This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

GENIE: Show Me the Data for Quantization

Yongkweon Jeon* Chungman Lee* Ho-young Kim*
Samsung Research
{dragwon.jeon, chungman.lee, hoyoung4.kim}@samsung.com

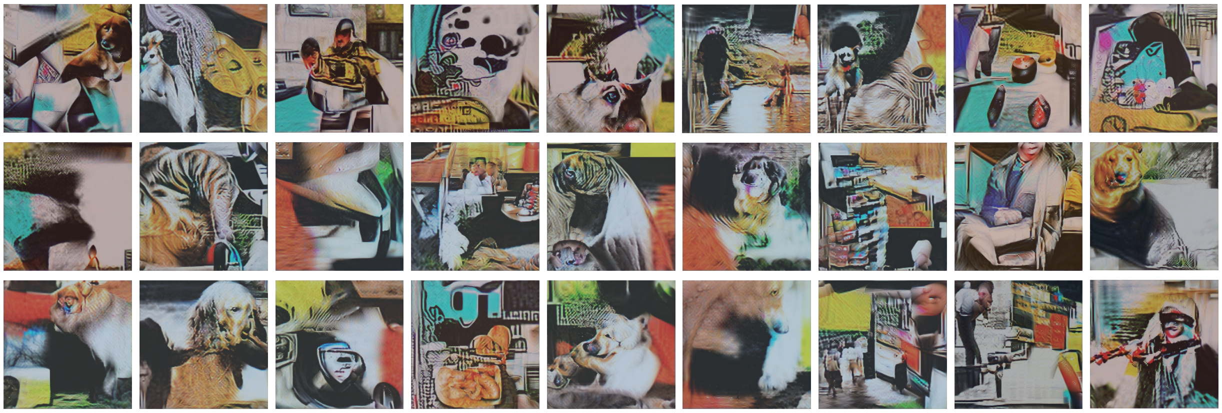


Figure 1. Distilled images by GENIE (without any image prior loss).

Abstract

Zero-shot quantization is a promising approach for developing lightweight deep neural networks when data is inaccessible owing to various reasons, including cost and issues related to privacy. By exploiting the learned parameters (μ and σ) of batch normalization layers in an FP32-pre-trained model, zero-shot quantization schemes focus on generating synthetic data. Subsequently, they distill knowledge from the pre-trained model (teacher) to the quantized model (student) such that the quantized model can be optimized with the synthetic dataset. However, thus far, zero-shot quantization has primarily been discussed in the context of quantization-aware training methods, which require task-specific losses and long-term optimization as much as retraining. We thus introduce a post-training quantization scheme for zero-shot quantization that produces high-quality quantized networks within a few hours. Furthermore, we propose a framework called GENIE that generates data suited for quantization. With the data synthesized by GENIE, we can produce robust quantized models without real datasets, which is comparable to few-shot quantization. We also propose a post-training quantization algorithm to enhance the performance of quantized models. By combining them, we can bridge the gap between zero-shot and few-shot quantization while significantly improving the quantization performance compared to that of existing approaches. In other words, we can obtain a unique state-of-the-art zero-shot quantization approach. The code is available at https://github.com/SamsungLabs/Genie.

1. Introduction

Quantization is an indispensable procedure for deploying models in resource-constrained devices such as mobile phones. By representing tensors using a lower bit width and maintaining a dense format of tensors, quantization reduces a computing unit to a significantly smaller size compared to that achieved by other approaches (such as pruning and low-rank approximations) and facilitates massive data parallelism with vector processing units. Most early studies utilized quantization-aware training (QAT) schemes [8, 23] to compress models, which requires the entire training dataset and takes as much time as training FP32 models. However, access to the entire dataset for quantizing models may not be possible in the real world or industry owing to a variety of reasons, including issues related to privacy preservation. Thus, recent studies have emphasized post-training quantization (PTQ) [12, 14, 17, 21] because it serves as a convenient method of producing high-quality quantized networks

[*] Equal contribution. Correspondence to: *dragwon.jeon@samsung.com*

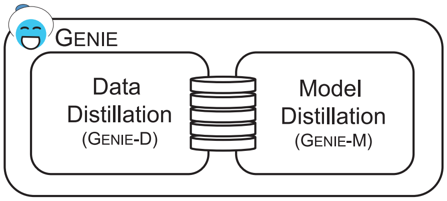


Figure 2. Conceptual illustration of GENIE, which consists of two sub-modules: synthesizing data and quantizing models

with only a small amount of unlabeled datasets or even in the absence of a dataset (including synthetic datasets). Because PTQ can compress models within a few hours but shows comparable performance to QAT, PTQ is preferred over QAT in practical situations.

Zero-shot quantization (ZSQ) [4, 7, 19] is another research regime that synthesizes data to compress models without employing real datasets. Starting from DFQ [22], schemes for ZSQ gradually pay more attention to generating elaborate replicas such that the distribution of intermediate feature maps matches the statistics of the corresponding batch normalization layers. Although many studies have achieved significant advancement in regards to quantization in the absence of real data, most of them have relied on QAT schemes that require task-specific loss, such as *cross-entropy (CE) loss* and *Kullback–Leibler (KL) divergence* [16], which requires more than 10 hours to complete the quantization of ResNet-18 [10] on Nvidia V100.

