LSQ+: Improving low-bit quantization through learnable offsets and better initialization

Yash Bhalgat[1] Jinwon Lee[1] Markus Nagel[2] Tijmen Blankevoort[2] Nojun Kwak[3†]

[1]Qualcomm AI Research, Qualcomm Technologies, Inc.
[2]Qualcomm AI Research, Qualcomm Technologies Netherlands B.V.
[3]Seoul National University

{ybhalgat, jinwonl, markusn, tijmen}@qti.qualcomm.com, nojunk@snu.ac.kr

Abstract

Unlike ReLU, newer activation functions (like Swish, H-swish, Mish) that are frequently employed in popular efficient architectures can also result in negative activation values, with skewed positive and negative ranges. Typical learnable quantization schemes [5, 7] assume unsigned quantization for activations and quantize all negative activations to zero which leads to significant loss in performance. Naively using signed quantization to accommodate these negative values requires an extra sign bit which is expensive for low-bit (2-, 3-, 4-bit) quantization. To solve this problem, we propose LSQ+, a natural extension of LSQ [7], wherein we introduce a general asymmetric quantization scheme with trainable scale and offset parameters that can learn to accommodate the negative activations. Gradient-based learnable quantization schemes also commonly suffer from high instability or variance in the final training performance, hence requiring a great deal of hyper-parameter tuning to reach a satisfactory performance. LSQ+ alleviates this problem by using an MSE-based initialization scheme for the quantization parameters. We show that this initialization leads to significantly lower variance in final performance across multiple training runs. Overall, LSQ+ shows state-of-the-art results for EfficientNet and MixNet and also significantly outperforms LSQ for low-bit quantization of neural nets with Swish activations (e.g.: 1.8% gain with W4A4 quantization and upto 5.6% gain with W2A2 quantization of EfficientNet-B0 on ImageNet dataset). To the best of our knowledge, ours is the first work to quantize such architectures to extremely low bit-widths.

1. Introduction

With the popularity of deep neural networks across various use-cases, there is now an increasing demand for methods that make deep networks run efficiently on resource-constrained edge-devices. These methods include model pruning, neural architecture search (NAS) and hand-crafted efficient networks made out of novel architectural blocks (e.g. depth-wise separable or group convolutions, squeeze-excite blocks, etc.). Finally we can also perform model quantization, where the weights and activations are quantized to lower bit-widths allowing efficient fixed-point inference and reduced memory bandwidth usage.

Due to the surge in more efficient architectures found with NAS, newer and more general activation functions (like Swish [20], H-swish [10], Leaky-ReLU) are replacing the traditional ReLU. Unlike ReLU, these activation functions also take over values below zero. Current state-of-the-art quantization schemes like PACT [5] and LSQ [7] assume unsigned quantization ranges for activation quantization where all the activation values below zero are discarded by quantizing them to zero. This works well for traditional ReLU-based architectures like ResNet [9], but leads to a significant loss of information when applied to modern architectures like EfficientNet [24] and MixNet [25], which employ Swish activations. For example, LSQ achieves W4A4 quantization of preactivation-ResNet50 with no loss in acccuracy but leads to a 4.1% loss in accuracy when quantizing EfficientNet-B0 to W4A4[1]. Naively using a signed quantization range to accommodate these negative values also results in a drop in performance.

To alleviate these drops in performance which are commonly observed with very low-bit (2-, 3-, 4-bit) quantization, we propose using a general asymmetric quantization

[†] Currently a Visiting Researcher at Qualcomm Technologies, Inc.

Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

[1] WxAx quantization indicates quantizing the weights and output activations of all layers to x bits

scheme with a learnable offset parameter as well as a learnable scale parameter. We show that the proposed quantization scheme learns to accommodate the negative activation values differently for different layers and recovers the accuracy loss incurred by LSQ, e.g. 1.8% accuracy improvement over LSQ with W4A4 quantization and upto 5.6% improvement with W2A2 quantization on EfficientNet-B0. To the best of our knowledge, ours is the first work to quantize modern architectures like EfficientNet and MixNet to extremely low bit-widths.

