Learning Non-Uniform Step Sizes for Neural Network Quantization (Supplemental materials)

A Reformulation of LSQ

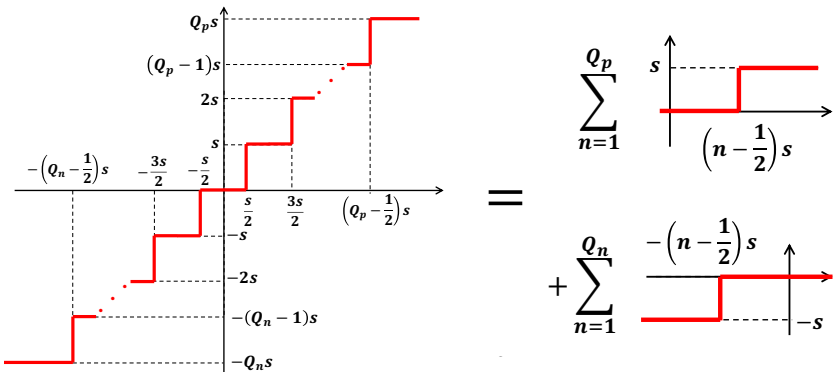

Fig. A: Decomposition of the uniform quantizer into a sum of step functions

As shown in Fig. A, the uniform quantizer of LSQ, $Q_{\mathrm{LSQ}}(x,s)$, is rewritten as

$$\begin{aligned} Q_{\mathrm{LSQ}}(x,s) &= \sum_{n=1}^{Q_p} s\sigma\left(x-\left(n-\frac{1}{2}\right)s\right) + \sum_{n=1}^{Q_n}(-s)\sigma\left(-x-\left(n-\frac{1}{2}\right)s\right) \\ &= \begin{cases} -\sum_{n=1}^{Q_n} s\sigma\left(-x-ns+\frac{s}{2}\right), & x<0 \\ \sum_{n=1}^{Q_p} s\sigma\left(x-ns+\frac{s}{2}\right). & x\geq 0 \end{cases} \end{aligned} \tag{1}$$

We perform its derivative with respect to the step size s,

$$\begin{aligned} \frac{\partial Q_{\mathrm{LSQ}}(x,s)}{\partial s} &= \sum_{n=1}^{Q_p}\sigma\left(x-\left(n-\frac{1}{2}\right)s\right) - \sum_{n=1}^{Q_n}\sigma\left(-x-\left(n-\frac{1}{2}\right)s\right) \\ &\quad + \sum_{n=1}^{Q_p} s\frac{\partial\sigma\left(x-\left(n-\frac{1}{2}\right)s\right)}{\partial s} - \sum_{n=1}^{Q_n} s\frac{\partial\sigma\left(-x-\left(n-\frac{1}{2}\right)s\right)}{\partial s} \\ &= \sum_{n=1}^{Q_p}\sigma\left(x-\left(n-\frac{1}{2}\right)s\right) - \sum_{n=1}^{Q_n}\sigma\left(-x-\left(n-\frac{1}{2}\right)s\right) \\ &\quad + \sum_{n=1}^{Q_p} s\frac{\partial\sigma\left(x-\left(n-\frac{1}{2}\right)s\right)}{\partial s} + \sum_{n=1}^{Q_n} s\frac{\partial\sigma\left(x+\left(n-\frac{1}{2}\right)s\right)}{\partial s} \\ &= \sum_{n=1}^{Q_p}\sigma\left(x-\left(n-\frac{1}{2}\right)s\right) - \sum_{n=1}^{Q_n}\sigma\left(-x-\left(n-\frac{1}{2}\right)s\right) + \sum_{n=-Q_n+1}^{Q_p} s\frac{\partial\sigma\left(x-\left(n-\frac{1}{2}\right)s\right)}{\partial s} \end{aligned} \tag{2}$$

where we used $\sigma(x) = 1-\sigma(-x)$ and its derivative $\frac{\partial\sigma(x)}{\partial s} = -\frac{\partial\sigma(-x)}{\partial s}$. As shown in Fig. B, we apply the STE to the derivative of each step function,

$$\frac{\partial Q_{\mathrm{LSQ}}(x,s)}{\partial s} \simeq \sum_{n=1}^{Q_p}\sigma\left(x-\left(n-\frac{1}{2}\right)s\right) - \sum_{n=1}^{Q_n}\sigma\left(-x-\left(n-\frac{1}{2}\right)s\right) - \sum_{n=-Q_n+1}^{Q_p}\frac{x}{s}1_{(n-1)s<x\leq ns}, \tag{3}$$

