Appendix for ‘GENIE: Show Me the Data For Quantization’

A. Implementation Details

For all experiments related to data distillation, we set the batch size to 128 and used the Adam [4] optimizer with an initial learning rate of 0.01 and 0.1 for the generator and latent vectors, respectively. The learning rate for the generator decays exponentially by gamma (= 0.95) every 100 steps, whereas the learning rate for the latent vectors is scheduled with ”*ReduceLROnPlateau*” like that in ZeroQ [1]. For all experiments, we distilled 1024 images, which were used for quantization. Each batch was independently distilled, and the weights of the generator were shared only within a batch. In other words, the weights of the generator are initialized when distilling another batch.

For model distillation, we set the batch size to 32 and used the Adam optimizer to train the quantization parameters, namely, the scaling factor s_w, *softbit* $\boldsymbol{V}$, and step size s_a of activations, the initial learning rates for which were 0.0001, 0.001, and 0.00004, respectively. We also used cosine annealing [6] to decay the learning rate to 0 for the scaling factors of weights and step size of activations during optimization. We obtained pre-trained models from public repositories[1,2].

B. Block-Wise Optimization

To optimize the quantized models, we minimize the reconstruction error between two blocks, which is sequentially performed from the input layers as follows:

$$\underset{s_w, s_a, \boldsymbol{V}}{\arg\min} \|\boldsymbol{z} - \boldsymbol{z}^q\|_2^2 , \tag{A1}$$

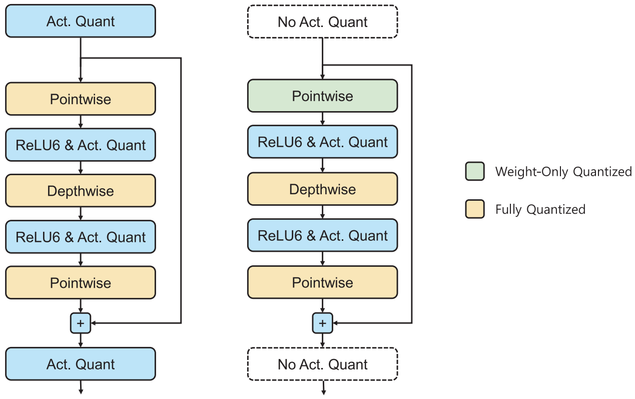
where $\boldsymbol{z}$ and $\boldsymbol{z}^q$ are the outputs of the two blocks in the pre-trained *teacher* and quantized *student* models, respectively. Subsequently, we ensure that the *softbits* $h(\boldsymbol{V})$[3] takes 0 or 1 by adding the regularization term to Eq. (A1). Namely,

$$\underset{s_w, s_a, \boldsymbol{V}}{\arg\min} \|\boldsymbol{z} - \boldsymbol{z}^q\|_2^2 + \lambda \sum_{i,j} (1 - \left|2h(\boldsymbol{V}_{i,j}) - 1\right|^{\beta}), \tag{A2}$$

where $h(\cdot)$ denotes the rectified sigmoid function [7] and β is annealed during optimization like that in AdaRound [8].

Algorithm A1 Layer-wise reconstruction for quantization

```
Input: Full-precision weights $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ and input activations $\boldsymbol{x}$
Output: Quantized layer
procedure GENIE-M($\boldsymbol{W}, \boldsymbol{x}, bits$)
    WeightQuant←GENIE-M($\boldsymbol{W}, bits$)                ▷ Algorithm 2
    $s_a$=Initialize($\boldsymbol{x}_1$)          ▷ Init. step size of act. by 1st batch
    for each input $\boldsymbol{x}$ do
        $\boldsymbol{y} = \boldsymbol{W} \cdot \boldsymbol{x}$
        $\boldsymbol{x}^q \leftarrow s_a \cdot \left\lceil \frac{\boldsymbol{x}}{s_a} \right\rfloor$                       ▷ Act. quant. using LSQ [3]
        $\boldsymbol{W}^q \leftarrow$ WeightQuant( )
        $\hat{\boldsymbol{y}} \leftarrow \boldsymbol{W}^q \cdot \boldsymbol{x}^q$
        $\mathcal{L} \leftarrow ||\hat{\boldsymbol{y}} - \boldsymbol{y}||_2^2 + \lambda f_{reg}(\boldsymbol{V})$                    ▷ See Eq. (A2)
        $\mathcal{L}$.backward()  ▷ Update $s_w$, $s_a$, and $\boldsymbol{V}$ with respect to $\mathcal{L}$
```



1) GENIE, BRECQ, QDROP 2) GDFQ, AIT

Figure A1. Comparison of quantization points in the inverted residual block of MobileNetV2, where AIT does not quantize the output of the block.

