Supplemental Material of *Rate-Distortion Optimized Post-Training Quantization for Learned Image Compression.*

A MORE DETAILS

A.1 ROUNDING STRATEGY

Usually rounding-to-nearest (i.e., $\lfloor \cdot \rceil$) is the method used the most. Numerous works (e.g., AdaRound (Nagel et al., 2020), AdaQuant (Hubara et al., 2020), BRECQ (Li et al., 2020)) have realized that rounding-to-nearest is not the optimal choice, and it is important to adopt a better rounding strategy. These new rounding strategies perform better than rounding-to-nearest. A brief introduction about rounding strategies is given below, more details can be found in their papers.

In AdaRound, the weights are initially rounded by floor rounding (i.e., $\lfloor \cdot \rfloor$). v is the continuous variable and $h(\boldsymbol{v})$ is a differentiable function that takes values between 0 and 1, i.e., $h(v) \in [0, 1]$.

$$\hat{\boldsymbol{w}} = s_{\boldsymbol{w}} \cdot clip(\lfloor \frac{\boldsymbol{w}}{s_{\boldsymbol{w}}} \rfloor + h(\boldsymbol{v}), n, p). \tag{1}$$

Besides, a differentiable regularizer $f_{reg}(\boldsymbol{v})$ is introduced to encourage the optimization variables $h(\boldsymbol{v})$ to converge towards either 0 or 1. The optimization problem together with the regularization is given by

$$\arg\min_{\boldsymbol{v}} \|\widehat{\boldsymbol{w}}\boldsymbol{x} - \boldsymbol{w}\boldsymbol{x}\|^2 + \lambda f_{reg}(\boldsymbol{v}). \tag{2}$$

Different from AdaRound, AdaQuant uses a wider range to round, where $\boldsymbol{v}$ is a continuous variable added to $\boldsymbol{w}$ and the quantized network weights are defined as $\hat{\boldsymbol{w}} = Q(\boldsymbol{w} + \boldsymbol{v})$. In this new objective the quantized tensor is not required to be "close" to the original tensor. We use similar approach to further reduce the loss of quantization.

A.2 WHY INT8?

The smaller the bit-width of the fixed-point quantization model is, the smaller the model storage is and the greater the execution speed is. However, as the bit width decreases, the performance of the fixed-point quantization model is more likely to decline and the risk of overflow is also greater. Combined with these, INT8 has been a very common quantization accuracy in the industry (Krishnamoorthi, 2018; Wu et al., 2020). Even if 8-bit integer arithmetic is not supported, quantizing the weight to the INT8 model size can be reduced by 4 times. INT4 accuracy still needs to test whether the performance decline of model after quantization is acceptable (Xilinx, 2020).

So we choose INT8 as our target. In addition, we present the results of INT10 quantization later.

A.3 BIAS QUANTIZATION

Generally, in quantization of neural network models, bias is stored as FP32 (most commonly used in fake quantization) or as INT32 (Nagel et al., 2021). However, bias of 32-bit is easy to cause data overflow in the same 32-bit accumulator. As a simple example, INT8's weight and activation are convolved to get INT16's feature map. When bias is mapped to INT32 by Min-Max, overflow occurs in the 32-bit accumulator where the bias is added to the feature map.

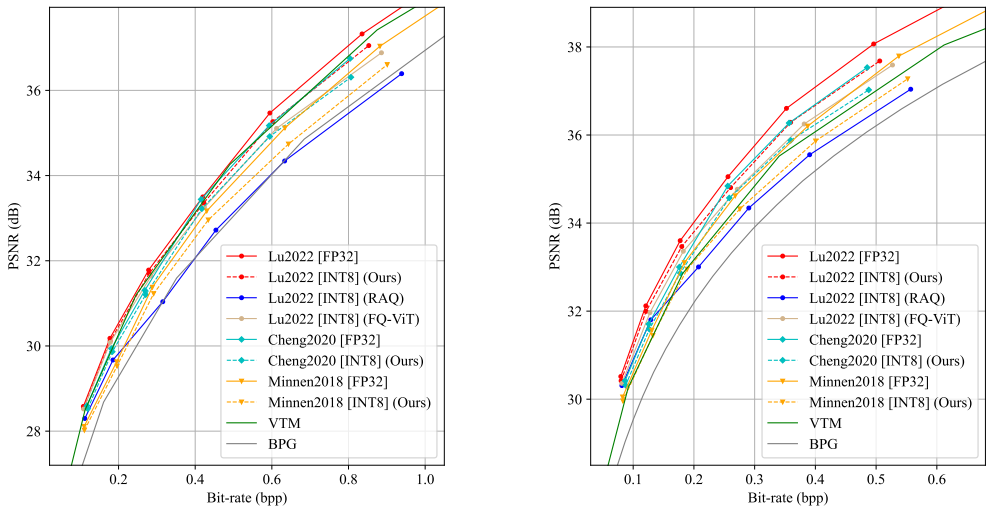
Therefore, although bias is stored as 32-bit in many works, the value range is smaller than 32-bit. For example, in Jacob et al. (2018), the quantization scale of bias S_{bias} is the product of the scales of the weight and of the input activation: $S_{bias} = S_w S_a$.