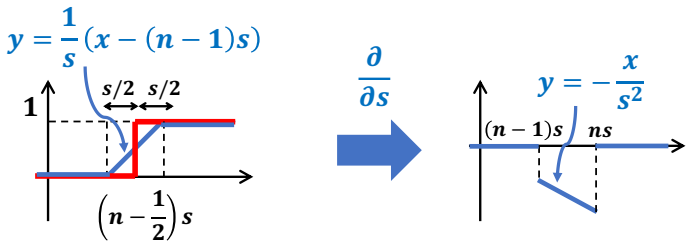

Fig. B: STE of each step function

where $1_{a<x\leq b}$ is defined as 1 if $a < x \leq b$, otherwise 0.
Using Eq.(1), this reduces to

$$\frac{\partial Q_{\mathrm{LSQ}}(x,s)}{\partial s} \simeq \frac{Q_{\mathrm{LSQ}}(x,s)}{s} - \frac{x}{s}1_{-Q_n s<x\leq Q_p s} = \begin{cases} -Q_n, & x/s \leq -Q_n \\ \lfloor x/s \rceil - x/s, & -Q_n < x/s < Q_p \\ Q_p, & x/s \geq Q_p. \end{cases} \tag{4}$$

This expression coincides with the form obtained by the original LSQ [5].

B Extension to nuLSQ

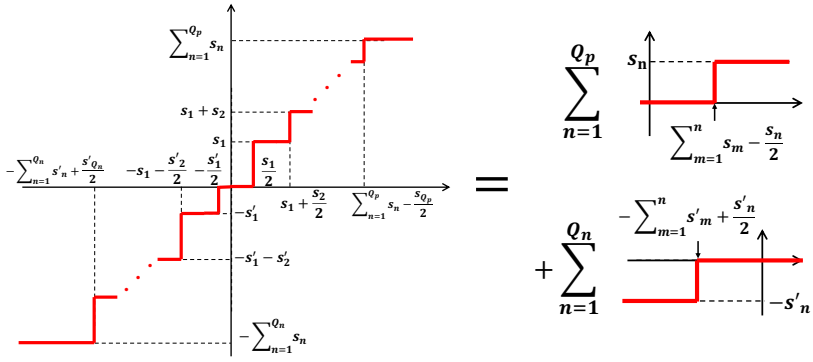

Fig. C: Decomposition of the nonuniform quantizer into a sum of step functions

Similar to Eq.(1), the nonuniform quantizer, $Q_{\mathrm{nuLSQ}}(x, \{s_i\}, \{s'_i\})$, is expressed by its decomposition into the step functions (see: Fig.C):

$$Q_{\mathrm{nuLSQ}}(x, \{s_i\}, \{s'_i\}) = \sum_{n=1}^{Q_p} s_n \sigma\left(x - \sum_{m=1}^{n} s_m + \frac{s_n}{2}\right) - \sum_{n=1}^{Q_n} s'_n \sigma\left(-x - \sum_{m=1}^{n} s'_m + \frac{s'_n}{2}\right) \tag{5}$$

The gradients of the nonuniform quantizer with respect to step sizes can be derived as was done in the reformulation of the LSQ gradient. Let us consider the derivatives with respect to s_k and s'_k:

$$\frac{\partial Q_{\mathrm{nuLSQ}}(x, \{s_i\}, \{s'_i\})}{\partial s_k} = \sigma\left(x - \sum_{m=1}^{k} s_m + \frac{s_k}{2}\right) + \sum_{n=1}^{Q_p} s_n \frac{\partial \sigma\left(x - \sum_{m=1}^{n} s_m + \frac{s_n}{2}\right)}{\partial s_k}, \tag{6}$$

$$\frac{\partial Q_{\mathrm{nuLSQ}}(x, \{s_i\}, \{s'_i\})}{\partial s'_k} = -\sigma\left(-x - \sum_{m=1}^{k} s'_m + \frac{s'_k}{2}\right) + \sum_{n=1}^{Q_n} s'_n \frac{\partial \sigma\left(x + \sum_{m=1}^{n} s'_m - \frac{s'_n}{2}\right)}{\partial s'_k}, \tag{7}$$

where we used $\sigma(x) = 1 - \sigma(-x)$ and its derivative $\frac{\partial \sigma(x)}{\partial s} = -\frac{\partial \sigma(-x)}{\partial s}$.