The Lagrange multiplier λ in Eq. (A2) is set to 1.0 or 0.1 for all experiments involving GENIE-M, QDROP [11] and BRECQ [5], respectively. Algorithm A1 summarizes our quantization approach, where we assume that a block consists of one layer for a concise explanation (*i.e*., layer-wise optimization). In practice, we designate a block (that consists of consecutive layers) as a residual block, like that in QDROP and BRECQ.

C. Quantization Setting

When quantizing models into WwAa, QDROP quantizes the weights and activations into w and a, respectively, ex-

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

[2] https://github.com/osmr/imgclsmob

[3] For a brief explanation, we notated *softbits* as only $\boldsymbol{V}$ with the sigmoid function $h(\cdot)$ omitted, in the manuscript.

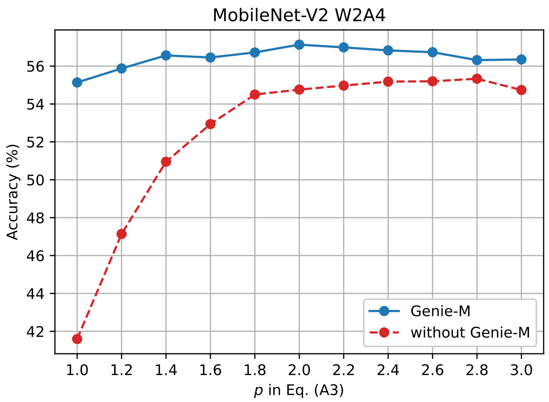


Figure A2. Comparison of accuracy depending on the p in Eq. (A3).

cluding the first and last layers where the weights of the first and last and input activations of the last layers are quantized into 8-bit fixed-point numbers. BRECQ additionally quantizes the output activations of the first layer into 8-bit while the other weights and activations are the same as in QDROP. In contrast, AIT [2] quantizes the weights and activations of all layers (including the first and last layers) into w and a, respectively. AIT quantizes activations only after the activation functions, so quantizing activations is often omitted when there is no activation function at the end of the residual block, as in MobileNetV2 [9] and MnasNet [10] (Figure A1). The methods depicted in Tables 2 and 3 follow the settings of BRECQ, whereas those depicted in Tables 4 and 5 follow the setting of AIT and QDROP, respectively.

D. Effect of the Initial Step Size

Figure A2 shows the performance of the quantized models depending on p used when the step size of weights is initialized as follows:

$$s^* = \arg\min_{s} \left\| \boldsymbol{W} - s \cdot clip\left(\left\lfloor \frac{\boldsymbol{W}}{s} \right\rceil, \text{n}, \text{p} \right) \right\|_{p,p} . \tag{A3}$$

Because QDROP and BRECQ maintain the initial step size of weights during optimization, the initialized step size can affect the performance of the quantized models. In contrast, GENIE-M learns the step size, and thus the initial step size has a negligible impact on the accuracy.

E. Generator

The generator used in GENIE-D is modified based on the generator of GDFQ [12], and it accepts latent vectors of size 256 as inputs. To reduce dependency on the generator, we use only one upsampling block, which performs the following sequence of operations: ”Upsampling-Conv2D-BatchNorm-LeakyReLU”. In contrast, GDFQ uses two upsampling blocks with latent vectors of size 100 as inputs