Excluding the data used, ZSQ and few-shot quantization[1](FSQ) commonly utilize FP32-pre-trained models (*teacher*) to optimize quantized models (*student*) by distilling knowledge. It is possible that ZSQ and FSQ share the quantization algorithm regardless of whether the data are real or synthetic. We thus adopt an up-to-date PTQ scheme to ZSQ so that breaking away from the quantization scheme conventionally used in ZSQ and then completing quantization within a few hours. Based on the existing method, we propose a framework called GENIE[2] that distill data suited for model quantization. We also suggest a novel quantization scheme, which is a sub-module of GENIE and available for both FSQ and ZSQ. As in Figure 2, GENIE consists of two sub-modules: synthesizing data (GENIE-D) and quantizing models (GENIE-M). By combining them, we bridge the gap between ZSQ and FSQ while taking an ultra-step forward from existing approaches. In other words, we achieve a state-of-the-art result that is unique among ZSQ approaches.

Our contributions are summarized as follows:

- First, we propose a scheme for synthesizing datasets by combining the approaches related to generation and distillation to take advantage of both approaches.
- Second, we suggest a method to substitute convolution of stride n ($n > 1$) by *swing convolution*. By applying randomness, various spatial information can be utilized when distilling datasets.
- Finally, we propose a new quantization scheme as a sub-module of GENIE (available for both FSQ and ZSQ), which is a simple but effective method that jointly optimizes quantization parameters.

2. Related Works

2.1. Uniform Quantization

Uniform quantization maps full-precision weights into fixed-point numbers. Supposing the step size $s \in \mathbb{R}$ is set by a certain algorithm, we can obtain the integers of weights as follows:

$$\boldsymbol{w}_{\text{int}} = clip\left(\left\lfloor \frac{\boldsymbol{w}}{s} \right\rceil + \boldsymbol{z}, n, p\right). \tag{1}$$

where $\lfloor \cdot \rceil$ denotes the nearest-rounding method, and n and p represent the lower and upper bound of the range, respectively. For example, when we asymmetrically quantize a layer to INTb, n and p are equal to 0 and 2^{b-1}, respectively. And the zero-point vector $\boldsymbol{z}$ represents an all-z vector, where $z = -\left\lfloor \frac{\min(\boldsymbol{w})}{s} \right\rceil$. Thus, the quantized weights $\boldsymbol{w}^q$ can be represented as follows:

$$\boldsymbol{w}^q = s(\boldsymbol{w}_{\text{int}} - \boldsymbol{z}). \tag{2}$$

To quantize a model, *Min-Max* algorithm sets the step size s as

$$s = \frac{\max(\boldsymbol{w}) - \min(\boldsymbol{w})}{2^b - 1}, \tag{3}$$

where b is the bit width for quantization. During optimization, *Min-Max* updates s in every step using an *exponential moving average* (EMA) of $\min(\boldsymbol{w})$ and $\max(\boldsymbol{w})$, and update $\boldsymbol{w}$ by *straight-through estimator* (STE) (*i.e.*, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^q}$) [2]. *Learned step size quantization* (LSQ) [8] learns the step size s along with $\boldsymbol{w}$ using STE, which is a state-of-the-art method in QAT. Both algorithms are mainly used for net-wise optimization or QAT but can be used in a divide-and-conquer approach or in PTQ.

PTQ [12, 14, 17, 21] optimizes networks mainly by the divide-and-conquer approach. Because they assume few shots are provided, they exploit knowledge from pre-trained models when quantizing models. AdaRound [21] have suggested a rounding scheme while minimizing the reconstruction error for each layer between the pre-trained and quantized models (*i.e.*, the mean squared error between activations of the two models). BRECQ [17] also has empirically shown the superiority of block-wise optimization. In contrast to ordinary knowledge distillation that aims to reduce

[1] This refers to post-training quantization with few real data

[2] Data generation scheme suited for quantization

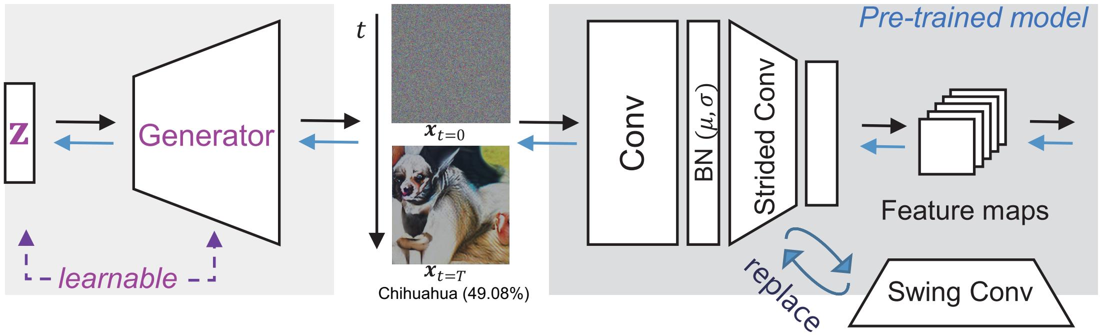


Figure 3. Data distillation (from noise ($t = 0$) to signal ($t = T$)). We train the generator and latent vectors $\boldsymbol{z}$, each of which is a kind of seed that generates its own image. The synthetic dataset is distilled indirectly by learning the latent vectors and the generator. When distilling images, $n(> 1)$-stride convolutions on pre-trained models are replaced by *swing convolutions*.

the scale (*e.g*., the number of layers), the layers between *teacher* and *student* are mapped one-to-one in quantization, and thus models can be optimized per layer or per block in a divide-and-conquer approach. Although the latest algorithms for PTQ have been verified on real data, they can also be employed for ZSQ.