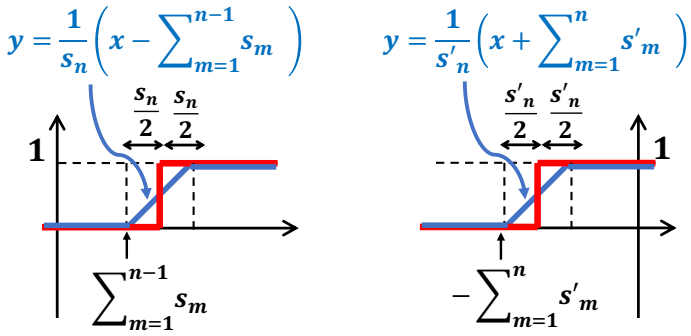

Fig. D: STE of each step function in the nonuniform quantizer

Let us first consider the derivative with respect to s_k. By applying the STE to the derivative of each step function as shown in Fig.D, we get

$$\begin{aligned}
\frac{\partial Q_{\text{nuLSQ}}(x,\{s_i\},\{s'_i\})}{\partial s_k} &\simeq \sigma\left(x - \sum_{m=1}^{k} s_m + \frac{s_k}{2}\right) + \sum_{n=1}^{Q_p} s_n \frac{\partial\left\{\frac{1}{s_n}\left(x - \sum_{m=1}^{n-1} s_m\right) 1_{\sum_{m=1}^{n-1} s_m < x \le \sum_{m=1}^{n} s_m}\right\}}{\partial s_k} \\
&= \sigma\left(x - \sum_{m=1}^{k} s_m + \frac{s_k}{2}\right) + \sum_{n=1}^{Q_p} s_n \left(-\frac{1}{s_n^2}\delta_{n,k}\left(x - \sum_{m=1}^{n-1} s_m\right) - \frac{1}{s_n}\sum_{m=1}^{n-1}\delta_{m,k}\right) 1_{\sum_{m=1}^{n-1} s_m < x \le \sum_{m=1}^{n} s_m} \\
&= \sigma\left(x - \sum_{m=1}^{k} s_m + \frac{s_k}{2}\right) - \frac{1}{s_k}\left(x - \sum_{m=1}^{k-1} s_m\right) 1_{\sum_{m=1}^{k-1} s_m < x \le \sum_{m=1}^{k} s_m} \\
&\quad - \sum_{n=1}^{Q_p}\left(\sum_{m=1}^{n-1}\delta_{m,k}\right) 1_{\sum_{m=1}^{n-1} s_m < x \le \sum_{m=1}^{n} s_m},
\end{aligned} \tag{8}$$

where we have defined $s_0 = 0$. Because the last term is rewritten as

$$\sum_{n=1}^{Q_p}\left(\sum_{m=1}^{n-1}\delta_{m,k}\right) 1_{\sum_{m=1}^{n-1} s_m < x \le \sum_{m=1}^{n} s_m} = \sum_{n=k+1}^{Q_p} 1_{\sum_{m=1}^{n-1} s_m < x \le \sum_{m=1}^{n} s_m} = 1_{\sum_{m=1}^{k} s_m < x \le \sum_{m=1}^{Q_p} s_m}, \tag{9}$$

