Supplementary Material for "Learnable Companding Quantization for Accurate Low-bit Neural Networks"

A. Relations between LCQ and the conventional methods

We clarify the difference between our proposed method and some similar conventional methods as follows: LSQ [4] uses a learnable clipping function similar to our method, however, they use a uniform quantization function. Therefore, the quantization levels are not learnable. Since the input distribution of DNNs is usually not uniform, non-uniform quantization is better than uniform quantization to reduce the quantization error. QIL [12] uses also the uniform quantization method that non-linearly transforms the input (corresponding to our "compressing function") and then uniformly quantizes it, however, it does not apply the expanding function as we proposed. Without the expanding function, the quantization error is likely to be large. APoT [16] uses non-uniform quantization, however, their quantization levels are not learnable. DQ [25] learns non-uniform quantization levels, however, their levels are optimized with simple heuristic gradients, while our levels are optimized with gradients based on the derivative of the companding function. LQ-Nets [30] also learns non-uniform quantization levels, however, it does not use gradients to optimize the levels, unlike our method.

B. Training algorithm for LCQ

When training quantized DNNs with LCQ, we independently apply the LCQ quantizer to the weights and activations for the convolutional or fully-connected layers. Algorithm S1 summarizes the LCQ training procedure for a convolutional layer as an example. Note that "$*$" denotes a convolutional operation, and that the LCQ parameters are given independently on a layer-by-layer basis.

C. Validity of comparing LCQ and uniform quantization methods

Our method assumes that multiplication is replaced by memory access to LUTs during inference, and the speed of the memory access depends on an efficient hardware accelerator design. Therefore, with respect to the comparison between the proposed and conventional methods, it is difficult to evaluate theoretical metrics (*e.g.*, FLOPs) for computational efficiency, and also to evaluate the actual speedup without dedicated hardware support. However, since there is almost no difference between our method and conven-

Algorithm S1 Training a convolutional layer with LCQ.

Input: full precision weights w and full precision inputs/activations a, and the corresponding parameters: the clipping parameters (α_w, α_a), the companding parameters (θ_w, θ_a), the bit-widths (b_w, b_a) and the outer bit-widths (b'_w, b'_a).

Output: updated parameters w, α_w, α_a, θ_w and θ_a.

1: Compute the quantized weights using Eq. (13) and Eq. (14): $w_q \leftarrow \text{Quantize}(w, \alpha_w, \theta_w, b_w, b'_w)$.
2: Compute the quantized activations using Eq. (14): $a_q \leftarrow \text{Quantize}(a, \alpha_a, \theta_a, b_a, b'_a)$.
3: Compute the convolution outputs: $y \leftarrow w_q * a_q$.
4: Compute the loss $\mathcal{L}$ and the gradients $\frac{\partial \mathcal{L}}{\partial y}$.
5: Compute the gradients for the weights $\frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial w}$.
6: Compute the gradients for the clipping parameters $\frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial \alpha_a}$ and $\frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial \alpha_w}$ based on Eq. (12).
7: Compute the gradients for the companding parameters $\frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial \theta_a}$ and $\frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial \theta_w}$ based on Eq. (7) and Eq. (11).
8: Update w, α_a, α_w, θ_w and θ_a with the corresponding gradients, respectively.

Table S1: Comparison of memory usage with/without LUT in bytes. b_w/b_a indicates the bit-width for weights and activations, respectively.

Model	b_w/b_a	w/ LUT	w/o LUT	Diff.
ResNet-18 (44.59 MB in FP32)	2/2	3.19 MB	3.19 MB	114 B
	3/3	4.52 MB	4.52 MB	798 B
	4/4	5.85 MB	5.85 MB	3990 B
ResNet-34 (83.15 MB in FP32)	2/2	5.63 MB	5.63 MB	210 B
	3/3	8.16 MB	8.16 MB	1470 B
	4/4	10.70 MB	10.69 MB	7350 B
ResNet-50 (97.46 MB in FP32)	2/2	7.73 MB	7.73 MB	312 B
	3/3	10.52 MB	10.52 MB	2184 B
	4/4	13.33 MB	13.32 MB	10920 B
MobileNet-V2 (13.37 MB in FP32)	4/4	2.41 MB	2.40 MB	10920 B

tional methods in terms of memory usage, we compare them in terms of accuracy at the same bit-widths. This accuracy comparison is worthwhile because it allows us to evaluate the model's portability to memory-constrained devices.

We then show the additional memory usage by LUTs is almost negligible. For the models and bit-widths used in the experiments in this paper, Table S1 shows the memory usage of the LCQ models (w/ LUT) and the uniform quantization models (w/o LUT). Note that we set 8 as the outer bit-width for both weights and activations. Clearly, there is almost no difference in their memory usage for all the combinations of the models and the bit-widths.