A. Gradient Derivation

We derive the gradient with respect to quantization parameters $\boldsymbol{\theta} = \{\boldsymbol{q}, \{\hat{g}_i\}_{i=1}^b\}$ in detail.

Gradient *w.r.t.* $\boldsymbol{q}$. Let $\boldsymbol{x}, \boldsymbol{x}_q$ be two vectors stacking values before and after quantization (x and x_q) respectively, the gradient of the loss function with respect to each entry q_k is given by

$$\frac{\partial \mathcal{L}}{\partial q_k} = \sum_{i=1}^{N} \frac{\partial \mathcal{L}}{\partial x_q^i} \frac{\partial x_q^i}{\partial q_k} \tag{1}$$

where $x_q^i = Q(x^i; \boldsymbol{\theta})$. From the definition of $Q(x^i; \boldsymbol{\theta})$ in Eqn.(3), we obtain

$$\frac{\partial x_q^i}{\partial q_k} = \begin{cases} \frac{1}{Z_U} & \text{if } k = \text{argmin}_j |\frac{1}{Z_U}(\boldsymbol{U}^\mathsf{T}\boldsymbol{q})_j - x^i| \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Hence, we have

$$\frac{\partial \mathcal{L}}{\partial q_k} = \frac{1}{Z_U} \sum_{i \in \mathcal{S}_k} \frac{\partial \mathcal{L}}{\partial x_q^i}, \tag{3}$$

where $\mathcal{S}_k = \{m \,|\, m \in [N] \text{ and } (\boldsymbol{x}_o^m)_k = 1\}$ represents a set of indexes of the values discretized to the quantization level q_k.

Remark. *For $\boldsymbol{A} \in^{m_1 \times n_1}$, $\boldsymbol{B} \in^{m_2 \times n_2}$, then*

$$\boldsymbol{B} \otimes \boldsymbol{A} = \boldsymbol{T}_{m_1,m_2}(\boldsymbol{A} \otimes \boldsymbol{B})\boldsymbol{T}_{n_1,n_2} \tag{4}$$

where $\boldsymbol{T}_{m,n} = \sum_{i=1}^{m}(e_i^\mathsf{T} \otimes \boldsymbol{I}_n \otimes \boldsymbol{e}_i) = \sum_{j=1}^{n}(\boldsymbol{e}_j \otimes \boldsymbol{I}_m \otimes \boldsymbol{e}_j^\mathsf{T})$ is the perfect shuffle permutation matrix. $\boldsymbol{e}_i$ denotes the i-th canonical vector that is the vector with 1 in the i-th coordinate and 0 elsewhere. $\otimes$ is the Kronecker product. $\boldsymbol{I}_n$ is a n-by-n identity matrix.

Gradients *w.r.t.* g_k. The gradients back-propagated through the Heaviside step function $g_k = H(\hat{g}_k)$ can be approximated by the Straight-Through Estimator (STE) (Bengio et al., 2013; Yin et al., 2019a). Denote $\tilde{\boldsymbol{U}}_k = \otimes_{t=k+1}^{K} \boldsymbol{U}_t \otimes_{t=1}^{k-1} \boldsymbol{U}_t$, $\boldsymbol{U}$ can be reformulated by Remark A as follows

$$\boldsymbol{U} = \frac{1}{Z_U}\left(\boldsymbol{T}^{k-1}_{2^{b-1},2}\right)\boldsymbol{U}_k \otimes \tilde{\boldsymbol{U}}_k \left(\boldsymbol{T}^{k-1}_{2^{b-1},2}\right)^\mathsf{T}. \tag{5}$$