Bias actually controls the state of neuron activation. In LIC, setting bias as zero would degrade the model performance significantly. For example, after discarding bias of convolution layers, average PSNR on Kodak dropped by more than 12dB on Lu2022. Whether to discard bias needs to consider the distribution of data and the nature of the activation function.

In this paper, we want bias to achieve homogenous bit-width with weight and activation. So we proposal bias rescaling by which perhaps a 24-bit or less accumulator will be enough.

A.4 RD PERFORMANCE CORRESPONDING TO TABLE 1

We plot R-D curves in Fig. 1 for various floating-point LIC models and their quantized INT8 counterparts.



(a) (b)

Figure 1: **R-D Performance.** (a) Kodak dataset; (b) Tecnick dataset; VTM stands for the VVC Intra while BPG is the HEVC Intra. Methods marked with [FP32] represent their original 32-bit floating-point models; while methods highlighted with [INT8] are quantized models using 8-bit fixed-point precision for processing.

B MSE OPTIMIZATION

B.1 METHOD

The MSE optimization is to minimize the mean square error between the full-precision and the quantized tensor $\boldsymbol{x}$ (weights, bias or activation). The optimized scaling factors are obtained by

$$s_{\boldsymbol{x}} = \arg\min_{s_{\boldsymbol{x}}} \|\boldsymbol{x} - Q(\boldsymbol{x})\|^2. \tag{3}$$

$Q(\cdot)$ is the quantization function mapping $\boldsymbol{x}$ from the floating-point domain to the fixed-point domain,

$$Q(\boldsymbol{x}) = s_{\boldsymbol{x}} \cdot clip(\lfloor \frac{\boldsymbol{x}}{s_{\boldsymbol{x}}} \rceil). \tag{4}$$

The scaling factor calculation formula simply is expressed as follows,

$$s_{\boldsymbol{x}} = \frac{r_{\boldsymbol{x}}}{2^{b-1} - 1}, \tag{5}$$

where $r_{\boldsymbol{x}}$ represent the dynamic range of $\boldsymbol{x}$. For symmetric quantization with Min-Max approach, $r_{\boldsymbol{x}} = \max(|\boldsymbol{x}|)$. We multiply by an additional continuous variable N to obtain the scaling factor for the minimum MSE, i.e., $r_{\boldsymbol{x}} = N \cdot \max(|\boldsymbol{x}|)$. Then, the scaling factors can be represented as

$$s_{\boldsymbol{x}} = N \cdot \frac{\max(|\boldsymbol{x}|)}{2^{b-1} - 1}. \tag{6}$$

The objective of MSE optimization is to obtain the corresponding N under minimum MSE. N can be optimized layer-wisely or channel-wisely.

B.2 COMPARISON OF R-D OPTIMIZATION AND MSE OPTIMIZATION

Fig. 4 and Fig. 5 of main paper shows the R-D performance on Lu2022 by MSE optimization and R-D optimization respectively. Smaller quantization errors do not lead to better performance. The difference becomes more pronounced as BPP rises.

Compared with the native floating-point model, the BD-rate loss is about 7.82% under MSE optimization, which is significantly higher than the results shown in the Table 1.