2.2. Zero-shot Quantization

Under the assumption that data are not available for quantization of models, ZSQ focuses on generating or distilling data by exploiting the information from pre-trained models. To synthesize the data, it uses the parameters in the batch normalization layers and assigns virtual hard labels $\boldsymbol{y}$ to that synthetic data $\boldsymbol{x}$ in order to utilize the *CE loss*, which can be represented as follows:

$$\mathcal{L}_{\text{CE}} = \mathbb{E}\left[\text{CE}(f_p(\boldsymbol{x}), \boldsymbol{y})\right], \tag{4}$$

where f_p denotes a pre-trained model. In addition, Qimera [6] attempted to make boundary-supporting samples when synthesizing data for ZSQ. ZAQ [19] generates data that maximize the discrepancy (L_1-distance) between the activations of the two models (*i.e*., *teacher* and *student*) while using the loss function that minimizes the discrepancy for quantization. To quantize models, GDFQ [34] define the distillation loss as the combination of the *CE loss* and *KL divergence*. AIT [7] utilizes *KL*-only loss based on the observation that it has flatter minima than *CE loss*. Most of these quantization techniques optimize generators and quantized networks alternately while employing *Min-Max* algorithm to quantize models. Furthermore, DFQ [22] and ZeroQ [4] utilize the synthetic data primarily to set the step size of the activations without gradient-based optimization.

Table 1 summarizes quantization algorithms for both real- and synthetic datasets. According to our experiments, there is a limitation to the diversification of samples by synthesis. Thus, it is more suitable for ZSQ to use PTQ algorithms than QAT which can be overfitted with a small amount of the data (or monotonous data).

Table 1. Categorization of quantization algorithms

	Divide-and-Conquer	Netwise
Real Data	AdaRound, BRECQ, AdaQuant	LSQ, *Min-Max*
Synthetic Data	GENIE, MixMix	GDFQ, AIT, Qimera, ZAQ, IntraQ

3. GENIE

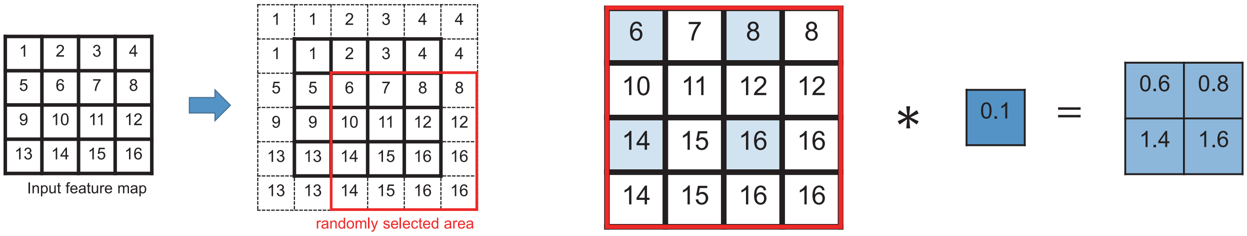
3.1. Data Distillation (GENIE-D)

To synthesize data, ZSQ commonly utilizes statistics (mean μ and standard deviation σ) in the batch normalization layers (BNS) of the pre-trained models as follows:

$$\mathcal{L}_{\text{BNS}}^{\text{D}} = \sum_{l=1}^{L} (\|\boldsymbol{\mu}_l^s - \boldsymbol{\mu}_l\|^2 + \|\boldsymbol{\sigma}_l^s - \boldsymbol{\sigma}_l\|^2) \tag{5}$$

where $\boldsymbol{\mu}_l^s/\boldsymbol{\sigma}_l^s$ and $\boldsymbol{\mu}_l/\boldsymbol{\sigma}_l$ represent the statistical parameters of the synthetic data and learned parameters in the l-th batch normalization layer, respectively. To minimize Eq. (5), generator-based schemes optimize weights in the generator, while distill-based schemes propagate the error directly to the synthetic data.

Generator-based approaches (GBA) [6, 7, 19, 34, 37] use latent vectors from a Gaussian distribution ($\mathcal{N}(0, \boldsymbol{I})$) as input of the generator in order to synthesize datasets. Thus, they have the advantage of synthesizing data infinitely as long as the input of the generator follows the designed distribution. Furthermore, the generator can be expected to learn *common knowledge* of the input domain. However, GBAs have been attempting to optimize the generator such that converting all noise to semantic signals, which not only takes a long time to converge but also converges at a relatively high loss (*i.e*., a low statistical similarity that is defined in Eq. (5)). Although it is also possible to generate infinite data, the information required for quantization is redundant, which limits the enhancement of quantized net-



(a) Reflection padding & random crop.

(b) 2-stride convolution (`conv2d(kernel_size=1, stride=2)`).

Figure 4. *Swing convolution*. (a) feature maps are extended by reflection padding and randomly cropped (b) The randomly selected areas in feature maps are convolved with the stride of n ($n > 1$)).

works, especially with QAT.

In contrast, distill-based approaches (DBA) [4, 18, 36] gradually update images from Gaussian random noises to semantic signals by distilling knowledge to the images. As it directly propagates the error to images, it converges relatively quickly to a low loss. However, there is no significant interaction between the instances except when measuring the loss in a batch.

