+. For SR networks, RUPQ allows us to significantly reduce the gap in quality compared to the FP model. For the EDSR model, $\sigma_{\mathbf{x}}$ normalization is not profitable and only worsens the quality. For YOLO-v3 model, the models quantized with LSQ+ converge only for per-tensor W4A4 or W3A3 quantization setting, whereas RUPQ diverges only in the most extreme case, when per-channel W2A2 quantization is applied. For a W3A3 quantization, our method outperforms LSQ+ by a large margin of 4.4 AP_{50}. The proposed RUPQ method shows results that are consistently better than LSQ+ for all the models.

Model	Method	Model Quality		
		W4A4	W3A3	W2A2
ResNet-18 FP: 70.4 %	LSQ+	**70.5**	69.1	65.2
	RUPQ	**70.5**	**69.3**	**65.4**
MobileNet-V2 FP: 71.6 %	LSQ+	70.5	66.7	53.5
	RUPQ	**70.6**	**66.9**	**54.4**
SRResNet FP: 28.34 dB	LSQ+	28.25	28.07	27.73
	RUPQ	**28.31**	**28.21**	**27.97**
EDSR FP: 33.46 dB	LSQ+	33.29	33.06	**32.55**
	RUPQ	33.30	32.87	32.25
	RUPQ w/o $\sigma_{\mathbf{x}}$	**33.33**	**33.08**	**32.55**
YOLO-v3 FP: 56.3 AP_{50}	LSQ+ per-tensor	52.7	47.9	diverged
	LSQ+ per-channel	diverged	diverged	diverged
	RUPQ per-tensor	54.3	51.0	**46.2**
	RUPQ per-channel	**54.5**	**52.3**	diverged

Table 2: Quantized neural network validation results. For ResNet-18 and MobileNet-v2, the model quality is measured as top-1 accuracy on ImageNet-2012 validation. For SRResNet and EDSR, the model quality is measured as peak signal-to-noise ratio (PSNR) on the Set-14 dataset. For YOLO-v3, the model quality is measured as average precision for IOU threshold $= 0.5$ (AP_{50}) on COCO validation. Bold indicates the best result.

5.3 Ablation studies

Table 3 shows the influence of each modification from Sec. 4.5 on the final quality of W2A2 SRResNet. This study shows that every feature brings a positive impact on the quantized model quality:

1. The motivation of lowering the learning rate for weight step optimization is described in Sec. 4.4. This feature explains 30 % of the increase in model quality compared to the baseline LSQ+ method.

2. Weight scaling brings the main contribution to the method improvement. According to the results, 60 % of the increase in quality is explained by this technique.

3. Input activation scaling brings a very slight but still statistically significant improvement. According to the results, 10 % of the increase in quality is explained by this technique. This technique is less efficient since the standard deviation $\sigma_{\mathbf{x}}$ is stochastic and difficult to approximate.

In Table 4, we present the importance of hyperparameter m from the formula (13) and learning rate for $s_{\mathbf{w}}$. This hyperparameter search proves the conclusions from Sec. 4.4 since the results for a fixed momentum become better with a lower learning rate. The choice of momentum m slightly affects the model quality, and the value $m = 0.9999$ is optimal.

6 Discussion

The networks with and without batch normalization layers require different approaches during quantization. As experiments show, a reduced learning rate for weight step and input

Model	Modifications			PSNR
	σ_w scaling	σ_x scaling	LR reduction for s_w from Table 1	
SRResNet FP: 28.34 dB W2A2	-	-	-	27.73
	-	-	✓	27.79
	✓	-	✓	27.94
	-	✓	-	27.73
	-	✓	✓	27.80
	✓	✓	✓	**27.97**

Table 3: Ablation study on proposed modifications importance for W2A2 SRResNet. The model quality is measured as PSNR on Set14. The confidence intervals for a p-value of 0.95 is equal to 0.01 dB.

σ_x EMA momentum	larning rate for s_w			
	10^{-4}	10^{-5}	10^{-6}	10^{-7}
0.999	27.91	27.96	27.93	27.96
0.9999	27.95	27.95	**27.97**	**27.97**
0.99999	27.93	27.93	27.93	27.96

Table 4: Ablation study on hyperparameters choice for W2A2 SRResNet. The results in each row correspond to a fixed momentum m, and results in each column correspond to a fixed learning rate for s_w optimization. The model quality is measured as PSNR on Set14.

activation normalization help in achieving better quality for the networks with BN layers, while for the networks without BN, these techniques are not useful since it is hard to adequately approximate σ_x for them. Nevertheless, the usage of normalization only for the weights still brings an improvement compared to LSQ+. We recommend to carefully use σ_x normalization since, as results of the experiments show, this technique can worsen the quality of the quantized model when the BN layers are absent.