Another problem faced especially by any *gradient-based* learnable quantization scheme is its sensitivity to initialization, meaning that a poor initialization can lead to a high variance in final performance across multiple training runs. This problem is especially observed with min-max initialization (used in [1]). We show that using an initialization scheme based on mean-squared-error (MSE) minimization [22, 23] for the offset and scale parameters leads to significantly higher stability in final performance than min-max quantization. We also compare this initialization scheme with the one proposed in [7].

In summary, our proposed method, called LSQ+, extends LSQ [7] by adding a simple yet effective learnable offset parameter for activation quantization to recover the lost accuracy on architectures employing *Swish-like* activations. Furthermore, our other contribution is showing the importance of proper initialization for stable training, especially in the low-bit regime.

2. Related Work

A good overview of the basics of quantization is given in [15], where the differences between asymmetric and symmetric quantization are explained. In general, we can classify quantization methods into *post-training methods* that work without fine-tuning and *quantization-aware training methods* that need fine-tuning.

Post-training quantization methods [2, 29, 6] optimize neural networks for quantization without full training and using a little amount of data. [18, 4] do this better without using any data at all. Although these methods work well on typical 8-bit quantization, they were not able to achieve good accuracy on very low-bit (2, 3, 4-bit) quantization.

Quantization-aware training generally outperforms these methods on low-bit tasks given enough time to optimize. Simulated quantization-aware training methods and improvements for these are discussed in Gupta2015,jacob2018cvpr,louizos2018relaxed. Essentially, operations are added to the neural network computational graph that simulate how quantization would be done on an actual device. Several recent papers improve over these methods by learning the quantization parameters, e.g. QIL [13], TQT [12] and LSQ [7]. This is the approach we build upon in our paper, but a similar asymmetric quantization scheme and initialization we suggest could be used for any other methods.

In a parallel line of research, some works [14, 16, 19] have tried to apply knowledge distillation to quantization resulting in improved performances. Also, some recent work [26] has been done on automatically learning the bit-width alongside of the ranges. Note that our proposed method is orthogonal to these works, and thus it can be jointly used with them. Lastly, several papers have introduced different quantization grids than uniform one we use. In [17] and [27], a logarithmic space or fully free-format quantization space are used to quantize the network. In this paper, we do not consider this, as the hardware implementations for these are simply inefficient, requiring costly lookup table or approximation on runtime.

3. Method

In LSQ [7], a symmetric quantization scheme with a trainable scale parameter is proposed for both weights and activations. This scheme is defined as follows:

$$\begin{aligned} \bar{x} &= \left\lfloor clamp\left(\frac{x}{s}, n, p\right)\right\rceil \\ \hat{x} &= \bar{x} \times s \end{aligned} \tag{1}$$

where $\lfloor\cdot\rceil$ indicates the round function and the $clamp(\cdot)$ function clamps all values between n and p. $\bar{x}$ and $\hat{x}$ denote the coded bits and quantized values, respectively. LSQ can make use of a signed or an unsigned quantization range. However, both are suboptimal for activation functions like Swish or Leaky-ReLU which have skewed negative and positive ranges[2]. Using an unsigned quantization range, i.e. $n = 0, p = 2^b - 1$, clamps all negative activations to zero leading to a significant loss of information. On the contrary, using a signed quantization range, i.e. $n = -2^{b-1}, p = 2^{b-1} - 1$, will quantize all negative activations to integers in the range $[-2^{b-1}, 0]$ and all positive activations to $[0, 2^{b-1} - 1]$, hence giving equal importance to the negative and positive portions of the activation function. However, this loses valuable precision for skewed distributions where the positive dynamic range is significantly larger than the negative one. In Sec. 4.1, we will show that both quantization schemes lead to a significant loss in accuracy when quantizing architectures with Swish activations.

The proposed method LSQ+ solves the above mentioned problem with a more general learnable asymmetric quantization scheme for the activations, described in Sec. 3.1. Sec. 3.2 describes the initialization scheme used in LSQ+.