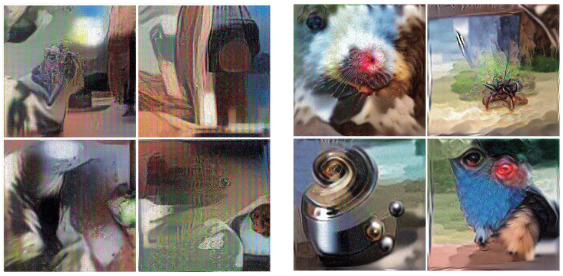
To take advantage of both approaches, inspired by *Generative Latent Optimization* (GLO) [3], we design a generator that produces synthetic data but distills the knowledge to latent vectors $\boldsymbol{z}$ from a normal distribution. In other words, the synthetic images are distilled indirectly by the latent vectors which are trained and updated in every iteration. Figure 3 illustrates the proposed method for distilling datasets. The latent vector initialized in the Gaussian form becomes an image via the generator, and the image takes the loss from the pre-trained model; the latent vector and generator are updated by the loss. The images from the initial vectors are close to noise, but they gradually mature into information suited for quantization as the optimization goes on. The image indicated by $\boldsymbol{x}_{t=T}$ in Figure 3 is a distilled image updated by the BNS loss (Eq. 5) and *swing convolution* (Figure 4) without any image prior loss. Genie-D synthesizes images that are very similar to the actual structure (Figure 1 and 3). By distilling latent vectors through the generator, GENIE-D converges as fast as DBA while learning the *common knowledge* of the input domain similar to GBA.

Indeed, we can consider GBA as Variational Auto-Encoder (VAE) [15] without the encoder. Suppose $\boldsymbol{x}$ and $\boldsymbol{x}'$ denote a real and fake sample (the output of the decoder or generator), respectively. The generator can be optimized by minimizing the distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$; it does not work well without the encoder that approximates true posterior $p_\theta(\boldsymbol{z}|\boldsymbol{x})$. Without variational inference, it is trying to match $\boldsymbol{x}'$ to $\boldsymbol{x}$ in pixel space (which has high dimensional). The distance loss can be replaced with BNS loss in GBA. Owing to the absence of real samples, GBA computes the distance indirectly by matching the distribution of the fake to its real (*i.e*., BNS loss) using pre-trained models. However, GLO explains that generative models can be trained without the encoder by optimizing the latent vector [3]. Furthermore, we can efficiently explore attributes of the real images pre-trained models utilized by optimizing in manifold space (latent vector) (n-dimension) rather than optimizing in pixel space (m-dim, $n \ll m$), while training the generator for *common knowledge* or image prior [1, 13, 30]. Which may explain the efficacy of optimizing latent vectors in addition to the generator compared to DBA and GBA.

3.1.1 Swing Convolution

When distilling images without a generator (*i.e*. DBA), we can consider the back-propagating error (with respect to $\mathcal{L}^{\mathrm{D}}_{\mathrm{BNS}}$) into the images as the process of image generation (*i.e*., model inversion [20]). Moreover, the backpropagation function (*backprop-op*) of convolutional layers (*conv*) is transposed convolution (*tconv*); the *backprop-op* for convolution of the stride n ($n > 1$, *s-conv*) is also n-stride *tconv* (*s-tconv*). Because *s-tconv* (which is commonly used to increase the resolution of images) can create checkerboard artifacts when generating images [24], the distilled images produced by *backprop-op* for *s-conv* (*i.e*. *s-tconv*) during model inversion also can result in checkerboard artifacts, which degrade the quality of images owing to information loss. Thus, we introduce *swing convolution* performing stochastic n-stride *conv*, which is simple but effective in reducing checkerboard artifacts caused by information loss and requires negligible extra computational cost.

With the reduction of information loss, distilled images become considerably robust and enhance the quality of quantized models. Figure 4 illustrates the mechanism of *swing conv*. Before 2-stride *conv*, the feature maps are extended by padding with their edge values (*i.e*., reflection padding) and randomly cropped to restore them to their original sizes, as shown in Figure 4a. Then, the randomly selected areas in the feature maps are stride-convolved as shown in Figure 4b. We refer to this series of processes as *swing conv*. We replace all *s-convs* to *swing convs* when only synthesizing the datasets. By applying randomness to the feature maps to be convolved, the distilled images can



(a) Distilling without *swing conv* (b) Distilling with *swing conv*

Figure 5. The effect of *swing convolution* that alleviates checkerboard artifacts resulting from information loss. The images in the same cell in each grid were distilled from the same seed. The images were directly distilled without the generator.

Algorithm 1 Data distillation (GENIE-D)

```
Input: Pre-trained model $f_p$
Output: Synthetic (distilled) data $\boldsymbol{x}^r$
1: procedure DATA DISTILLATION($f_p$)
2:     $\hat{f}_p$ = Strided_Conv_To_Swing ($f_p$)
3:     Init. latent vectors $\boldsymbol{z} \sim \mathcal{N}(0, \boldsymbol{I})$ and weights $\boldsymbol{W}_{\mathcal{G}}$ of generator $\mathcal{G}$
4:     repeat
5:         $\boldsymbol{x}^r = \mathcal{G}(\boldsymbol{z})$
6:         Compute $\hat{f}_p(\boldsymbol{x}^r)$
7:         Update $\boldsymbol{z}$ and $\boldsymbol{W}_{\mathcal{G}}$ with respect to $\mathcal{L}^{\mathrm{D}}_{\mathrm{BNS}}$    ▷ See Eq.(5)
8:     until converged
```

be updated so that the statistics of the outputs match the BNS regardless of the feature maps selected randomly.