Note that Z_U is also a function of g_k and $Z_U = \prod_{i=1}^{b}(2 - g_i)$, the derivative of $\boldsymbol{U}$ w.r.t. g_k can be derived as follows:

$$\begin{aligned} \frac{\partial \boldsymbol{U}}{\partial g_k} &= \frac{1}{Z_U} T^{k-1}_{2^{b-1},2}\left[\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \otimes \tilde{U}_k\right]\left(\boldsymbol{T}^{k-1}_{2^{b-1},2}\right)^\mathsf{T} \\ &\quad - \frac{1}{Z_U(2-g_k)}\boldsymbol{T}^{k-1}_{2^{b-1},2}\left[\begin{bmatrix} 1 & 1-g_k \\ 1-g_k & 1 \end{bmatrix} \otimes \tilde{\boldsymbol{U}}_k\right]\left(\boldsymbol{T}^{k-1}_{2^{b-1},2}\right)^\mathsf{T} \\ &= \frac{1}{Z_U(2-g_k)}\boldsymbol{T}^{k-1}_{2^{b-1},2}\left[\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \otimes \tilde{U}_k\right]\left(\boldsymbol{T}^{k-1}_{2^{b-1},2}\right)^\mathsf{T}. \end{aligned} \tag{6}$$

where $\boldsymbol{T}^{k-1}_{2^{b-1},2}$ is a perfect shuffle permutation matrix (Davio, 1981). From Eqn.(6), we obtain $\frac{\partial \boldsymbol{U}}{\partial g_k}|_{g_k=1} = 2^{b+1}\frac{\partial \boldsymbol{U}}{\partial g_k}|_{g_k=0}$, implying that DDQ assigns smaller gradients to those inhibited gate ($g_k = 0$). In other words, once g_k decreases to 0, it is unlikely to return to 1, making DDQ appealing to achieve mixed-precision training. With Eqn.(1) and (6), the quantization parameters of DDQ and the weights of the network can be jointly optimized by using SGD.

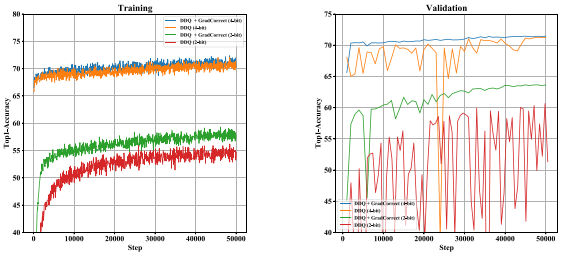

Figure 1. Training dynamics of fixed precision DDQ w/ (or w/o) gradient correction.

B. Summary of Existing Quantization Approach

Table 1 gives an overall summary of existing quantization methods. For uniform quantization methods, to reduce quantization error, (Zhou et al., 2016; Mishra et al., 2017; Choi, 2018) use tanh function to project quantization levels, but they restrict quantization levels in specific patterns. Besides, other methods such as (Choi, 2018; McKinstry et al., 2018; Jain et al., 2019), calibrate quantizer with estimated or learned centre points and thresholds, also yielding better performance. (Miyashita et al., 2016; Zhou et al., 2017; Cai et al., 2017) show that non-uniform quantization levels can outperform uniform counterparts in specific situations, and they can perform better if we learn them from data, as discussed in (Zhang et al., 2018; Jung et al., 2019). More recently, Mixed precision quantization techniques are introduced by (Wang et al., 2019; Yazdanbakhsh et al., 2018) and (Uhlich et al., 2020), further improving quantization methods by assigning different bitwidth to each layer using Reinforcement-Learning or Gradient-based methods. As shown in Table 1, the proposed DDQ can integrate main properties of above methods, learning to select optimal quantization policy according to corresponding data and model architectures.