3.1. Learnable asymmetric quantization

As a solution to the above mentioned problem, we propose a general asymmetric activation quantization scheme

[2] For example, the negative portion of Swish activation lies only between -0.278 and 0 whereas the positive portion is unbounded.

where not only the scale parameters but also the offset parameters are learned during training to handle skewed activation distributions:

$$\bar{x} = \left\lfloor clamp\left(\frac{x-\beta}{s}, n, p\right)\right\rceil \tag{2}$$
$$\hat{x} = \bar{x} \times s + \beta$$

Here, the offset parameter β and the scale s are both learnable. The gradient update of the parameter s is calculated using:

$$\frac{\partial \hat{x}}{\partial s} = \frac{\partial \bar{x}}{\partial s} s + \bar{x}$$
$$\simeq \begin{cases} -\frac{x-\beta}{s} + \left\lfloor \frac{x-\beta}{s} \right\rceil & \text{if } n < \frac{x-\beta}{s} < p \\ n \text{ or } p & \text{otherwise.} \end{cases} \tag{3}$$

And the gradient update of β is calculated using:

$$\frac{\partial \hat{x}}{\partial \beta} = \frac{\partial \bar{x}}{\partial \beta} s + 1 \simeq \begin{cases} 0 & \text{if } n < (x-\beta)/s < p \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

In both (3) and (4), straight-through-estimator (STE) [3] is used in approximating $\partial \bar{x}/\partial s$ and $\partial \bar{x}/\partial \beta$.

For weight quantization, we use symmetric signed quantization (1) since the layer weights can be empirically observed to be distributed symmetrically around zero. Because of this, asymmetric quantization of activations has no additional cost during inference as compared to symmetric quantization since the additional offset term can be precomputed and incorporated into the bias at compilation time:

$$\hat{w}\hat{x} = (\bar{w} \times s_w)(\bar{x} \times s_x + \beta) = \bar{w}\bar{x}s_w s_x + \underbrace{\beta s_w \bar{w}}_{bias}. \tag{5}$$

Table 1 shows four possible parametrizations for the proposed quantization scheme in (2). Configurations 1 and 2 do not use an offset parameter, hence following the learnable symmetric quantization scheme proposed in LSQ [7]. Since Configuration 1 uses an unsigned range with this symmetric quantization scheme, it corresponds exactly to the parametrization proposed in LSQ for activation quantization. Configurations 3 and 4 learn both the scale and offset parameter for activation quantization, the only difference being signed and unsigned quantization ranges. We will analyze these different parametrizations in the experiments section.

3.2. Initialization of quantization parameters

As we enter the *extremely low bit-width* regime with gradient-based learnable quantization methods, the final performance after training becomes highly sensitive to the initialization of the quantization hyperparameters. This sensitivity problem is amplified in the presence of depthwise separable convolutions which are known to be challenging to quantize [28]. In this work, we propose an initialization scheme for the scale and offset parameters that achieves significantly more stable and sometimes better performance than other initializations (while keeping the quantization configuration unchanged) proposed in the literature [11, 7].

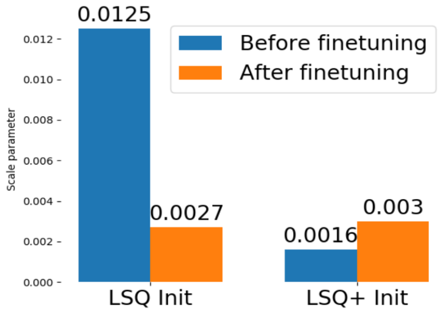

Figure 1. Figure shows the scale parameter of weight quantizer in **blocks.1.conv.0** layer of EfficientNet-B0 before and after finetuning with LSQ and LSQ+ initializations. For both experiments, we used configuration 4 for activation quantization. As shown, LSQ init of the scale is further from the converged value as compared to LSQ+. More on effects of initialization in Sec 4.3