Suppose that there is a 1×1 convolutional layer of stride 2 with 4×4 feature maps as input, pixels in the second and fourth row and columns are not utilized in the BNS loss and thus not used for back-propagation. With *swing convolution*, however, all pixels in the feature maps for 2-stride *conv* can contribute toward distilling images across the optimization, which provides various spatial information with distilled images and results in enhancing the quantized models without information loss. Note that shift operations such as *swing conv* have been used to enhance models in various variations [5, 32, 35]. Especially, we have found that the mechanism of random shifting convolution [35] is the same as *swing convolution* although the purpose and way of use are different. In model inversion, including distillation for ZSQ, to the best of our knowledge, this is the first instance of using stochastic convolution.

Algorithm 1 describes our proposed method GENIE-D used for synthesizing datasets. Before optimization, GENIE-D replaces all *sconvs* to *swing convs* in the pre-trained model (*line 2*). Note that the *sconvs* are substituted by *swing convs* only when distilling data and not during the quantization of models. After initializing the latent vectors $\boldsymbol{z}$ and weights of the generator (*line 3*), GENIE-D optimizes them with respect to $\mathcal{L}^{\mathrm{D}}_{\mathrm{BNS}}$ (*line 4–8*). Moreover, we applied various methods including existing works to explicitly generate diverse data by modifying or adding any loss (including *CE* loss) on top of GENIE, but there was no significant improvement at least in PTQ.

3.2. Quantization Algorithm (GENIE-M)

When quantizing a model of weights $\boldsymbol{W} \in \mathbb{R}^{m \times n}$ with a fixed-point representation (*e.g.*, INT8) in a post-training approach, we can set the step size (or scaling factor) s from the pre-trained model as follows:

$$s^* = \arg\min_s \left\| \boldsymbol{W} - s \cdot clip\left(\left\lfloor \frac{\boldsymbol{W}}{s} \right\rceil, n, p \right) \right\|_F, \tag{6}$$

where $\lfloor \cdot \rceil$ denotes the nearest-rounding method, and n and p represent the lower and upper bounds of the range, respectively. For example, when we symmetrically quantize a model to INTb, n and p are equal to 2^{b-1} and $2^{b-1}{-}1$, respectively. With the step size s, $\boldsymbol{W}_{\text{int}}$ and $\boldsymbol{W}^q$ can be defined as follows:

$$\boldsymbol{W}_{\text{int}} := clip\left(\left\lfloor \frac{\boldsymbol{W}}{s} \right\rceil, n, p \right) \tag{7}$$

$$\boldsymbol{W}^q := s \cdot \boldsymbol{W}_{\text{int}}. \tag{8}$$

Using the above formulation, we can quantize neural networks even in the absence of data. To further optimize the networks with an unlabeled dataset of small size, AdaRound [21] proposed a rounding scheme that allocated weights to one of the two nearest quantization points. Let the *base* integer matrix $\boldsymbol{B} \in [n, p]^{m \times n}$ be defined as

$$\boldsymbol{B} := clip\left(\left\lfloor \frac{\boldsymbol{W}}{s} \right\rfloor, n, p \right), \tag{9}$$

where $\lfloor \cdot \rfloor$ denotes the floor function. Then, AdaRound sets the quantized weights $\boldsymbol{W}^q$ as

$$\boldsymbol{W}^q = s \cdot (\boldsymbol{B} + \boldsymbol{V}) \tag{10}$$

where $\boldsymbol{V} \in [0, 1]^{m \times n}$ is the *softbit*[3] matrix to be optimized such that each weight $w_{\text{int},i}$ ($\in \boldsymbol{W}_{\text{int}}$) is converged to either b_i ($\in \boldsymbol{B}$) or $b_i + 1$. However, AdaRound does not jointly optimize the step size s with *softbit* $\boldsymbol{V}$, and the reason is explained as follows: "It is non-trivial to combine the two tasks: any change in the step size would result in a different quadratic unconstrained binary optimization (QUBO) problem" [21]. Because optimizing the step size s results in the change in the *base* $\boldsymbol{B}$ (Eq. (9)), it can cause a conflict with the *softbits* $\boldsymbol{V}$ being optimized. To resolve this issue, we suggest a method to enable joint optimization without

[3] We have omitted details for a concise description. Refer to the appendix for further details on *softbit*.

Table 2. Result of the ablation study on CNN Models (top-1 accuracy (%))

	#Bits	Ablation Settings				ResNet-18	ResNet-50	MobileNetV2	MobileNet-b	MnasNet-1.0
	(W/A)	Swing	Generator	z	Genie-M					
FP	32/32					71.08	77.00	72.49	74.53	73.52
M1	4/4					69.19	74.87	66.22	66.01	58.52
M2					✓	69.25	74.94	66.25	66.45	58.82
M3		✓				69.49	75.43	67.80	67.14	63.98
M4			✓			69.17	74.96	66.41	65.75	64.63
M5			✓	✓		69.58	75.39	67.92	67.40	66.15
M6		✓	✓	✓		69.62	75.47	68.28	68.02	66.55
M7		✓	✓	✓	✓	**69.66**	**75.59**	**68.38**	**68.58**	**66.94**
M1	2/4					61.96	66.72	36.58	23.18	31.22
M2					✓	62.62	66.95	37.12	27.44	32.45
M3		✓				63.74	69.44	44.00	27.85	34.64
M4			✓			60.13	65.28	34.92	27.51	35.50
M5			✓	✓		64.06	70.16	47.96	32.53	45.47
M6		✓	✓	✓		64.34	69.87	49.89	36.48	47.34
M7		✓	✓	✓	✓	**65.10**	**69.99**	**53.38**	**48.36**	**48.21**