B.3 DISCUSSION

Although MSE optimization is convenient to use, it has recently been recognized that this method may lead to the sub-optimality. Nagel et al. (2020) shows that rounding-to-nearest (e.g., Choi et al. (2018)) is not the best way to use and propoes the AdaRound, an adaptive rounding mechanism. Hubara et al. (2020) believes that MSE optimization penalizes all the quantization errors equally. However, the loss should penalize more quantization errors which more affect the classification. Li et al. (2020) analyses the second-order error and proposes the block-wise optimization, which is conducted for both image classification and object detection tasks. In this paper, we study the MSE optimization for the Learned Image Compression, and show the nondeterministic relationship between MSE and Rate-Distortion performance, suggesting that MSE optimization is not a good choice in tasks of Learned Image Compression.

C COMPARED WITH METHODS NEEDING RETRAINING

We compared our method with two SOTA methods (Sun et al., 2020; 2022), both of which require retraining. Unfortunately, we did not find any publicly accessible code, so we tried to reproduce experimental results according to the setup of original papers.

C.1 EXPERIMENT SETTINGS

For each model, six pre-trained models are experimented with six different λs, e.g., $\{0.0018, 0.0035, 0.0067, 0.013, 0.025, 0.0483\}$. We choose the DIV2K (Timofte et al., 2017) as the training dataset to fine-tune models in Sun et al. (2020; 2021). Image samples are randomly cropped into fixed patches at a size of 256 × 256 × 3. All training threads run on a single 1080Ti GPU for 160 epochs in total, having the learning rate at 10^{-4} initially, and then at 10^{-5} after 80 epochs. In our method, we set the learning rate at 10^{-3} initially and then dynamically adjust the learning rate. We evaluate on Lu2022.

C.2 EXPERIMENTAL RESULTS

Performance comparisons are presented in the main paper as part of the ablation experiment (Section 5.3). Here we show the approximate running time of the different algorithm. The specific running time is related to data loading and writing, and algorithm logic. We give a reference result in the following table.

Table 1: Optimization time on Lu2022

	λ	channel numbers	optimization running		
			Ours	Sun et al. (2020)	Sun et al. (2021)
low bit rate	0.0018-0.0130	(128,192)	8h	30h	35h
high bit rate	0.0250-0.0483	(192,320)	11h	51h	56h

Compared to Sun et al. (2020; 2021), our method reduces the time by about 78%. Although it still takes a few hours to optimize the quantization parameters (e.g., scale factors), the model parameters (e.g., weight and bias) do not need to be fine-tuned and we pay less attention to the structure of the model, which makes optimization easier.

The bottlenecks that limit our algorithm are: 1) optimization layer-wisely makes it hard to compute in parallel; 2) after each update of quantization parameters, the model needs to be re-quantized. This might be improved by storing more intermediate results and refining the algorithm logic. We will work on this in the future.

D Evaluation on MS-SSIM loss trained models

Results for MS-SSIM loss trained models are provided in Fig. 2 . For each model, six bitrates are experimented by setting six different λs, e.g., $\{2.40, 4.58, 8.73, 16.64, 31.73, 60.60\}$.

