Data Generation for Hardware-Friendly Post-Training Quantization

Lior Dikstein   Ariel Lapid   Arnon Netzer

Hai Victor Habi

Sony Semiconductor Israel

{lior.dikstein, ariel.lapid, arnon.netzer, hai.habi}@sony.com

Abstract

Zero-shot quantization (ZSQ) using synthetic data is a key approach for post-training quantization (PTQ) under privacy and security constraints. However, existing data generation methods often struggle to effectively generate data suitable for hardware-friendly quantization, where all model layers are quantized. We analyze existing data generation methods based on batch normalization (BN) matching and identify several gaps between synthetic and real data: 1) Current generation algorithms do not optimize the entire synthetic dataset simultaneously; 2) Data augmentations applied during training are often overlooked; and 3) A distribution shift occurs in the final model layers due to the absence of BN in those layers. These gaps negatively impact ZSQ performance, particularly in hardware-friendly quantization scenarios. In this work, we propose Data Generation for Hardware-Friendly Quantization (DGH), a novel method that addresses these gaps. DGH jointly optimizes all generated images, regardless of the image set size or GPU memory constraints. To address data augmentation mismatches, DGH includes a preprocessing stage that mimics the augmentation process and enhances image quality by incorporating natural image priors. Finally, we propose a new distribution-stretching loss that aligns the support of the feature map distribution between real and synthetic data. This loss is applied to the model's output and can be adapted to various tasks. DGH demonstrates significant improvements in quantization performance across multiple tasks, achieving up to a 30% increase in accuracy for hardware-friendly ZSQ in both classification and object detection, often performing on par with real data.

1. Introduction

Deep learning models (DNNs) are revolutionizing the field of computer vision, but deploying them on devices with limited memory or processing power remains a challenge [5, 17, 20, 27, 51]. Quantization, a powerful optimization technique, compresses the size of the model and reduces the computational demands, enabling efficient deployment [26, 42]. While quantization-aware training (QAT) incorporates quantization during the training process to help maintain the original accuracy [11, 23], post-training quantization (PTQ) [1, 9, 16, 30, 40] has gained significant traction for its ability to compress the model without requiring retraining. However, PTQ's effectiveness depends on the availability of representative data that aligns with the statistics of the original training set [21, 35, 53, 55, 58, 59, 61].

In many practical scenarios, access to such representative data samples is limited by privacy or security concerns. Regulatory restrictions or proprietary constraints may prevent the collection or use of even a few samples during model deployment. Zero-shot quantization (ZSQ) offers an alternative solution, enabling PTQ by utilizing synthetic data generation.

State-of-the-art data generation methods often rely on batch normalization (BN) [22] statistics matching to ensure the generated data aligns with the real data distribution [4, 18, 25, 53, 54, 56, 59, 60]. Despite considerable advancements, these methods often fail to generate data that is suitable for hardware-friendly quantization, where all model layers are quantized. To overcome this challenge, we analyze current methods and identify three key gaps in the generation process. The first gap lies in the inconsistency between how BN statistics are collected during training and how current data generation methods handle them. Typically, BN aggregates statistics across the entire dataset. However, existing data generation techniques optimize each sample [4] or batch [18, 25, 54, 56, 60] independently. This can result in synthetic datasets that fail to accurately represent the global statistical properties and diversity of the training data. The second gap stems from the data augmentation applied during training, which affects BN-collected statistics. These effects are often overlooked in synthetic data generation. The final gap occurs since existing methods focus primarily on minimizing a loss which is derived from BN statistics. However, there is no BN comparison point towards the later stages of deep neural networks, particularly at the output. This results in a distribution mis-

match between the feature maps of generated data and real data at these stages, a critical factor for hardware-friendly quantization schemes [15, 16, 31] that quantize the entire model, especially the output layer.