3.2.1 Scale initialization for weight quantization

As mentioned before, we use signed symmetric quantization for the weights (similar to Configuration 2) in our method. Hence, no offset is used for weight quantization. LSQ [7] proposes using the square-root normalized average absolute value of layer weights, i.e. $2\langle|w|\rangle/\sqrt{p}$, to intialize the scale parameter. This leads to a very large initialization for 2-, 3- or 4-bit quantizaiton, e.g. $s_{init} = \langle|w|\rangle/\sqrt{2}$ for 4-bit case. From our experiments, this initialization was observed to be far from the converged values of the scale parameters. One of the instances of this phenomenon is shown in Figure 1.

We fix this problem by using the statistics of the weight distribution rather than the actual weight values for the initialization. Similar to [18], we use a Gaussian approximation for the weight distribution in each layer. Following this, we initialize the scale parameter for each layer by:

$$s_{init} = max(|\mu - 3 * \sigma|, |\mu + 3 * \sigma|)/2^{b-1}$$

where μ and σ are the mean (same as $\langle|w|\rangle$) and standard deviation of the weights in that layer.

3.2.2 Scale/offset initialization for activation quantization

Let x_{min} and x_{max} denote the min and the max value of the activation function. For example, $x_{min} = 0$ for ReLU and

Table 1. Different possible parametrizations for LSQ+'s learnable asymmetric quantization scheme

Configuration	s	β	n	p
Config 1 : Unsigned + Symmetric (LSQ)	trainable	N/A	0	$2^b - 1$
Config 2 : Signed + Symmetric	trainable	N/A	-2^{b-1}	$2^{b-1} - 1$
Config 3 : Signed + Asymmetric	trainable	trainable	-2^{b-1}	$2^{b-1} - 1$
Config 4 : Unsigned + Asymmetric	trainable	trainable	0	$2^b - 1$

$x_{min} = -0.278$ in case of Swish activations[3]. Intuitively, a full utilization of the quantization range can be obtained when x_{min} is quantized to the lower bound of the quantization range and x_{max} to the upper bound. Following this intuition, an initialization for s and β would satisfy:

$$\frac{x_{min} - \beta_{init}}{s_{init}} \to n \quad , \quad \frac{x_{max} - \beta_{init}}{s_{init}} \to p. \tag{6}$$

Solving these constraints yields:

$$s_{init} = \frac{x_{max} - x_{min}}{p - n} \quad , \beta_{init} = x_{min} - n * s_{init}. \tag{7}$$

But the above initialization is highly prone to outliers in the activation distribution, especially since the activation ranges are dynamic. To overcome this, we propose initializing the scale and offset parameters per layer by optimizing the MSE minimization problem, similar to [22, 23]:

$$s_{init}, \beta_{init} = \arg\min_{s,\beta} ||\hat{x} - x||_F^2 \tag{8}$$

where $\hat{x}$ is given by (2). There is no closed-form solution to (8). Hence, we embed equations (3) and (4) into PyTorch's autograd functionality to optimize for $\{s_{init}, \beta_{init}\}$ over a few batches of data.

4. Experiments

We evaluate the effectiveness of our method by quantizing architectures with Swish activations to W2A2, W3A3 and W4A4. To the best of our knowledge, ours is the first work to quantize such architectures to extremely low bitwidths. As a sanity check, we show that LSQ+ also maintains the performance of LSQ [7] on traditional architectures with ReLU activation function. Finally, we show the effect of using different initializations on the performance of the proposed quantization method. All experiments are performed on the ImageNet [21] dataset.

In all configurations and all experiments, the weight parameters are initialized with the pretrained floating point weights of the deep network. Although we will compare the effectiveness of different initializations for the scale/offset parameters in Sec 4.3, we use our proposed initialization from Sec 3.2 for experiments in sections 4.1 and 4.2.