C. Experimental Details

C.1. Evaluation on ImageNet

The ImageNet dataset consists of 1.2M training and 50K validation images. For ResNet and MobileNet, we adopt standard data preprocessing in the original paper (He et al., 2016; Sandler et al., 2018). All DNN+DDQs are trained for 30

Method	Differentiablity	Mixed Precision	Quantization Level	Step Size	Quantizer Calibration	Gradient Calibration
X-Nor-Net (Rastegari et al., 2016)	✓		UQ			
DoReFa-Net (Zhou et al., 2016)	✓		UQ+Tanh			
WPRN (Mishra et al., 2017)	✓		UQ+Tanh			
PACT (Choi, 2018)	✓		UQ+Tanh	✓	✓	
FAQ (McKinstry et al., 2018)	✓		UQ		✓	
NICE (Baskin et al., 2018)	✓		UQ		✓	
LSQ (Esser et al., 2020)	✓		UQ	✓	✓	✓
BCGD (Yin et al., 2019b)	✓		UQ			✓
TQT (Jain et al., 2019)	✓		UQ	✓	✓	✓
HAQ (Wang et al., 2019)		✓	UQ	✓	✓	
Releq (Yazdanbakhsh et al., 2018)		✓	UQ			
Uhlich *et al.* (Uhlich et al., 2020)	✓	✓	UQ/Non-UQ	✓	✓	✓
INQ (Zhou et al., 2017)			Non-UQ			
PoT (Miyashita et al., 2016)	✓		Non-UQ			
HWGQ (Cai et al., 2017)	✓		Non-UQ	✓	✓	✓
QIL (Jung et al., 2019)	✓		Learned	✓	✓	
LQ-Net (Zhang et al., 2018)	✓		Learned	✓		
DDQ (ours)	✓	✓	Learned	✓	✓	✓

Table 1. Overall summary of state-of-art quantiztaion methods. "Differentiablity" column shows whether this method can be implemented with one-stage and gradient-based methods. "UQ" and "Non-UQ" indicate uniform / non-uniform quantization respectively. "Step Size" column denotes the ability to adjust quantization step size. "Quantizer Calibration" means if the method calibrates the quantizer with centre points and thresholds. "Gradient Calibration" shows if the quantization gradients for parameters in quantizer are corrected.

epochs with cosine learning rate scheme(Loshchilov & Hutter, 2016) like (Esser et al., 2020). We choose PACT (Choi, 2018) with gradient calibration (Jain et al., 2019; Esser et al., 2020; Jin et al., 2019; Li et al., 2020) pipeline as baselines. All hyper-parameters follow prior arts such as PACT(Choi, 2018) for fair comparisons, *e.g.* l2-regularization coefficient $\lambda = 1e-2$. Network weights are quantized using uniform quantization (UQ), power-of-two quantization (PoT) and DDQ respectively, and all activations are uniformly quantized for fair comparison. For PACT, parameterized clipping values are initialized to 6.0 for activations and 3.0 for weights. We adopt per-tensor quantization for activations and per-channel quantiztaion for weights as recommended in (Rastegari et al., 2016; Krishnamoorthi, 2018; Goncharenko et al., 2019) to handle the widely-varying range between channels. Note that weights of all layers are quantized with UQ/PoT/DDQ directly except those of first and last layer, for which we employ 8-bit uniform quantization to observe standard practice of all state-of-the-art works. In addition, training DDQ will cause extra computation, but is still efficient and comparable to training UQ and PoT. Specially, training DDQ-MobileNetV2 cost 29.3 min averagely for each epoch on 8 Nvidia GTX 1080Ti, less than 5% longer than PoT (28.4 min) and UQ (27.9 min).

C.2. Gradient Correction.

For fixed-precision DDQ, we have an interesting observation that the proposed gradient correction could stabilize training. For instance, Fig. 1 illustrates training dynamics for 2/4-bit DDQ-quantized MobileNetV2. With gradient correction, the quantized model not only yields better performance (both training and validation), but also converges with less jitters in validation accuracy.