We address these gaps and introduce a data generation method for hardware-friendly post-training quantization (DGH) that enables ZSQ on real-world devices. To handle the BN statistic collection gap, DGH generates images while computing the BN statistics over the entire set of generated images. This is achieved by using a statistical aggregation approach that is not limited by available GPU memory. Moreover, optimizing BN statistics over the entire set relaxes the constraints on individual images, allowing them to deviate from the global mean and standard deviation, better approximating real data characteristics. Additionally, to account for the effects of data augmentation and to incorporate characteristics typical of natural images, we introduce an image-enhancement preprocessing step that includes smoothing operations and data augmentation. Finally, to address the stages unaffected by the BN loss, we propose a novel output distribution stretching loss mechanism that encourages generated images to utilize the model output's dynamic range. This loss can be applied to any task. This loss combined with the statistical aggregation scope are critical for generating images for ZSQ in deployment environments requiring full model quantization.

To summarize, our contributions are as follows:

- We identify and analyze three key gaps in existing data generation methods: the optimization scope, the impact of overlooking data augmentation, and the distribution mismatch in later network stages where BN layers are absent.
- We present DGH, a data generation method that enables hardware-friendly ZSQ. The method integrates three key components: statistics aggregation across the entire set of generated images, image enhancement preprocessing, and an output distribution stretching loss.
- We present experimental results on both ImageNet-1k classification (ResNet18, ResNet50 and MobileNet-V2) and COCO object detection (RetinaNet, SSDLite-MobilenetV3, and FCOS), demonstrating DGH's benefits for hardware-friendly ZSQ. For instance, DGH achieves up to 30% improvement over all state-of-the-art methods in hardware-friendly quantization. Additionally, we provide an ablation study that analyzes DGH and highlights the impact of addressing all identified gaps.

The appendices are provided in the online Supplementary Material. To promote reproducible research, the code developed in this work is openly available as part of Sony's open-source Model Compression Toolkit library at: https://github.com/sony/model_optimization.

2. Related Work

2.1. Quantization

Quantization methods are broadly divided into two categories [13, 32, 42]. The first category is QAT [2, 10–12, 20, 23, 43], which fine-tunes model weights by alternately performing quantization and backpropagation on abundant labeled training data. The performance achieved by QAT is often comparable to that of the respective floating-point models. However, this strategy is impractical in many real-world scenarios due to the expensive and time-consuming computation, algorithmic complexity, and the need for access to the entire training dataset.

The second category is PTQ [16, 21, 24, 30, 40, 44, 52], offering advantages in terms of data requirements and computational efficiency compared to QAT. Although PTQ methods do not rely on the availability of complete labeled training data, they require a small set of unlabeled domain samples for calibration.

2.2. Data Generation

Zero-shot quantization (ZSQ) methods aim to eliminate the need for representative data during quantization [8]. These methods can be roughly divided into two categories. The first category includes methods that operate without any data, such as DFQ [41], Squant [14], and OCS [57]. However, these methods are unsuitable for low-bit precision or activation quantization.

The second category leverages synthetic data generation for calibration. These methods exploit the inherent knowledge within a pre-trained model to generate synthetic images that closely resemble the true data distribution. One approach within this category is generator-based, where a generator is trained alongside the quantized model [6, 7, 28, 38, 53, 62]. The generator is trained to produce synthetic images with properties similar to the training data. However, generator-based methods demand extensive computational resources, as the generator requires training from scratch for each bit-width configuration.

Another approach treats data synthesis as an optimization problem. These methods are generally much faster than generator-based approaches. Methods such as ZeroQ [4], DeepInversion [54] and Harush *et al.* [18] begin with random noise images and iteratively update them by extracting the mean and standard deviation of activations from the BN layers, aligning them with the model's BN parameters. DSG [56] revealed that distilled images optimized to match BN statistics alone suffer from homogenization compared to real data. To address this, the authors suggested relaxing the BN statistics alignment constraint by introducing margin constants per layer and proposed two additional objectives to diversify the distilled data at the sample level. IntraQ [60] further demonstrated that enhancing

inter-class and intra-class heterogeneity in the distilled data used for calibration can improve the performance of a quantized model.

GENIE [25] leverages both approaches by using random inputs as latent vectors, passing them through a generator, and feeding the outputs into the model. The generator parameters, along with the random inputs, are updated iteratively. While this method delivers superior quantization results, it is considerably slower due to the need to train the generator from scratch for each batch.