Algorithm 2 CLASS GENIE-M

```
def: __INIT__(self, $\boldsymbol{W}$, $bits$)
    self.$s$ ← SetStepSize($\boldsymbol{W}$, $bits$)                                ▷ Eq. (6)
    self.$\boldsymbol{B}$ ← clip($\left\lfloor \frac{\boldsymbol{W}}{\text{self}.s} \right\rceil$, $n, p$).detach()          ▷ Eq. (9)
    self.$\boldsymbol{V}$ ← $\frac{\boldsymbol{W}}{self.s} - self.\boldsymbol{B}$

def: FORWARD(self)
    return self.$s$×(self.$\boldsymbol{B}$+self.$\boldsymbol{V}$)                      ▷ Eq. (10)
```

conflict via a sub-module of GENIE (GENIE-M), which is simple yet effective. Regardless of the s being optimized, we maintain $\boldsymbol{B}$ as initialized by releasing the mutual dependency between $\boldsymbol{B}$ and s (*i.e*., $\boldsymbol{B}$.`detach()`). In other words, we consider s as a learnable parameter that does not affect $\boldsymbol{B}$. In Eq. (10), $\boldsymbol{B}$ is considered constant and not dependent on s, and the losses propagated to s, v $(\in \boldsymbol{V})$, and b $(\in \boldsymbol{B})$ during optimization are computed as follows:

$$\frac{\partial w^q}{\partial s} = b + v, \quad \frac{\partial w^q}{\partial v} = s \quad \text{and} \quad \frac{\partial w^q}{\partial b} = 0. \tag{11}$$

Note that, such a joint optimization method can be applied to other post-training quantization algorithms such as AdaQuant [12] in addition to AdaRound [21], that optimize only the integers of weights while maintaining the step size as the initialized states.

Algorithm 2 presents the pseudo-code for GENIE-M class, which is the sub-module used for distilling models in GENIE. Using this quantizer, we optimize the quantized models by minimizing the block-wise reconstruction error, like that in BRECQ [17]. GENIE-M also uses LSQ [8] to optimize the step size of activations, which can be combined with QDROP [33]. All activations in a block are simultaneously optimized with all the weights in the block.

4. Experimental Results

We evaluate our proposed method by testing it on convolutional neural networks (CNNs), such as ResNet [10], MobileNet [11, 27], RegNet [25], and MnasNet [29], where only $1K$ synthetic images distilled by GENIE-D are used. To optimize the quantized model, GENIE-M quantizes each channel with asymmetrical ranges, whereas it quantizes the activations per tensor with symmetrical ranges. GENIE-M also exploits LSQ [8] with QDrop [33] for activation quantization.

4.1. Ablation Study

We build a combination of existing methods, such as ZeroQ+QDROP, and use it as the baseline of our approach (labeled **M1** in Table 2). Subsequently, we conduct an ablation study on various combinations to justify our design and verify the performance of GENIE. The results of the ablation study are presented in Table 2:

- **M1** vs. **M6** and **M4** vs. **M6** describe the performance of GENIE-D, the data distiller, compared to the existing methods such as ZeroQ (**M1**) and GBA (**M4**).
- **M4** vs. **M5** explains the efficacy when optimizing latent vector in addition to the generator.
- **M3** and **M5** describe the performance of each factor constitutive of GENIE-D. In **M5**, images are distilled indirectly by training the latent vector $\boldsymbol{z}$ without replacing the strided convolution with *swing*, while images in **M3** are distilled directly without the generator but with *swing*.
- **M1** vs. **M2** and **M6** vs. **M7** show the performance of GENIE-M, the model distiller, compared to that of an existing PTQ scheme (*i.e*., QDROP [33]).
- **M7** demonstrates the performance when harmonizing the proposed methods, GENIE-M and GENIE-D.

Table 3. Evaluation of CNN Models I (top-1 accuracy (%))

	Methods	#Bits (W/A)	ResNet-18	ResNet-50	MobileNetV2	MobileNet-b	MnasNet-1.0
	Full Prec.	32/32	71.08	77.00	72.49	74.53	73.52
Single Model	ZeroQ+BRECQ‡	4/4	69.32	73.73	49.83	55.93	52.04
	KW+BRECQ‡		69.08	74.05	59.81	61.94	55.48
	IntraQ†+BRECQ		68.77	68.16	63.78	-	-
	Qimera+BRECQ		67.86	72.90	58.33	-	-
	GENIE-D+BRECQ [ours]		69.70	74.89	64.68	68.61	55.42
	GENIE [ours]		**69.66**	**75.59**	**68.38**	**68.58**	**66.94**
Mix*	MixMix+BRECQ‡		69.46	74.58	64.01	65.38	57.87
	GENIE-D+BRECQ [ours]		69.71	74.89	64.97	62.70	51.25
	GENIE [ours]		**69.77**	**75.41**	**68.70**	**69.04**	**67.45**
Real	QDROP§		69.62	75.45	68.84	-	-
	GENIE-M [ours]		**69.81**	**75.61**	**69.23**	**69.80**	**68.29**
Single Model	ZeroQ+BRECQ	2/4	61.63	64.16‡	34.39	23.53	13.83
	KW+BRECQ‡		-	57.74	-	-	-
	IntraQ†+BRECQ		55.39	44.78	35.38	-	-
	Qimera+BRECQ		47.80	49.13	3.73	-	-
	GENIE-D+BRECQ [ours]		64.24	69.38	45.28	42.50	29.72
	GENIE [ours]		**65.10**	**69.99**	**53.38**	**48.36**	**48.21**
Mix*	MixMix+BRECQ‡		-	66.49	-	-	-
	GENIE-D+BRECQ [ours]		64.91	69.96	42.19	28.50	31.22
	GENIE [ours]		**65.44**	**70.62**	**53.36**	**49.89**	**49.65**
Real	QDROP§		65.25	70.65	54.22	-	-
	GENIE-M [ours]		**66.23**	**71.06**	**57.74**	**51.90**	**55.57**

‡, § The figures are taken from [18]‡ and [33]§. † It synthesizes $5K$ images while others synthesize only $1K$ images.
* The synthetic datasets are distilled from multiple models like ensemble learning [18].