4.1. Results on Swish activation

Tables 2 and 3 show the performance impact of quantization with all the configurations of the proposed method on EfficientNet-B0 [24] and MixNet-S [25], respectively. MixNet-S uses ReLU activation in the initial 3 layers and Swish activation in rest of the layers. By using the learnable offset parameter, we observe a 1.6-1.8% and 1.2-1.3% performance improvement for W4A4 quantization on EfficientNet-B0 and MixNet-S respectively (see Configurations 3 and 4 compared to Configuration 1 (LSQ)). This performance improvement using our proposed learnable asymmetric quantization scheme is most prominent in the case of W2A2 quantization.

The performance of Configuration 3 (signed range + learnable offset) and 4 (unsigned range + learnable offset) is almost similar for all the bit-widths. This is because, since we learn the offset parameter, the activation range is appropriately mapped to the quantization range irrespective of it being signed or unsigned.

Another interesting observation is that Configuration 2 performs consistently worse than all other configurations. This is because, due to the lack of an offset parameter, only 2^{b-1} quantization levels are utilized by the positive part of the activation range while the positive portion of the Swish activation is much larger than the negative portion, as mentioned in Section 3. Hence, compared to Configurations 3 and 4 which allocate 2^b quantization levels for the entire activation range, Configuration 2 has a poor utilization of its quantization range, leading to a worse performance.

4.2. Results on ReLU activation

The results on ResNet shown in the LSQ paper [7] use the pre-activation version of ResNet architecture [9] which has about 0.4-0.6% higher top-1 ImageNet accuracy than the standard ResNet(s). Hence, for a fair comparison with other state-of-the-art methods, we run our own implementation of LSQ (Configuration 1) and all other configurations on the standard ResNets. Tables 4 shows the quantization performance of all the configurations of the proposed method on ResNet18. Our implementation of LSQ (Configuration 1) can achieve a 70.7% accuracy with W4A4 quantization which is more than full-precision accuracy of 70.1%. This is sanity check that proves that our LSQ results are *at par* with the original LSQ paper [7]. Also, Con-

[3] For unbounded activation functions (e.g. positive portion of Swish), x_{min} or x_{max} can be estimated from a few forward passes.

Table 2. Comparison of all configurations of quantization with EfficientNet-B0 (FP accuracy: 76.1%)

Method	W2A2	W3A3	W4A4
Config 1 : LSQ (Unsigned + Symmetric)	43.5%	67.5%	71.9%
Config 2 : Signed + Symmetric	23.7%	54.8%	68.8%
Config 3 : Signed + Asymmetric	**49.1%**	**69.9%**	73.5%
Config 4 : Unsigned + Asymmetric	48.7%	69.3%	**73.8%**

Table 3. Comparison of all configurations of quantization with MixNet-S (FP accuracy: 75.9%)

Method	W2A2	W3A3	W4A4
Config 1 : LSQ (Unsigned + Symmetric)	39.9%	64.3%	70.4%
Config 2 : Signed + Symmetric	23.4%	62.1%	67.2%
Config 3 : Signed + Asymmetric	42.5%	**66.7%**	71.6%
Config 4 : Unsigned + Asymmetric	**42.8%**	66.1%	**71.7%**

figurations 1, 3 and 4 outperform existing state-of-the-art methods, namely PACT [5], DSQ [8] and QIL [13]. It is worth noting that, unlike EfficientNet and MixNet, there is almost no performance gap between Configurations 1, 3 and 4 when quantizing ResNet18. We attribute this to the fact that ReLU activation function has no negative component.

4.3. Effect of quantization parameter initialization

In this section, we compare three schemes for initializing the quantization scale and offset parameters. Since we use symmetric quantization for weights, no offset is used for weight quantization. Also, configurations 1 and 2 for activation quantization don't use an offset. The three compared initialization methods are as follows:

1. **Min-max initialization.** We use the minimum and maximum values of each layer's weights and activations (obtained over first batch of input images) to initialize the quantization scale and offset parameters. This initialization scheme is formalized in (7).
2. **LSQ initialization.** The scale for both weight quantization and activation quantization is initialized as $2 * mean(|v|)/\sqrt{p}$, where v indicate layer weights or activations and p is the upper bound of the quantization range.
3. **LSQ+ initialization.** We intialize the weight quantization and activation quantization parameters as proposed in Sec. 3.2