3. Notation and Preliminaries

To generate a synthetic dataset, $\mathcal{D}$, for use in quantization, previous methods [4, 18, 25, 54, 60] typically utilize statistics, namely, the mean and standard deviation, extracted from the BN layers of pre-trained models. Specifically, let $f_l : \mathcal{I} \rightarrow \mathbb{R}^{c_l \times d_l}$ denotes the output of the lth layer, where $\mathcal{I}$ represents the input domain, c_l refers to the number of channels in layer l and d_l represents the feature dimensions in l^{th} layer. For example, in the case of image data, d_l represents the spatial dimensions $d_l = h_l w_l$. Next $g_{aug} : \mathcal{I} \rightarrow \mathcal{I}$ denotes the augmentation function, then the BN statistics of the lth layer are denoted as $\boldsymbol{\mu}_l \in \mathbb{R}^{c_l}$ and $\boldsymbol{\sigma}_l \in \mathbb{R}^{c_l}$ given by:

$$[\boldsymbol{\mu}_l]_i = \mathbb{E}_{\boldsymbol{Y}_l} \left[\frac{1}{d_l} \sum_j [\boldsymbol{Y}_l]_{i,j} \right], \quad (1a)$$

$$[\boldsymbol{\sigma}_l]_i = \mathbb{E}_{\boldsymbol{Y}_l} \left[\frac{1}{d_l} \sum_j \left([\boldsymbol{Y}_l]_{i,j} - [\boldsymbol{\mu}_l]_i \right)^2 \right], \quad (1b)$$

where $\boldsymbol{Y}_l = f_l \left(g_{aug} \left(\boldsymbol{X} \right) \right)$ is the output of the lth layer during training and $\boldsymbol{X}$ represents the distribution of the training set. Given a batch of K images $\boldsymbol{B} \triangleq \{\boldsymbol{x}_k\}_{k=1}^K$ to generate, the BN statistics (BNS) loss is defined as:

$$\mathcal{L}_{BNS}(\boldsymbol{B}) = \sum_{l=1}^L \|\bar{\boldsymbol{\mu}}_l(\boldsymbol{B}) - \boldsymbol{\mu}_l\|_2^2 + \|\bar{\boldsymbol{\sigma}}_l(\boldsymbol{B}) - \boldsymbol{\sigma}_l\|_2^2, \quad (2)$$

where L is the number of BN layers, $\|\boldsymbol{a}\|_2 \triangleq \sqrt{\sum_{i=1} [\boldsymbol{a}]_i^2}$ denotes the L2 norm of vector $\boldsymbol{a}$, $[\boldsymbol{a}]_i$ denotes the ith element of vector $\boldsymbol{a}$, $\bar{\boldsymbol{\mu}}_l(\boldsymbol{B})$ and $\bar{\boldsymbol{\sigma}}_l(\boldsymbol{B})$ denote the empirical mean and variance of the lth layer computed over the batch $\boldsymbol{B}$. The empirical mean and variance of the lth layer computed over the batch $\boldsymbol{B}$ are given by:

$$[\bar{\boldsymbol{\mu}}_l(\boldsymbol{B})]_i = \frac{1}{K d_l} \sum_{k,j=1}^{K,d_l} \left[\boldsymbol{y}_l^{(k)} \right]_{i,j}, \quad (3a)$$

$$[\bar{\boldsymbol{\sigma}}_l(\boldsymbol{B})]_i = \frac{1}{K d_l} \sum_{k,j=1}^{K,d_l} \left(\left[\boldsymbol{y}_l^{(k)} \right]_{i,j} - [\bar{\boldsymbol{\mu}}_l(\boldsymbol{B})]_i \right)^2, \quad (3b)$$

where $\boldsymbol{y}_l^{(k)} = f_l(\boldsymbol{x}_k)$ is the feature map of the lth layer and k^{th} sample. State-of-the-art methods, such as [4,18,25, 54, 60], divide $\mathcal{D} = \bigcup_{n=1}^N \boldsymbol{B}_n$ into N batches. The loss in Eq. (2) is optimized independently for each batch $\boldsymbol{B}_n \in \mathcal{D}$, where each batch contains K samples, resulting in a dataset of size $|\mathcal{D}| = K \cdot N$. For example, in ZeroQ [4], the statistics of each image are optimized to match those of the entire training set, *i.e.*, $K = 1$. In contrast, in [18,25,54,60], the statistics for each batch are optimized individually to match the training set, where $K \geq 1$ is a hyperparameter restricted by available GPU memory.