4.2. Performance of GENIE

To verify the performance of GENIE, we compare various ZSQ algorithms, including ZeroQ [4], KW [9], GDFQ [34], Qimera [6], ARC [37], AIT [7], ZAQ [19], MixMix [18] and IntraQ [36]. Tables 3 and 4 summarize the comparison results. In Table 3, we present the comparison of GENIE with other methods that utilize BRECQ as the quantizer. Among them, MixMix distills images from various models (a total of twenty-one) like ensemble learning. To compare with MixMix, we also distilled images from five models of Table 3. Despite ensembling fewer models, Genie-M achieves superiority over MixMix. Using the same quantizer (BRECQ), we verified the performance of GENIE-D, the data synthesizer. Even with 256 images, GENIE (62.46%) also shows better performance compared to others with 1K images as shown in Figure 6: Qimera (47.99%) and ZeroQ (61.6%). Based on the observation, we can consider the fake data distilled by GENIE-D more informative. The influence on other models can be found in the appendix. All methods in Table 3 and Figure 6 quantized the first and the last layer into 8 bits like that in BRECQ.

For a fair comparison, we also quantize the first and last layers and follow the setting of quantization points of GDFQ, AIT, ZAQ, and IntraQ. The results are presented in Table 4, where the pre-trained models we use are the same as they utilized. As shown in the table, GENIE has significant differences from existing methods. Notably, when the bit width is W2A4, GENIE outperforms other methods, which empirically shows that algorithms for PTQ are more suitable for ZSQ than schemes for QAT. In Table 6, we

Table 4. Evaluation of CNN Models II (top-1 accuracy (%))

Methods		ResNet-18	ResNet-50	MobileNetV2
Full Prec.		71.47	77.73	73.03
GDFQ+AIT*	4/4	65.51	64.24	65.39
Qimera+AIT*		66.83	67.63	66.81
ARC+AIT*		65.73	68.27	66.47
ZAQ†		-	70.06	-
IntraQ‡		66.47	-	65.10
GENIE-D+AIT		66.91	-	-
GENIE [ours]		**68.69**	**74.21**	**69.59**
GDFQ+AIT	2/4	0.10	0.10	0.11
Qimera+AIT		0.10	0.10	0.12
ARC+AIT		0.11	0.10	0.13
IntraQ		0.14	-	0.17
GENIE-D+AIT		**0.50**	-	-
GENIE [ours]		**58.73**	**54.83**	**45.84**

,†,‡ The figures are taken from [7], [19]†, and [36]‡.

Table 5. Performance comparison using ***real samples*** ($1K$) (top-1 Accuracy (%))

Methods	#Bits (W/A)	ResNet-18	ResNet-50	MobileNetV2	RegNetX-600M	RegNetX-3.2G	MnasNet-2.0
Full Prec.	32/32	71.08	77.00	72.49	73.71	78.36	76.68
AdaRound+QDROP†	4/4	69.10	75.03	67.89	70.62	76.33	72.39
GENIE-M+No Drop [ours]		69.13	74.93	68.22	70.87	76.50	72.68
GENIE-M+QDROP [ours]		**69.35**	**75.21**	**68.65**	**71.13**	**76.75**	**73.37**
AdaRound+No Drop†	2/4	64.16	69.60	51.61	61.52	70.29	60.00
AdaRound+QDROP†		64.66	70.08	52.92	63.10	70.95	62.36
GENIE-M+No Drop [ours]		65.27	70.39	55.55	63.66	71.79	62.76
GENIE-M+QDROP [ours]		**65.77**	**70.51**	**56.38**	**64.55**	**72.35**	**64.10**
AdaRound+QDROP†	3/3	65.56	71.07	54.27	64.53	71.43	63.47
GENIE-M+No Drop [ours]		65.50	71.08	55.28	64.37	72.05	62.17
GENIE-M+QDROP [ours]		**66.16**	**71.61**	**57.54**	**65.68**	**72.72**	**64.80**
AdaRound+No Drop†	2/2	46.64	47.90	4.55	25.52	39.76	9.51
AdaRound+QDROP†		51.14	54.74	8.46	38.90	52.36	22.70
GENIE-M+No Drop [ours]		50.52	51.80	12.63	34.03	40.97	19.60
GENIE-M+QDROP [ours]		**53.71**	**56.71**	**17.10**	**42.00**	**55.31**	**28.56**

† The figures are taken from [33].

Table 6. Elapsed time to complete ZSQ (Hours)

	GDFQ +AIT	Qimera +AIT	ARC +AIT	GENIE
ResNet-18	10.11 (1.71)	10.19 (2.23)	19.04 (10.56)	2.73 (2.40)
ResNet-50	21.03 (2.71)	19.08 (5.10)	25.01 (10.96)	6.22 (5.07)
MobileNetV2	17.21 (2.15)	19.50 (3.61)	25.75 (10.26)	4.10 (3.33)

[*] The number in brackets denotes the elapsed time to train the generator.

also measure the elapsed time to complete ZSQ on Nvidia V100. Because GBA such as GDFQ, Qimera, and AIT utilize QAT scheme for quantization, they devote quantization more time rather than training the generator. In contrast, GENIE focuses more on training the generator with the latent vectors while reducing the time to quantize models by using PTQ. In summary, the PTQ approach with distilled data is very efficient for both time and accuracy in ZSQ.