Eq.(8) is reduced to

$$\begin{aligned}
\frac{\partial Q_{\text{nuLSQ}}(x,\{s_i\},\{s'_i\})}{\partial s_k} &\simeq \sigma\left(x - \sum_{m=1}^{k} s_m + \frac{s_k}{2}\right) - \frac{1}{s_k}\left(x - \sum_{m=1}^{k-1} s_m\right) 1_{\sum_{m=1}^{k-1} s_m < x \le \sum_{m=1}^{k} s_m} - 1_{\sum_{m=1}^{k} s_m < x \le \sum_{m=1}^{Q_p} s_m} \\
&= \left\{\sigma\left(x - \sum_{m=1}^{k} s_m + \frac{s_k}{2}\right) - \frac{1}{s_k}\left(x - \sum_{m=1}^{k-1} s_m\right)\right\} 1_{\sum_{m=1}^{k-1} s_m < x \le \sum_{m=1}^{k} s_m} + 1_{\sum_{m=1}^{Q_p} s_m < x} \\
&= \begin{cases} 0, & x < \sum_{m=1}^{k-1} s_m \\ D_k(x,\{s_i\}), & \sum_{m=1}^{k-1} s_m \le x < \sum_{m=1}^{k} s_m \\ 0, & \sum_{m=1}^{k} s_m \le x < \sum_{m=1}^{Q_p} s_m \\ 1, & \sum_{m=1}^{Q_p} s_m \le x \end{cases}
\end{aligned} \tag{10}$$

with

$$D_k(x,\{s_i\}) = \sigma\left(x - \sum_{m=1}^{k} s_m + \frac{s_k}{2}\right) - \frac{x - \sum_{m=1}^{k-1} s_m}{s_k}. \tag{11}$$

Next, let us consider the derivative with respect to s'_k in Eq.(7). The approximated gradient using STE can be readily obtained in the same way (see Fig.D),

$$\frac{\partial Q_{\text{nuLSQ}}(x,\{s_i\},\{s'_i\})}{\partial s'_k} \simeq -\sigma\left(-x - \sum_{m=1}^{k} s'_m + \frac{s'_k}{2}\right) + \sum_{n=1}^{Q_n} s'_n \frac{\partial\left\{\frac{1}{s'_n}\left(x + \sum_{m=1}^{n} s'_m\right) 1_{-\sum_{m=1}^{n} s'_m < x \le -\sum_{m=1}^{n-1} s'_m}\right\}}{\partial s'_k}$$

$$
\begin{aligned}
&= -\sigma\left(-x - \sum_{m=1}^{k} s'_m + \frac{s'_k}{2}\right) - \frac{1}{s'_k}\left(x + \sum_{m=1}^{k} s'_m\right) 1_{-\sum_{m=1}^{k} s'_m < x \leq -\sum_{m=1}^{k-1} s'_m} + 1_{-\sum_{m=1}^{Q_n} s'_m < x \leq -\sum_{m=1}^{k-1} s'_m} \\
&= \left\{-\sigma\left(-x - \sum_{m=1}^{k} s'_m + \frac{s'_k}{2}\right) - \frac{1}{s'_k}\left(x + \sum_{m=1}^{k-1} s'_m\right)\right\} 1_{-\sum_{m=1}^{k} s'_m < x \leq -\sum_{m=1}^{k-1} s'_m} - 1_{x < -\sum_{m=1}^{Q_n} s'_m} \\
&= \begin{cases} -1, & x < -\sum_{m=1}^{Q_n} s'_m \\ 0, & -\sum_{m=1}^{Q_n} s'_m \leq x < -\sum_{m=1}^{k} s'_m \\ B_k(x, \{s'_i\}), & -\sum_{m=1}^{k} s'_m \leq x < -\sum_{m=1}^{k-1} s'_m \\ 0, & -\sum_{m=1}^{k-1} s'_m \leq x \end{cases}
\end{aligned} \tag{12}
$$

with

$$
B_k(x, \{s'_i\}) = -\sigma\left(-x - \sum_{m=1}^{k-1} s'_m + \frac{s'_k}{2}\right) - \frac{x + \sum_{m=1}^{k-1} s'_m}{s'_k}. \tag{13}
$$

C Hyperparameters and other settings

We utilized the pre-trained models from the timm library for MobileNetV2, original version of ResNet-18, Swin-T and ConvNeXt [2]. Additionally, the pre-trained models for the preactivation versions of ResNet-18, and -34 on Imagenet as well as ResNet-20 and -56 on CIFAR-10 and -100 were taken from PytorchCV [1].