For the experiments, we quantize EfficientNet-B0 using Configuration 4 and perform multiple training runs with each of these initialization methods. Table 5 shows the variation (Δ_{acc}) in the final performance across 5 training runs with each of these initializations. We can observe a high instability in the final performance with W2A2 quantization, especially with min-max quantization. This is because the tail of the weight or activation distribution can easily influence the scale parameter intialization with 2-bit quantization. The LSQ initialization method, which initializes the weight quantization scale parameter with the square-root normalized mean absolute value, also has a higher variation in training performance. This is because LSQ initialization leads to a large value for the s_{init} which is far from the converged value as was shown in Figure 1.

5. Discussion

5.1. Learned offset values

It is interesting to observe the layer-wise offset values learned by the network. Figure 2 shows one such example with Configuration 4 for W4A4 quantization of EfficientNet-B0. Note that an offset is not used for quantizing the squeeze-excite layers because *sigmoid* activation function has no negative component. Also, there is no activation applied at the end of a bottleneck block in EfficientNet, hence we use symmetric-signed-quantization for those activation layers. These layers are not shown in the plot.

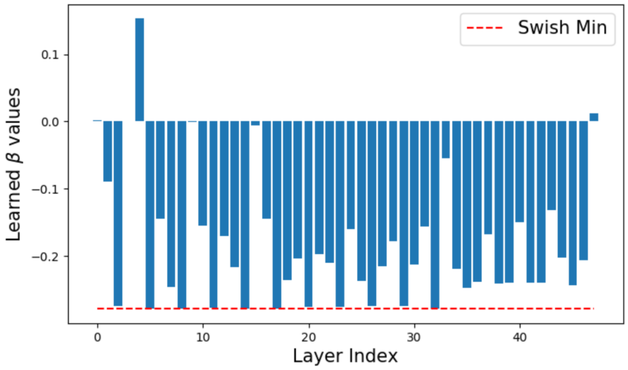

Figure 2. Layerwise β values after covergence for EfficientNet-B0

We can observe that most of the β values are negative, meaning that the activations are shifted "up" before being

Table 4. Comparison of all configurations of quantization with ResNet18 (FP accuracy: 70.1%)

Method	W2A2	W3A3	W4A4
PACT [5]	64.4%	68.1%	69.2%
DSQ [8]	65.2%	68.7%	69.6%
QIL [13]	65.7%	69.2%	70.1%
Config 1 : LSQ (Unsigned + Symmetric)	66.7%	**69.4%**	70.7%
Config 2 : Signed + Symmetric	64.7%	66.1%	69.2%
Config 3 : Signed + Asymmetric	66.7%	**69.4%**	70.7%
Config 4 : Unsigned + Asymmetric	**66.8%**	69.3%	**70.8%**

Table 5. Δ_{acc} around mean accuracy across 5 training runs for EfficientNet quantization using Config 4 with different initializations. Note: other tables show the *best* accuracy after grid search on hyperparameters, which is different from mean accuracy.

Quantization Parameter Initialization	Mean Acc $\pm\Delta_{acc}$	
	W4A4	W2A2
Mix-max	$71.3 \pm 2.2\%$	$43.8 \pm 4.7\%$
LSQ	$72.0 \pm 1.6\%$	$44.4 \pm 2.9\%$
LSQ+	$73.0 \pm \mathbf{0.9}\%$	$46.8 \pm \mathbf{1.9}\%$

scaled and clamped between the quantization range. This shows that the quantization layers learn to accommodate the negative activation values. None of the learned β values are lower than the min value of the Swish activation function (red dotted line). Because, from (4),

$$\beta < x_{min} \implies \frac{x-\beta}{s} > 0 \ \forall x > x_{min} \implies \frac{\partial \hat{x}}{\partial \beta} = 0$$

Hence, gradient for β becomes zero as soon as $\beta < x_{min}$.