4.3. Comparison using Real Data

We conduct experiments with randomly sampled datasets from ImageNet (ILSVRC12) [26] on various CNN models to verify our quantization algorithm, GENIE-M. Table 5 shows the results of the comparisons; all methods quantized the first and the last layer into 8 bits like that in QDROP [33], a state-of-the-art method used for PTQ. QDROP is a scheme for generalizing quantized models by randomly dropping the quantization operation for activations, such as Dropout [28] or DropConnect [31]. QDROP thus can be compatible with other quantization schemes such as LSQ. As shown in the table, GENIE-M outperforms the existing methods while verifying the effect of joint optimization. We use the average values in the table after evaluating the accuracy of 20 runs with randomly sampled images.

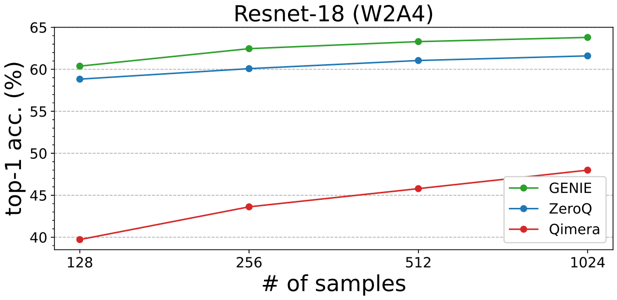


Figure 6. The influence of the number of samples on ResNet-18

5. Conclusion

We have proposed a framework called GENIE for ZSQ. By distilling the latent vectors through the generator, GENIE-D converges quickly to a low loss while learning *common knowledge* of the input domain from the pre-trained model. By replacing the n-stride convolutions (n>1) with *swing convolutions*, GENIE minimizes the information loss, resulting in enhanced performance of the quantized models. We have also suggested a PTQ scheme that jointly optimizes quantization parameters regardless of the data properties. In summary, GENIE has achieved a new state-of-the-art performance on both ZSQ and FSQ.

Acknowledgements

We would like to appreciate Kyungphil Park for his help in conducting the experiments and Seijoon Kim, Ph.D. for his helpful discussion.

References

[1] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *ACM Transactions on Graphics (TOG)*, 38(4):1–11, 2019. 4

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 2

[3] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. 4

[4] 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. 2, 3, 4, 7

[5] Weijie Chen, Di Xie, Yuan Zhang, and Shiliang Pu. All you need is a few shifts: Designing efficient convolutional neural networks for image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7241–7250, 2019. 5

[6] Kanghyun Choi, Deokki Hong, Noseong Park, Youngsok Kim, and Jinho Lee. Qimera: Data-free quantization with synthetic boundary supporting samples. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:14835–14847, 2021. 3, 7

[7] 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, 7

[8] 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, 2, 6

[9] Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8494–8502, 2020. 7

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 2, 6

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 6

[12] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 4466–4475. PMLR, 2021. 1, 2, 6

[13] Jinwoo Jeon, Kangwook Lee, Sewoong Oh, Jungseul Ok, et al. Gradient inversion with generative image prior. *Advances in neural information processing systems*, 34:29898–29908, 2021. 4

[14] Yongkweon Jeon, Chungman Lee, Eulrang Cho, and Yeonju Ro. Mr. BiQ: Post-training non-uniform quantization based on minimizing the reconstruction error. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12329–12338, 2022. 1, 2

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 4

[16] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951. 2

[17] 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, 2, 6

[18] Yuhang Li, Feng Zhu, Ruihao Gong, Mingzhu Shen, Xin Dong, Fengwei Yu, Shaoqing Lu, and Shi Gu. MixMix: All you need for data-free compression are feature and data mixing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 4410–4419, 2021. 4, 7

[19] Yuang Liu, Wei Zhang, and Jun Wang. Zero-shot adversarial quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (ECCV)*, pages 1512–1521, 2021. 2, 3, 7

[20] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Deepdream-a code example for visualizing neural networks. *Google Research*, 2(5), 2015. 4

[21] 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, 2, 5, 6

[22] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1325–1334, 2019. 2, 3

[23] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pages 16318–16330, 2022. 1

[24] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 4

[25] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10428–10436, 2020. 6

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy,

Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 8

[27] 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. 6

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 8

[29] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnas-Net: 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. 6

[30] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 4

[31] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR, 2013. 8

[32] Guangting Wang, Yucheng Zhao, Chuanxin Tang, Chong Luo, and Wenjun Zeng. When shift operation meets vision transformer: An extremely simple alternative to attention mechanism. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI)*, 2022. 5

[33] 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. 6, 7, 8

[34] 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. 3, 7

[35] Gangming Zhao, Jingdong Wang, Zhaoxiang Zhang, et al. Random shifting for CNN: a solution to reduce information loss in down-sampling layers. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3476–3482, 2017. 5

[36] Yunshan Zhong, Mingbao Lin, Gongrui Nan, Jianzhuang Liu, Baochang Zhang, Yonghong Tian, and Rongrong Ji. Intraq: Learning synthetic images with intra-class heterogeneity for zero-shot network quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12339–12348, 2022. 4, 7

[37] Baozhou Zhu, Peter Hofstee, Johan Peltenburg, Jinho Lee, and Zaid Alars. AutoReCon: Neural architecture search-based reconstruction for Data-free. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2021. 3, 7