C-I MobileNetV2, original and preactivation versions of ResNet-18, Swin-T, and ConvNeXt on Imagenet

Settings for nuLSQ on MobileNetV2 at Tables 3 and 4 of our main paper We used the SGD optimizer with a Nesterov momentum of 0.9 for weights and the AdamW optimizer with MSE-base initialization for step sizes with a batch size of 512. The cosine learning rate decay with the linear warm-up method was applied. The initial learning rate for weights were set at 0.4 in 2-bit and 0.04 in 3- and 4-bits. We set the initial learning rate for step sizes at 1/100 of the learning rate of the weights. The weight decays for weights, w_d, were set to be 6.25e-6 in 2- and 3-bits, and 1.25e-5 in 4-bit. The optimal weight decays for step-sizes were selected in the range of $\{0, 0.5, 1, 2\} \times w_d$ from the performance of the validation data comprising 10% of the original training dataset on ImageNet after training with 90% of the original training dataset on ImageNet for 15 epochs. Following the hyperparamter optimization, we conducted training for 90-epochs with the selected hyperparamters on the full training dataset, and evaluated performance on the original validation dataset.
Settings for existing methods and nuLSQ on original version of ResNet-18 at Table 5 of our main paper We used the SGD optimizer with a Nesterov momentum of 0.9 for weights and the AdamW optimizer with the MSE-base initialization for quantizer parameters. The cosine learning rate decay with the linear warm-up method was applied. The quantizer parameters are given by clipping threshold in PACT and APoT, step size in LSQ and nuLSQ-A, and companding paramters and clipping threshold in LCQ. DoReFa does not have the parameters, while LQ-Nets has the parameters, quantizer basis, determined via MSE loss minimization in forward pass, not updated by the optimizer in backward pass. For all quantizers, the initial learning rate for weights, lr, was set to be 0.004 and the weight decays for weights, w_d to be 5e-5. The optimal initial learning rates and weight decays for quantizer parameters were selected in the range of $\{1, 0.1, 0.01\} \times lr$ and $\{0, 0.5, 1, 2\} \times w_d$, respectively, via validation data comprising 10% of the original training dataset on ImageNet after 15-epoch training. With the selected hyperparamters, we evaluated the performance using the original validation dataset.
Settings for nuLSQ on preactivation version of ResNet-18 at Table 1 of our main paper We used the SGD optimizer with a Nesterov momentum of 0.9 for weights and the AdamW optimizer with MSE-base initialization for step sizes with a batch size of 512. The cosine learning rate decay with the linear warm-up method was applied. We set the initial learning rate for weights to be 0.2, and the initial learning rate for step sizes to be $2e-5$, the weight decays for weights to be 2.5e-5, and the weight decays for step-sizes to be 0. We trained the network for 90 epochs and evaluated the test accuracy. As an ablation study, we conducted the experiments using SGD optimizer with the LSQ initialization, the AdamW optimizer with the LSQ initialization in addition to the AdamW optimizer with the MSE initialization, using the same hyperparameters mentioned above.
Settings for nuLSQ and LSQ on Swin-T and ConvNeXt at Table 7 of our main paper We used the AdamW optimizer for both weights and step sizes with a batch size of 1024. The cosine learning rate decay with the linear warm-up method was applied. We initalized step sizes using the MSE-base initialization. For Swin-T, we set the initial learning rate for weights, lr, to be 0.0004 in 2-, 3-, and 4-bits, and the weight decays for weights, w_d, to be 1.25e-5 in 2-bit, and 3.125e-6 in 3- and 4-bits. The optimal initial learning rate and weight decay for step sizes were selected in the range of

$\{1, 0.1\} \times lr$ and $\{0, 0.5, 1, 2\} \times w_d$, respectively, from the performance of validation data comprising 10% of the original training dataset on ImageNet after we trained in 15 epochs. Using the selected hyperparamters, we evaluated the test accuracy using the original validation dataset. For ConvNeXt, we set the initial learning rate for both weights and step sizes to be 0.0004, the weight decay for weights to be 6.25e-6, and the weight decay for step-sizes to be 0. We trained the network for 15 epochs and evaluated the performance using the original validation dataset.