4. Analyzing Batch Normalization Based Data Generation

We identified three key gaps between how existing BN-based data generation methods generate data and how models collect statistics during training:

- **Statistics Aggregation Scope:** BN layers aggregate statistics over the entire dataset, while data generation techniques optimize individual batches or images independently to fit the BN statistics.
- **Data Augmentation:** Data augmentations applied during training, which affect BN statistics, are often neglected in data generation.
- **Output Distribution Mismatch:** The BNS loss has a limited impact on specific feature maps within the model, particularly the model outputs. This discrepancy results in a distribution mismatch between the synthesized and the real data in these feature maps.

4.1. Statistics Aggregation Scope

Notably, BN layers calculate mean and standard deviation across the entire dataset, *i.e.*, Eq. (1), rather than on individual images or batches [3, 22]. As a result, a BN layer's statistics represent the mean/std of activations averaged across all training images.

However, existing data generation techniques typically optimize each batch or image individually to match the statistics of the BN layers, *i.e.*, Eq. (3), ignoring the collective characteristics of the training data.

The impact of the generated images with varying statistics aggregation scopes on the accuracy of ZSQ for a ResNet18 model is illustrated in Fig. 1.

The curve clearly shows that quantization accuracy varies with the statistics aggregation scope, indicating that smaller scopes (batches) lead to generated images resulting in lower accuracy. In contrast, the red star highlights the accuracy achieved by our proposed aggregation algorithm,
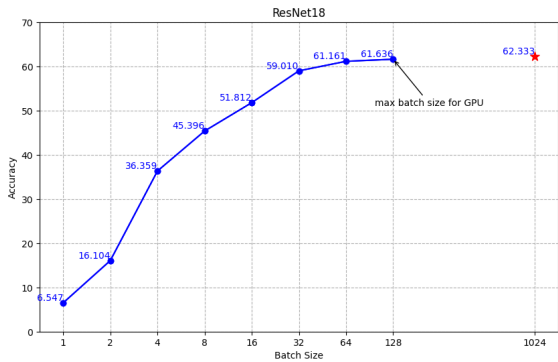
Figure 1. The blue curve represents the Top-1 accuracy on the ImageNet-1k validation set of ResNet-18 quantized to W3A8-bit precision using AdaRound [40] with 1024 generated images, where each batch is optimized separately according to Eq. (2). The x-axis denotes the statistics aggregation scope (batch size), K, used in the image generation process. The red star indicates the result of our proposed aggregation algorithm, which utilizes the statistics of all images collectively.

which optimizes the statistics across all images simultaneously, resulting in improved performance.

To further explore the impact of statistical calculation granularity on image generation, we visualized the embeddings of a ResNet-18 model using t-SNE [50]. Fig. 2 shows a two-dimensional t-SNE visualization: comparing real images versus those generated with global optimization and ZeroQ (sample optimization). Notably, embeddings from global optimization substantially overlap with real images, indicating a close match in the embedding space. In contrast, embeddings from ZeroQ are distinct from real images, showing significant separation. This highlights the benefits of our method's global statistic calculation over ZeroQ's per-image approach, suggesting our method more accurately captures the training data distribution.
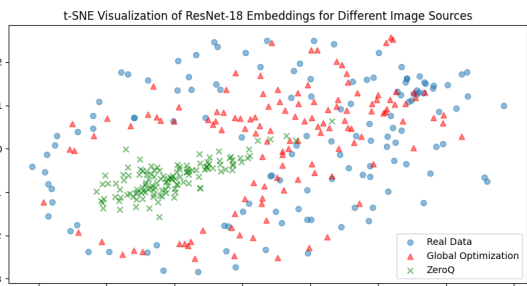