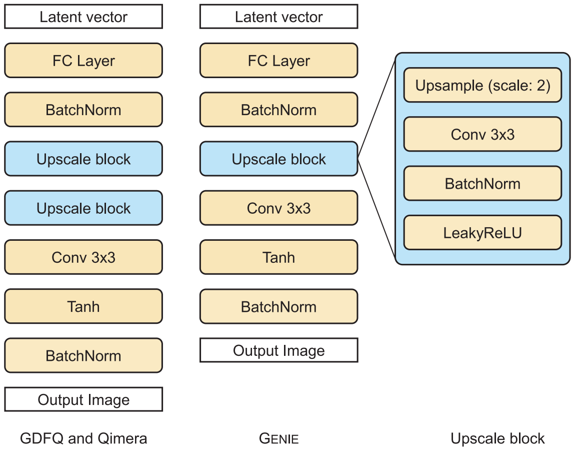
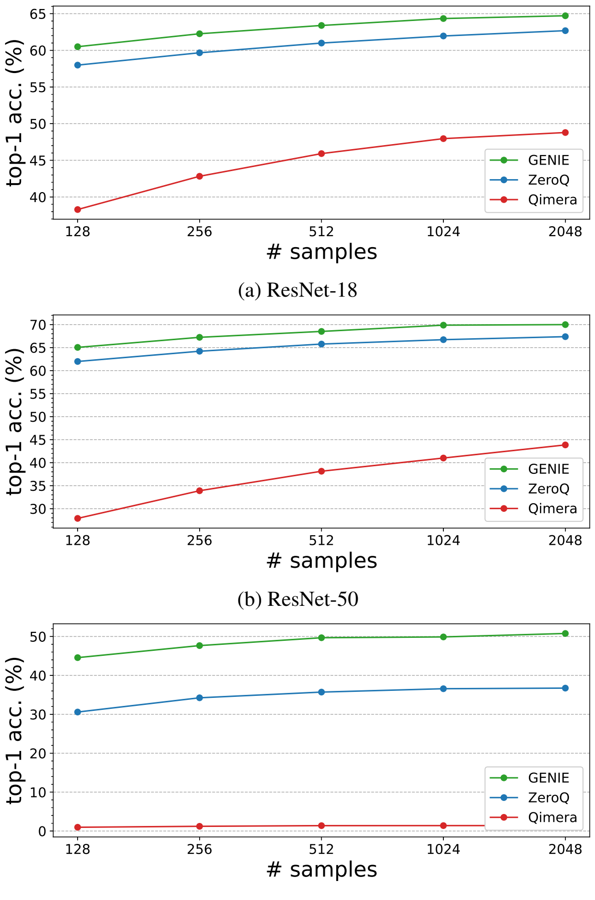


Figure A3. Structure of the generator and upscale block



(a) ResNet-18

(b) ResNet-50

(c) MobileNetV2

Figure A4. The influence of the number of samples on model accuracy (W2A4)

Table A1. The influence of the number of samples on model accuracy (W2A4)

	ResNet-18			ResNet-50			MobileNetV2		
# samples	ZeroQ	Qimera	GENIE	ZeroQ	Qimera	GENIE	ZeroQ	Qimera	GENIE
128	57.99	38.29	60.50	61.98	27.25	65.04	30.59	0.97	44.57
256	59.67	42.82	62.26	64.21	33.23	67.23	34.25	1.22	47.66
512	61.00	45.92	63.39	65.76	38.10	68.51	35.71	1.38	49.69
1024	61.96	47.96	64.34	66.72	41.00	69.87	36.58	1.40	49.89
2048	62.68	48.79	64.72	67.37	43.85	69.99	36.74	1.40	50.78

(Figure A3). Intuitively, increasing the size of the latent vectors could produce diverse data while a deeper generator could help in learning more *common knowledge* of the input domain. However, the performance of the quantized models does not highly depend upon the depth of the generator and the size of the latent vectors in our experiments.

F. Informativeness of Synthetic Data

To measure the informativeness of the synthetic data, we conduct experiments on the influence of the number of samples on model accuracy, where we identify how much information the distilled data provide when quantizing networks. As shown in Table A1 and Figure A4 (where we use QDROP as the quantizer), the distilled data by GENIE-D is more contributing to the enhancement of quantized networks. Especially, the quantization results with 128 images from GENIE-D outperform those with 1K images from Qimera throughout all models. In other words, a small quantity of images produced by GENIE-D provides more helpful information with quantized networks, compared with other methods. Thus, we can consider that the images by GENIE-D are more informative to model quantization than those by other methods.

G. Comparison on Convergence

Figure A5 compares the trace of the BNS loss (Eq. (5)) for the three approaches: ZeroQ distills knowledge to the images directly. GBA uses the generator using the Gaussian noise as the input to synthesize samples. GENIE-D distills knowledge into the latent vectors while optimizing the generator. Unlike GBA, by training latent vectors, the loss of GENIE-D converges to a lower loss than that of GBA; however, it is not lower than that of ZeroQ in spite of its better quantization performance. This implies that learning *common knowledge* or image prior by the generator is as important as achieving a low loss.