C-II Resnet-20 and -56 on CIFAR-100

Settings for LSQ and nuLSQ We adopted the same settings for LSQ and nuLSQ for a fair comparison. We trained 2-, 3-, and 4-bits from the pre-trained floating-point model for 300 epochs by using SGD optimizer with a momentum of 0.9 for weights and AdamW optimizer for step sizes with a batch size of 128 and cosine learning rate decay with the linear warm-up method. The learning rate was linearly increased from $1e-5$ to the initial learning rate for 10 epochs, and then the cosine learning rate decay without restart was applied. We used the initial learning rate of $4e-2$ for weights. We used MSE-base uniform initialization as step-size initialization from the initial 100 training batches proposed in [11]. The other hyperparameters for both LSQ and nuLSQ were explored in the following same ranges, and the optimal one was selected from the mean value in five-times experiments: For the initial learning rate for step sizes, a rough grid search was performed in the range of $\{4e-2, 4e-3, 4e-4, 4e-5\}$. The weight decay for weights was explored in $\{1e-4, 2.5e-4, 5e-4, 7.5e-4, 1e-3\}$, and the weight decay for step sizes was set to the same value or 0. Both SGD with conventional and Nesterov momentum were explored with the above parameters.

C-III Resnet-20 and -56 on CIFAR-10

Settings for nuLSQ We replicated nearly identical settings to those employed in the CIFAR-100 experiment. The exploration range for initial learning rates for step sizes and weight decays was more restricted. We explored the initial learning rate in the range of $\{4e-2, 4e-3, 4e-4\}$ and the weight decay for weights in the range of $\{1e-4, 2.5e-4, 5e-4\}$. The weight decay for step sizes was either set to the same value or 0.

Settings for LCQ We followed the settings used in the original paper [13]. The number of intervals in the piecewise linear function as a companding function, K, was set to 16. We used SGD with a Nesterov momentum of 0.9, a batch size of 128, and cosine learning rate decay with initial values of 0.04 for weights and 0.02 for the clipping and companding parameters. The weight decay was set to $1e-4$. The clipping parameters for weights and activations were initialized to be 3.0 and 8.0, respectively, while the companding parameters were initialized as 0 to obtain uniform initialization. We trained 2-, 3-, and 4-bits for 300 epochs from the same pre-trained floating-point model used in nuLSQ. We tested the following two types of uniform quantizers for 2-bit weights: (i)LCQ type: Perfect symmetric quantizer which has three quantization levels, $\pm s, 0$, with latent weights updated with clipped STE. We expected that this was used for 2-bit weights in LCQ's experiment. (ii) APoT type: The same perfect symmetric quantizer with latent weights updated with unclipped STE. This seems to be used for the 2-bit weights in the original implementation of APoT [1]. We performed five times experiments and calculated the mean accuracy and standard deviation.

D Additional results

D-I Comparison between our results and original results of LCQ on CIFAR-10

We carefully compared our method with LCQ, the state-of-the-art nonuniform quantization method. Because the source code of LCQ is not officially released, we implemented LCQ by ourselves and conducted experiments with the settings described in the original paper. Here, we compared the performance of LCQ with our implementation and ones reported in the original paper to check the consistency. Our results of the mean accuracy and the best accuracy in 5 trials are listed in Table A, together with the original results. Although direct comparison is not inherently possible due to the use of the different pre-trained models, the best accuracy shows similar values to the original one in all cases except for 2bit, where Resnet-56 leads to a clearer difference. We found visible difference at 2-bit Resnet on Imagenet as well. Besides the use of different pre-trained models, one possible reason for the difference at only 2-bit could be the differences in the implementation of 2-bit weight uniform quantization. Regarding this, we examined several uniform quantizers of the weight at 2-bit, and observed a very small improvement over the naive one (LCQ type), but even the improved score did not reach the reported score. For example, as shown in Table A, APoT-type in ResNet-56 shows 0.4% improvement over the LCQ-type and achieves 92.3 % in best accuracy, which is still 1.2% lower than the reported score. Again we think that these differences most likely come from the differences in the implementation of the uniform quantizer, not from the expressivity of the nonuniform quantizer.