The work of [34] shows that LARS optimization method improves the training stability. Our experiment results prove this statement and show that equalizing relative updates for the training parameters indeed improves the training stability. For YOLO-v3 model and for some quantization settings, LSQ+ diverges whereas RUPQ works stable and allows to achieve much better quality compared to LSQ+.

7 Conclusion

In this work, we show that the relative update for quantization steps is unstable during training with the LSQ+ method. We propose tackling this problem by using the Adam optimizer and data normalization. We show that our quantization method has more stable relative updates for quantization steps than LSQ+. The experiments demonstrate that the proposed method, RUPQ, works consistently better than LSQ+ for low-bit quantization and sets the new SOTA results. The results of the experiments also prove the hypothesis that the stability of relative updates for quantization steps affects the model quality.

References

[1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *arXiv:1308.3432*, 2013.

[3] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. LSQ+: Improving Low-Bit Quantization Through Learnable Offsets and Better Initialization. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

[4] Zhaowei Cai and Nuno Vasconcelos. Rethinking Differentiable Search for Mixed-Precision Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[5] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized Clipping Activation for Quantized Neural Networks. *arXiv:1805.06085*, 2018.

[6] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural Architecture Search: A Survey. *The Journal of Machine Learning Research*, 2019.

[7] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned Step Size Quantization. In *International Conference on Learning Representations*, 2020.

[8] Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *arXiv:1803.03635*, 2018.

[9] Song Han, Huizi Mao, and William J Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv:1510.00149*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[11] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. *arXiv:1503.02531*, 2015.

[12] Cheeun Hong, Heewon Kim, Sungyong Baik, Junghun Oh, and Kyoung Mu Lee. DAQ: Channel-Wise Distribution-Aware Quantization for Deep Image Super-Resolution Networks. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022.

[13] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*, 2015.

[14] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[15] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014.

[16] Ivan Koryakovskiy, Alexandra Yakovleva, Valentin Buchnev, Temur Isaev, and Gleb Odinokikh. One-Shot Model for Mixed-Precision Quantization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2023.

[17] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and others. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[18] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully Quantized Network for Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[19] Yuhang Li, Xin Dong, and Wei Wang. Additive Powers-of-Two Quantization: An Efficient Non-Uniform Discretization for Neural Networks. *arXiv:1909.13144*, 2019.

[20] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. BRECQ: Pushing the Limit of Post-Training Quantization by Block Reconstruction. *arXiv:2102.05426*, 2021.

[21] Yuhang Li, Mingzhu Shen, Jian Ma, Yan Ren, Mingxin Zhao, Qi Zhang, Ruihao Gong, Fengwei Yu, and Junjie Yan. MQBench: Towards Reproducible and Deployable Model Quantization Benchmark. *arXiv:2111.03759*, 2021.

[22] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced Deep Residual Networks for Single Image Super-Resolution. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2017.

[23] 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 *European Conference on Computer Vision*, 2014.

[24] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming Oscillations in Quantization-Aware Training. *arXiv:2203.11086*, 2022.

[25] Eunhyeok Park and Sungjoo Yoo. PROFIT: A Novel Training Method for Sub-4-Bit Mobilenet Models. In *European Conference on Computer Vision*, 2020.

[26] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model Compression Via Distillation and Quantization. *arXiv:1802.05668*, 2018.

[27] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet Classification Using Binary Convolutional Neural Networks. In *European Conference on Computer Vision*, 2016.

[28] Joseph Redmon and Ali Farhadi. Yolov3: An Incremental Improvement. *arXiv:1804.02767*, 2018.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, and others. Imagenet Large Scale Visual Recognition Challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[31] Yuzhang Shang, Dan Xu, Bin Duan, Ziliang Zong, Liqiang Nie, and Yan Yan. Lipschitz Continuity Retained Binary Neural Network. In *European Conference on Computer Vision*, 2022.

[32] George K Thiruvathukal, Yung-Hsiang Lu, Jaeyoun Kim, Yiran Chen, and Bo Chen. *Low-Power Computer Vision: Improve the Efficiency of Artificial Intelligence*. CRC Press, 2022.

[33] Hu Wang, Peng Chen, Bohan Zhuang, and Chunhua Shen. Fully Quantized Image Super-Resolution Networks. In *ACM International Conference on Multimedia*, 2021.

[34] Yang You, Igor Gitman, and Boris Ginsburg. Large Batch Training of Convolutional Networks. *arXiv:1708.03888*, 2017.

[35] Roman Zeyde, Michael Elad, and Matan Protter. On Single Image Scale-up Using Sparse-Representations. In *Curves Surfaces*, 2012.

[36] Kang Zhao, Sida Huang, Pan Pan, Yinghan Li, Yingya Zhang, Zhenyu Gu, and Yinghui Xu. Distribution Adaptive Int8 Quantization for Training Cnns. In *AAAI Conference on Artificial Intelligence*, 2021.

[37] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. DorefaNet: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *arXiv:1606.06160*, 2016.