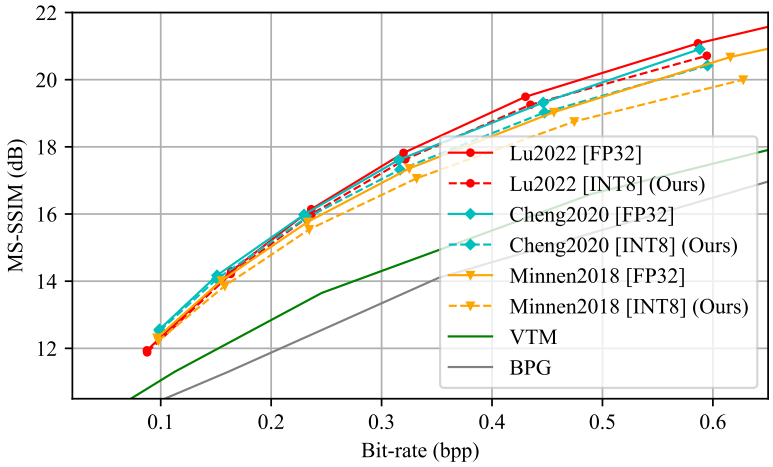


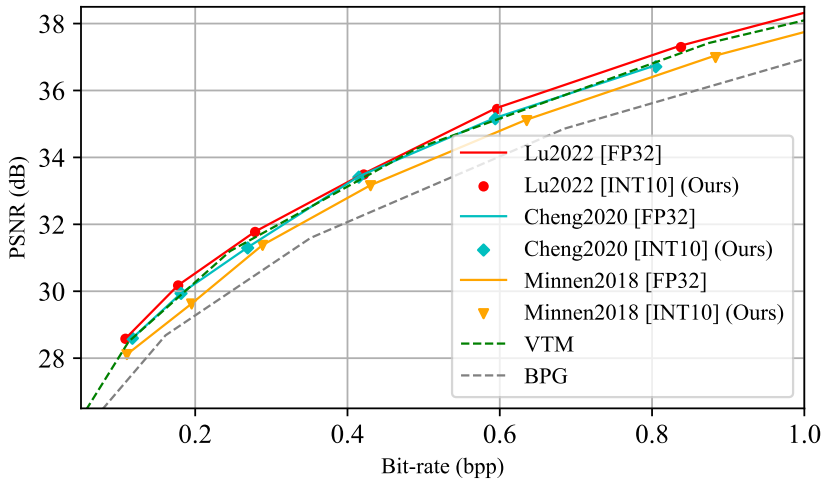
Figure 2: R-D performance of MS-SSIM models on Kodak.

E Evaluation under 10-bit

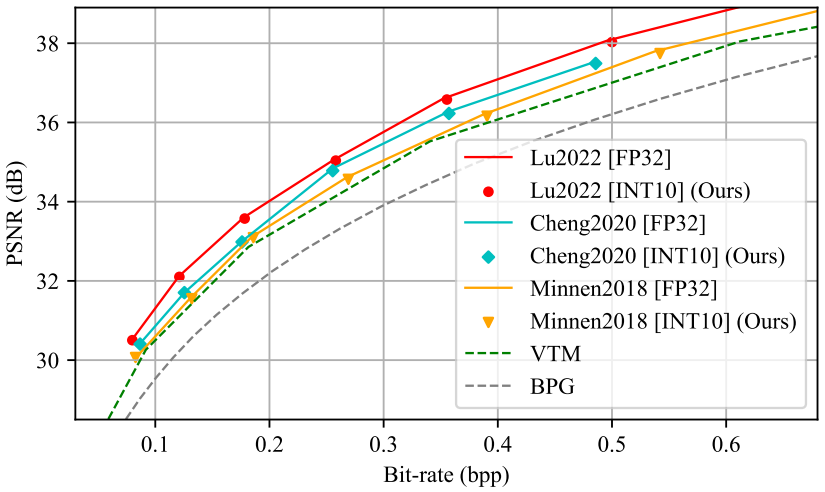
We plot R-D curves in Fig. 3 for various floating-point LIC models and their quantized INT10 counterparts, and further report the BD-rate performance over different 32-bit floating-point models in Table 2. As seen, quantized INT10 LICs achieves similar performance as its native FP30 models with negligible loss.

Table 2: BD-rate loss over floating-point models

	Lu2022	Cheng2020	Minnen2018
Kodak	0.49%	0.43%	0.41%
Tecnick	1.03%	0.65%	1.22%



(a) Kodak dataset



(b) Tecnick dataset

Figure 3: **R-D Performance.** VTM stands for the VVC Intra while BPG is the HEVC Intra. Methods marked with [FP32] represent their original 32-bit floating-point models; while methods highlighted with [INT10] are quantized models using 10-bit fixed-point precision for processing.

REFERENCES

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:1805.06085*, 2018.

Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518*, 2020.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

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

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. In *International Conference on Learning Representations*, 2020.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pp. 7197–7206. PMLR, 2020.

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

Heming Sun, Zhengxue Cheng, Masaru Takeuchi, and Jiro Katto. End-to-end learned image compression with fixed point weight quantization. In *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3359–3363. IEEE, 2020.

Heming Sun, Lu Yu, and Jiro Katto. Learned image compression with fixed-point arithmetic. In *2021 Picture Coding Symposium (PCS)*, pp. 1–5. IEEE, 2021.

Heming Sun, Lu Yu, and Jiro Katto. Q-lic: Quantizing learned image compression with channel splitting. *arXiv preprint arXiv:2205.14510*, 2022.

Radu Timofte, Eirikur Agustsson, Luc Van Gool, Ming-Hsuan Yang, and Lei Zhang. Ntire 2017 challenge on single image super-resolution: Methods and results. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 114–125, 2017.

Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

Xilinx. Convolutional neural network with int4 optimization on xilinx. In *WP521 (v1.0.1) June 24*, 2020.