[1] https://github.com/yhhhli/APoT_Quantization

Table A: Comparing our implementation to original LCQ results on CIFAR-10

Network	FP	Methods	Bit-width(W/A)		
			2/2	3/3	4/4
ResNet-20	93.49	our results (mean)	90.94 ± 0.38 (APoT type) 90.66 ± 0.08 (LCQ type)	92.44 ± 0.15	93.16 ± 0.17
		our results (best)	91.4 (APoT type)	92.8	93.3
	93.4	original results [13]	91.8	92.8	93.2
ResNet-56	95.51	our results (mean)	91.82 ± 0.24 (APoT type) 91.40 ± 0.20 (LCQ type)	94.54 ± 0.18	94.67 ± 0.14
		our results (best)	92.3 (APoT type)	94.9	95.0
	94.5	original results [13]	93.5	94.6	94.7

D-II Full comparison between nuLSQ-A, nuLSQ-W, nuLSQ-WA and LCQ on CIFAR-100

Table.B summarizes the results of three configurations of nuLSQ and LCQ on CIFAR-100. We used the same pre-trained model for both nuLSQs and LCQ. The LCQ was trained using the hyperparameters followed by the settings on CIFAR-10. We can see that three types of nuLSQ consistently outperform LCQ in all cases.

Table B: Full comparison between three configurations of nuLSQ and LCQ on CIFAR-100. Results marked with * are from our implementation.

Network	Methods	Bit-width(W/A)		
		2/2	3/3	4/4
ResNet-20 (FP: 69.8)	**nuLSQ-A**	$\mathbf{66.02 \pm 0.29}$	68.58 ± 0.21	$\mathbf{69.42 \pm 0.43}$
	nuLSQ-W	66.00 ± 0.39	68.70 ± 0.11	69.40 ± 0.14
	nuLSQ-WA	66.00 ± 0.33	$\mathbf{68.76 \pm 0.14}$	69.40 ± 0.21
	*LCQ [13]	65.60 ± 0.23 (APoT type) 65.46 ± 0.28 (LCQ type)	67.36 ± 0.22	67.64 ± 0.15
ResNet-56 (FP: 74.9)	**nuLSQ-A**	70.66 ± 0.29	$\mathbf{72.98 \pm 0.10}$	$\mathbf{73.48 \pm 0.25}$
	nuLSQ-W	$\mathbf{70.82 \pm 0.22}$	72.80 ± 0.18	73.42 ± 0.25
	nuLSQ-WA	$\mathbf{70.86 \pm 0.26}$	72.80 ± 0.17	73.42 ± 0.26
	*LCQ [13]	69.60 ± 0.26 (LCQ type)	72.38 ± 0.21	73.24 ± 0.28

D-III Progressive fine tuning and comparison of existing methods in ResNet-18 and -34

Progressive fine-tuning. Apart from MSE init. + AdamW method, a way to mitigate the emergence of negative step sizes is to use progressive fune-tuning proposed by [15]. This approach involves gradually reducing the precision of quantization by training in multiple stages: the trained model with higher bit precision provides initial values for subsequent lower-bit models. However, in the case of non-uniform quantization, extra care has to be given since the number of step sizes differs in higher-bit and lower-bit networks. We found that merging two adjacent step sizes at (b + 1)-bit yields effective initialization at b-bit.

Effect of Progressive training. As an ablation study, we showed the results of 2-bit ReNet-18 about progressive fine-tuning training and AdamW optimizer with MSE initialization in Table C. We found that progressive fine-tuning training shows slightly better accuracy than the AdamW optimizer with MSE initialization.

Table C: NuLSQ-A results at 2-bit ResNet-18 with progressive fine-tuning training (PG) and AdamW optimizer with MSE initialization for step-sizes (AWamW + MSE)

PG	AdamW + MSE	Acc(%)
✓		67.9
	✓	67.8

Evaluation on Resnet architecture. We evaluated nuLSQ-A and nuLSQ-W in comparison with existing methods for the pre-activation version of ResNet-18 and -34 in 2-, 3-, and 4-bits, using a batch size of 256. We employed the progressive fine-tuning method in 2-bit: The 2-bit quantized models were trained from the pre-trained 3-bit models, while 3- and 4-bit quantized models were trained from the pre-trained floating-point model. We utilized SGD with a momentum of 0.9, and cosine learning rate decay without restart. Instead of the AdamW optimizer for each step-size update, the gradient scaling method was adopted [3, 5]. We trained 3- and 4-bits with an initial learning rate of 0.01 for 90 epochs, while we trained 2-bit from the trained 3-bit model with an initial learning rate of 0.001 for 60 epochs. The weight decay was set to $1e-4$ for 4-bit, $0.5e-4$ for 3-bit, and $0.25e-4$ for 2-bit. For 3- and 4-bits, we employed the improved LSQ initialization [5]: Each step size is uniformly initialized as $2\langle|x|\rangle/\sqrt{Q_p}$ where x is the initial weights values or the mean of the initial $1,000$ training batches of activations. For 2-bit, we initialized the step sizes by merging two adjacent step sizes from the trained 3-bit network.