	Methods	Accuracy		
		2-bit	3-bit	4-bit
ResNet20 (FP : 92.4)	DoReFa-Ne (Zhou et al., 2016)	88.2	89.9	90.5
	PACT (Choi, 2018)	89.7	91.1	91.7
	LQ-Net (Zhang et al., 2018)	90.2	91.6	-
	SAWB (Choi et al., 2018)	90.5	-	-
	TQT (Jain et al., 2019)	91.2	-	-
	Uhlich *et al.* (Uhlich et al., 2020)	91.4	-	-
	DDQ (mixed)	**91.6**	**92.2**	**92.7**

Table 2. Comparison of Cifar-10 Top1-accuracy towards existing quantization methods. All the reported results use 32-bit activation by following prior work.

C.3. Mixed Precision Training.

Figure.D shows depicts training dynamics of bitwidth for all layers when quantizing a 4-bit ResNet18 using DDQ with maximum bitwidth 8. As demonstrated, DDQ could learn to assign bitwidth to each layer, in a data-driven manner.

D. Evaluation on Cifar-10

Additionally, we quantize ResNet20 on Cifar-10 with mixed precision. For weight quantization, we adopt 2-/3-/4-bit target bitwidth and initialize DDQ with maximum bitwidth 8. Tabel 2 compares our results with other weight-only quantization methods. For Cifar-10, all layers of the model are quantized using DDQ. The quantized models are trained for 200 epochs with learning rate 0.01, batch size 1024 and cosine scheduler.

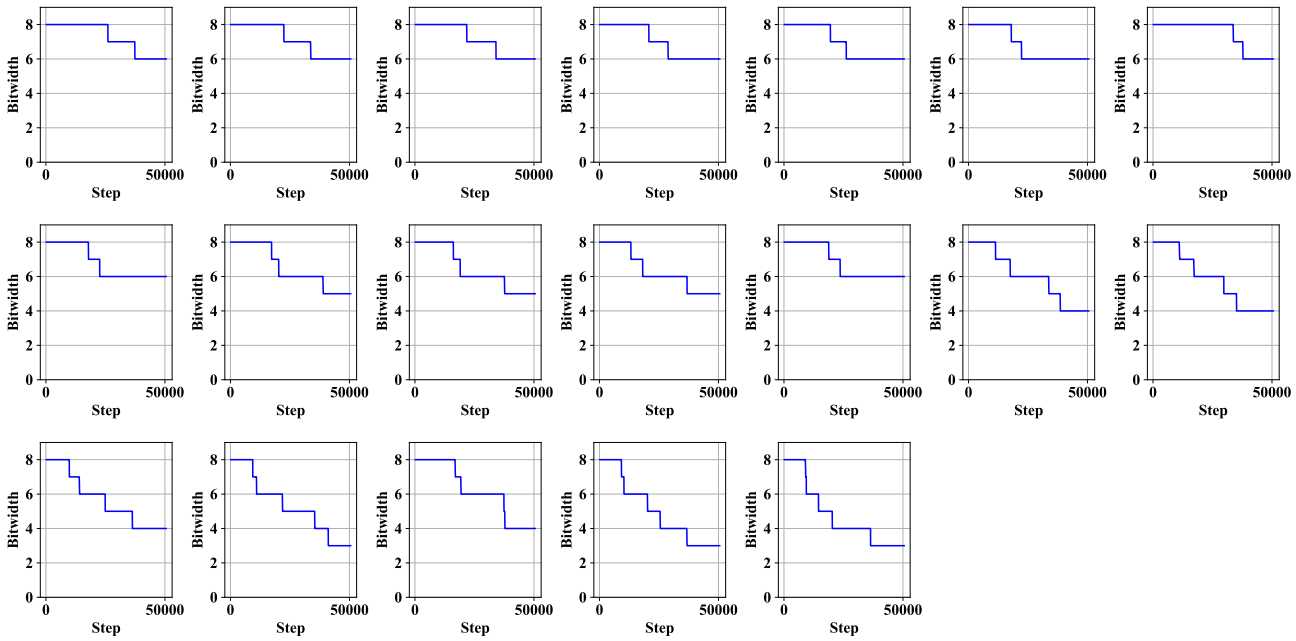

Figure 2. Evolution of bitwidth of layers when training ResNet18. We can see that DDQ can learn to assign bitwidth to each layer under given memory footprint constraints.