Figure 2. Two-dimensional t-SNE [50] visualization of ResNet-18 embeddings comparing real images (blue) with those generated with global optimization (red) and ZeroQ (green).
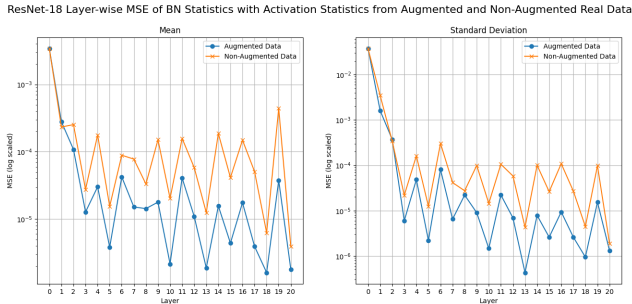

Figure 3. A layer-wise comparison of the MSE for mean (left) and standard deviation (right) of model activations relative to BN statistics in a ResNet-18 model. The blue line indicates MSE values calculated using augmented real data, while the orange line represents values from non-augmented real data. Each variant runs 1024 images to calculate the respective activation statistics, with MSE averaged over five experiments.

4.2. Data Augmentation

Training data typically undergoes augmentation before being fed to the model, with BN statistics collected on these augmented images [48]. However, existing data generation methods overlook this step. This is displayed in Eq. (1) and Eq. (3), where the statistics are calculated on $f_l(x)$ instead of $f_l(g_{aug}(x))$. Our approach directly addresses this by replicating the augmentation process during data generation.

To empirically validate the impact of our augmentation process on the alignment of activation statistics with BN statistics, we performed an experiment comparing MSE across various layers of a ResNet-18 model. This experiment involved running 1024 images, both with and without augmentation, and comparing the resulting activation statistics to the BN statistics of the model. The results, as illustrated in Fig. 3, show that the augmented data consistently exhibited lower MSE values compared to the non-augmented data. These findings support incorporating similar augmentation techniques into the data generation process.

4.3. Output Distribution Mismatch

While most data generation methods for ZSQ focus on minimizing the BNS loss, this approach often fails to match the statistics of feature maps unaffected by this loss. For example, BN layers are absent in the last stages of the network. This mismatch poses a significant challenge to quantization algorithms, as generated data may exhibit mismatched distributions in these feature maps. Consequently, quantization algorithms may choose suboptimal quantization parameters that do not align with the real data distribution, leading to degraded performance. This issue is particu-
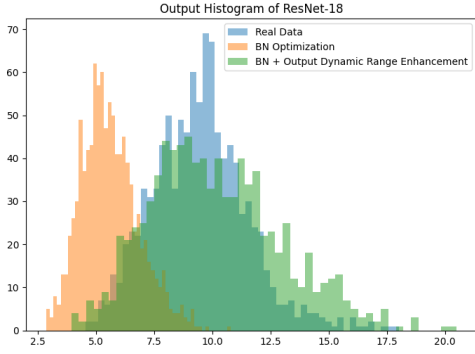
Figure 4. A comparison between the output distributions of a ResNet-18 model when inferring three different data sources: real images, synthetic images generated using only the BNS loss, and synthetic images enhanced with our proposed ODSL.

larly challenging in hardware-friendly quantization settings, where all layers must be quantized.

We illustrate this problem in Fig. 4, which compares the output distributions of ResNet-18 model for real images and generated images using only BNS loss. The figure shows that the logit values produced by the generated images can deviate significantly from those derived from real data. Previous methods [18,53,54,60] employed a cross-entropy loss function with a random class to encourage class-dependent image generation. Although this approach partially aligns the support of the distributions by pushing generated images towards strong classification, it has two main limitations: 1) it is confined to classification tasks; 2) it may push the logits to infinity without any constraints, affecting the loss landscape and hindering the convergence of the BNS loss. To address these limitations, we have introduced an output distribution stretching loss applicable to various tasks beyond classification. Our approach encourages generated images to mimic the output distribution support of real data. This strategy assists the quantization algorithm in selecting parameters that more accurately represent the data.