5.2. Learned vs Fixed offset

On further observation of Figure 2, the learned offset for most layers is away from the Swish minimum value. This is because, if we try to represent the entire activation range using the quantization grid (refer (6)), it leads to coarser representation since the number of bits are fixed causing a higher quantization error. The purpose of learning the s and β values is to learn this trade-off between resolution of the quantization grid and the proportion of activation range represented by the quantization grid. Hence, the learned β values are not exactly equal to the min value of the activation function. But one might wonder about the performance achieved when β for each layer is fixed to x_{min}. Table 6 shows the difference of performance between fixed and learned offset methods.

Table 6. Performance difference between fixed and learned offset for EfficientNet quantization at W4A4 using Config 4

Method	W4A4
Fixed $\beta = 0$ (LSQ)	71.9%
Fixed $\beta = x_{min}$	72.5%
Learned β	73.8%

6. Conclusion

In this work, targeting the low-bit quantization domain, we solve two problems: (1) quantization of deep neural networks with signed activation functions and (2) stability of training performance w.r.t. quantization. To do so, we propose a general asymmetric quantization scheme with trainable scale and offset parameters that can learn to accommodate the negative activations without using an extra sign bit. In (5), we show that using such asymmetric quantization for activations incurs zero runtime overhead. Our work is the first to quantize modern efficient architectures like EfficientNet and MixNet to extremely low bits. We show that LSQ+ significantly improves the performance of 2-, 3- and 4-bit quantization on these architectures. Our experiments with traditional ReLU-based ResNet18 architecture show that we can use LSQ+ instead of LSQ everywhere without hurting performance. Finally, we show that using MSE-minimization based initialization scheme for the activation quantization parameters leads to a more stable performance, which is of high importance for low-bit quantization-aware training.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 2

[2] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-

deployment. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 7948–7956, 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] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. *CoRR*, abs/2001.00281, 2020. 2

[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 preprint arxiv:805.06085*, 2018. 1, 5, 6

[6] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pages 3009–3018, 2019. 2

[7] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 1, 2, 3, 4

[8] 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. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4852–4861, 2019. 5, 6

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 1, 4

[10] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 1

[11] 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. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 3

[12] Sambhav R Jain, Albert Gural, Michael Wu, and Chris H Dick. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. *arXiv preprint arXiv:1903.08066*, 2(3):7, 2019. 2

[13] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4350–4359, 2019. 2, 5, 6

[14] Jangho Kim, Yash Bhalgat, Jinwon Lee, Chirag Patel, and Nojun Kwak. Qkd: Quantization-aware knowledge distillation. *arXiv preprint arXiv:1911.12491*, 2019. 2

[15] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, Jun 2018. 2

[16] Asit Mishra and Debbie Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *arXiv preprint arXiv:1711.05852*, 2017. 2

[17] Daisuke Miyashita, Edward H. Lee, and Boris Murmann. Convolutional neural networks using logarithmic data representation. *arXiv preprint arxiv:1603.01025*, 2016. 2

[18] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1325–1334, 2019. 2, 3

[19] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018. 2

[20] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 1

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 4

[22] Sungho Shin, Yoonho Boo, and Wonyong Sung. Fixed-point optimization of deep neural networks with adaptive step size retraining. In *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pages 1203–1207. IEEE, 2017. 2, 4

[23] Wonyong Sung, Sungho Shin, and Kyuyeon Hwang. Resiliency of deep neural networks under quantization. *arXiv preprint arXiv:1511.06488*, 2015. 2, 4

[24] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 1, 4

[25] Mingxing Tan and Quoc V Le. Mixconv: Mixed depthwise convolutional kernels. *CoRR, abs/1907.09595*, 2019. 1, 4

[26] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso García, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed precision dnns: All you need is a good parametrization. 2

[27] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. In *International Conference on Learning Representations (ICLR)*, 2017. 2

[28] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8612–8620, 2019. 3

[29] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Christopher De Sa, and Zhiru Zhang. Improving neural network quantization

without retraining using outlier channel splitting. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 7543–7552, 2019. 2