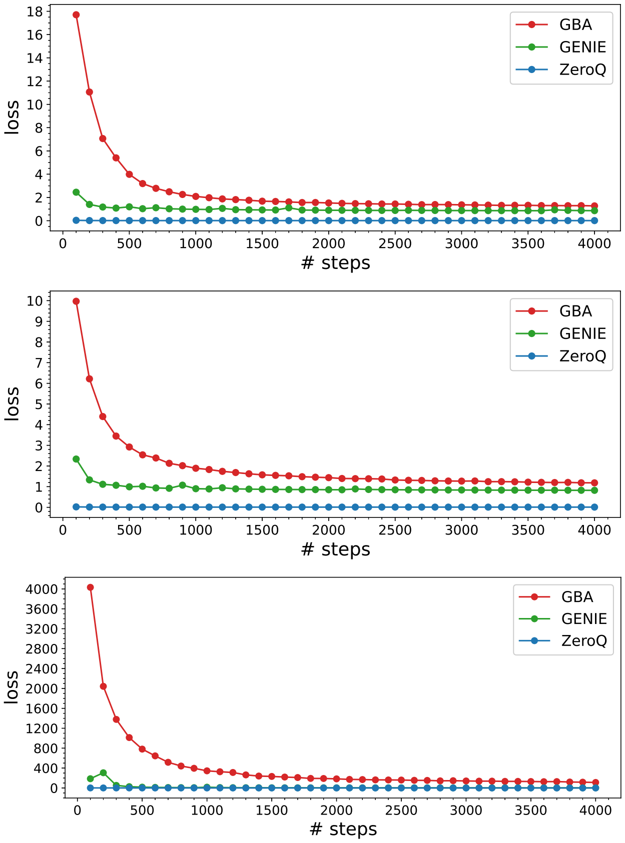


Figure A5. The trace of the BNS loss for the three approaches: ZeroQ, GBA, and GENIE

H. PTQ vs. QAT

To evaluate the performance of the data distilled by GENIE-D with QAT, we adopt AIT as the quantizer and vary the number of samples to identify how much data to need for QAT. As shown in Table A2, the size of the synthetic data does not significantly affect the performance of quantized networks. As well, the results show poor performance rather than that of PTQ with only $1K$ images. Considering both the performance and time for image generation and training (including time for hyperparameter searching), PTQ is more efficient and suitable for ZSQ. Existing works train only the generator, and thus can generate data infinitely. Because they use $80K$ steps with 16 batches for QAT, they generate a total of $1.28M$ during the training.

	Methods	#Bits (W/B)	#Synthetic dataset	Top-1 Accuracy(%)
-	Full Prec.	32/32	-	71.47
QAT	GDFQ+AIT	4/4	$1.28M$	65.51
	ARC+AIT			65.73
	Qimera+AIT			66.83
	GENIE-D+AIT		$1K$	63.98
			$5K$	65.88
			$10K$	66.55
			$20K$	66.67
			$100K$	66.91
PTQ	GENIE [ours]		$1K$	**68.51**

Table A2. Comparison between PTQ and QAT on ResNet-18

References

[1] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. ZeroQ: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13169–13178, 2020. 1

[2] Kanghyun Choi, Hye Yoon Lee, Deokki Hong, Joonsang Yu, Noseong Park, Youngsok Kim, and Jinho Lee. It's all in the teacher: Zero-shot quantization brought closer to the teacher. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8311–8321, 2022. 2

[3] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2019. 1

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015. 1

[5] 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 (ICLR)*, 2021. 1

[6] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations (ICLR)*, 2017. 1

[7] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations (ICLR)*, 2018. 1

[8] 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 (ICML)*, pages 7197–7206, 2020. 1

[9] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. 2

[10] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. MnasNet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2820–2828, 2019. 2

[11] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. QDrop: Randomly dropping quantization for extremely low-bit post-training quantization. In *International Conference on Learning Representations (ICLR)*, 2021. 1

[12] Shoukai Xu, Haokun Li, Bohan Zhuang, Jing Liu, Jiezhang Cao, Chuangrun Liang, and Mingkui Tan. Generative low-bitwidth data free quantization. In *European Conference on Computer Vision (ECCV)*, pages 1–17. Springer, 2020. 2