From Table D, we can see that nuLSQ-A outperforms or is on par with all of the existing methods. In particular, the accuracy for ResNet-34 at 2-bit is 0.5% higher than the existing methods. When compared to the most related method, LSQ, we observe 0.3-0.5% performance improvements in ResNet-18, and more significantly, 0.9-1.2% improvement in ResNet-34. We also observe that nuLSQ-A is more effective than nuLSQ-W in Resnet architecture on ImageNet.

Table D: Comparison with state-of-the-art quantization methods on Resnet architecture. Results marked with * were obtained in our implementation from the same pre-trained model, using the original setup given in their corresponding papers. The other results of existing methods are cited from their corresponding papers. Results marked with † were obtained from the pre-activation version of ResNet.

Methods	Bit-width (W/A)		
	2/2	3/3	4/4
ResNet-18 (FP: 71.6)			
†PACT [4]	64.4	68.1	69.2
LQ-Nets [14]	64.9	68.2	69.3
DSQ [6]	65.2	68.7	69.6
QIL [7]	65.7	69.2	70.1
†LSQ [5]	67.6	70.2	71.1
QKD [9]	67.4	70.2	71.4
APoT [10]	67.3	69.9	70.7
LSQ+ [3]	66.8	69.3	70.8
†UniQ [11]	67.8	70.5	**71.5**
†DAQ [8]	66.9	69.6	70.5
†*LCQ [13]	66.2	70.4	71.4
LLT [12]	66.0	69.5	70.4
†**nuLSQ-W (ours)**	66.5	70.4	71.4
†**nuLSQ-A (ours)**	**67.9**	**70.7**	**71.5**

Methods	Bit-width (W/A)		
	2/2	3/3	4/4
ResNet-34 (FP: 75.1)			
LQ-Nets [14]	69.8	71.9	-
DSQ [6]	70.0	72.5	72.8
QIL [7]	70.6	73.1	73.7
†LSQ [5]	71.6	73.4	74.1
QKD [9]	71.6	73.9	74.6
APoT [10]	70.9	73.4	73.8
†UniQ [11]	72.1	74.2	75.0
†DAQ [8]	71.0	73.1	73.7
†*LCQ [13]	71.4	74.0	74.5
†**nuLSQ-W (ours)**	71.6	**74.3**	**75.3**
†**nuLSQ-A (ours)**	**72.6**	**74.3**	**75.3**

References

1. Pytorchcv. https://pypi.org/project/pytorchcv/. 4
2. timm. https://github.com/rwightman/pytorch-image-models. 4
3. 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 Workshops, pages 696–697, 2020. 7
4. Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. International Conference on Learning Representation, 2018. 7
5. Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In International Conference on Learning Representations, 2019. 2, 7
6. 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/CVF international conference on computer vision, pages 4852–4861, 2019. 7
7. 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/CVF Conference on Computer Vision and Pattern Recognition, pages 4350–4359, 2019. 7
8. Dohyung Kim, Junghyup Lee, and Bumsub Ham. Distance-aware quantization. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 5271–5280, 2021. 7

9. Jangho Kim, Yash Bhalgat, Jinwon Lee, Chirag Patel, and Nojun Kwak. Qkd: Quantization-aware knowledge distillation. arXiv preprint arXiv:1911.12491, 2019. 7
10. Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In International Conference on Learning Representations, 2020. 7
11. Phuoc Pham, Jacob A Abraham, and Jaeyong Chung. Training multi-bit quantized and binarized networks with a learnable symmetric quantizer. IEEE Access, 9:47194–47203, 2021. 5, 7
12. Longguang Wang, Xiaoyu Dong, Yingqian Wang, Li Liu, Wei An, and Yulan Guo. Learnable lookup table for neural network quantization. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 12423–12433, 2022. 7
13. Kohei Yamamoto. Learnable companding quantization for accurate low-bit neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 5029–5038, 2021. 5, 6, 7
14. Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In Proceedings of the European conference on computer vision (ECCV), pages 365–382, 2018. 7
15. Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7920–7928, 2018. 6