References

Baskin, C., Liss, N., Chai, Y., Zheltonozhskii, E., Schwartz, E., Giryes, R., Mendelson, A., and Bronstein, A. M. Nice: Noise injection and clamping estimation for neural network quantization. *arXiv preprint arXiv:1810.00162*, 2018.

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

Cai, Z., He, X., Sun, J., and Vasconcelos, N. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5918–5926, 2017.

Choi, J. Pact: Parameterized clipping activation for quantized neural networks. *arXiv: Computer Vision and Pattern Recognition*, 2018.

Choi, J., Chuang, P. I.-J., Wang, Z., Venkataramani, S., Srinivasan, V., and Gopalakrishnan, K. Bridging the accuracy gap for 2-bit quantized neural networks (qnn). *arXiv preprint arXiv:1807.06964*, 2018.

Davio, M. Kronecker products and shuffle algebra. *IEEE Transactions on Computers*, 100(2):116–125, 1981.

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. Learned step size quantization. 2020.

Goncharenko, A., Denisov, A., Alyamkin, S., and Terentev, E. Fast adjustable threshold for uniform neural network quantization. *International Journal of Computer and Information Engineering*, 13(9):499–503, 2019.

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

Jain, S. R., Gural, A., Wu, M., and Dick, C. H. Trained quantization thresholds for accurate and efficient fixed-point inference of deep neural networks. *arXiv: Computer Vision and Pattern Recognition*, 2019.

Jin, Q., Yang, L., and Liao, Z. Towards efficient training for neural network quantization. *arXiv preprint arXiv:1912.10207*, 2019.

Jung, S., Son, C., Lee, S., Son, J., Han, J.-J., Kwak, Y., Hwang, S. J., and Choi, C. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4350–4359, 2019.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Li, Y., Dong, X., and Wang, W. Additive powers-of-two quantization: A non-uniform discretization for neural networks. 2020.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

McKinstry, J. L., Esser, S. K., Appuswamy, R., Bablani, D., Arthur, J. V., Yildiz, I. B., and Modha, D. S. Discovering low-precision networks close to full-precision networks for efficient embedded inference. *arXiv preprint arXiv:1809.04191*, 2018.

Mishra, A., Nurvitadhi, E., Cook, J. J., and Marr, D. Wrpn: wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*, 2017.

Miyashita, D., Lee, E. H., and Murmann, B. Convolutional neural networks using logarithmic data representation. *arXiv preprint arXiv:1603.01025*, 2016.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Uhlich, S., Mauch, L., Cardinaux, F., Yoshiyama, K., Garcia, J. A., Tiedemann, S., Kemp, T., and Nakamura, A. Mixed precision dnns: All you need is a good parametrization. In *International Conference on Learning Representations*, 2020.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8612–8620, 2019.

Yazdanbakhsh, A., Elthakeb, A. T., Pilligundla, P., Mireshghallah, F., and Esmaeilzadeh, H. Releq: An automatic reinforcement learning approach for deep quantization of neural networks. 2018.

Yin, P., Lyu, J., Zhang, S., Osher, S., Qi, Y., and Xin, J. Understanding straight-through estimator in training activation quantized neural nets. *International Conference on Learning Representations*, 2019a.

Yin, P., Zhang, S., Lyu, J., Osher, S., Qi, Y., and Xin, J. Blended coarse gradient descent for full quantization of deep neural networks. *Research in the Mathematical Sciences*, 6(1):14, 2019b.

Zhang, D., Yang, J., Ye, D., and Hua, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 365–382, 2018.

Zhou, A., Yao, A., Guo, Y., Xu, L., and Chen, Y. Incremental network quantization: Towards lossless cnns with low-precision weights. *International Conference on Learning Representations*, 2017.

Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.