5. Method

In this section, we present our proposed method DGH. It is designed to address the challenges associated with data generation for hardware-friendly ZSQ, where all the model layers, including the output, are quantized. Our method leverages a combination of statistics aggregation to optimize the entire image set simultaneously, image pre-processing to integrate augmentations and image priors, and an output distribution stretching loss that improves the model's output dynamic range. DGH generates synthetic data that significantly improves performance across various hardware-friendly ZSQ tasks.

5.1. Statistics Aggregation Scope

We propose a novel approach to aggregate statistics throughout the entire set of generated images, which effectively implies that $N = 1$. This approach offers two significant advantages. First, it replicates the process of collecting BN statistics. This leads to better minimization of the BNS loss, as defined in Eq. (2). This is because the statistical estimations computed over a bigger batch become more accurate representations of the dataset statistics. Second, this approach relaxes the statistical constraints on individual images by considering a larger set of images during optimization. This relaxation allows each image to deviate from the mean and standard deviation of the BN statistics, similar to real data.

$$\mathcal{L}_{BNS}(\mathcal{D}) = \sum_{l=1}^{L} \|\bar{\boldsymbol{\mu}}_l(\mathcal{D}) - \boldsymbol{\mu}_l\|_2^2 + \|\bar{\boldsymbol{\sigma}}_l(\mathcal{D}) - \boldsymbol{\sigma}_l\|_2^2. \quad (4)$$

However, it's important to note that as the set $\mathcal{D}$ grows in size, computation resources may face limitations in terms of memory capacity, making it impractical to process the entire dataset simultaneously. To address this constraint, we introduce an aggregation algorithm, that effectively estimates the loss in Eq. (4). This algorithm allows us to optimize the entire set of images and ensure closer alignment of the generated data with the statistics of the pre-trained model's BN layers. We leverage the linearity of expectation and the relationship between variance and second moment to establish the following properties:

$$[\bar{\boldsymbol{\mu}}_l(\mathcal{D})]_i = \frac{1}{N} \sum_{n=1}^{N} [\bar{\boldsymbol{\mu}}_l(\boldsymbol{B}_n)]_i \quad (5a)$$

$$[\bar{\boldsymbol{\sigma}}_l(\mathcal{D})]_i = \frac{1}{N} \sum_{n=1}^{N} \left[\bar{\boldsymbol{M}}_l^{(2)}(\boldsymbol{B}_n)\right]_i - [\bar{\boldsymbol{\mu}}_l(\mathcal{D})]_i^2, \quad (5b)$$

where $\bar{\boldsymbol{\mu}}_l(\boldsymbol{B}_n)$, $\left[\bar{\boldsymbol{M}}_l^{(2)}(\boldsymbol{B}_n)\right]_i = \frac{1}{K d_l} \sum_{k,j=1}^{K,d_l} \left[\boldsymbol{y}_l^{(k)}\right]_{i,j}^2$ are the mean and second moment of the nth batch, respectively. The derivation of (5) is presented in Appendix A.2. The main idea of the algorithm is to store the statistics $\bar{\boldsymbol{\mu}}_l(\boldsymbol{B}_n)$ and $\bar{\boldsymbol{M}}_l^{(2)}(\boldsymbol{B}_n)$ for all batches. When a batch is optimized, its images are inferred and its layers' statistics are recalculated. Next, the stored statistics from all other batches are averaged together with the new recomputed statistics to recompute the global layers'. Subsequent backpropagation on the loss Eq. (4) is performed only on the specific batch. The detailed steps for computing global statistics are highlighted in red in Algorithm 1.

This algorithm enables calculating global statistics on any number of images, regardless of memory constraints. By aligning the statistics of the entire image set with

Algorithm 1 Image Generation using DGH. The global statistics computation is shown in red, the image prior computation in blue, and the output distribution stretching loss is shown in green.

Input: A pre-trained model $\mathcal{M}$ with L BN layers.
Output: A set of images: $\mathcal{D} = \{\boldsymbol{x}_m\}_{m=1}^M$.

1: Initialize $\mathcal{D}$ as standard Gaussian noise and split into N batches.
2: **for** each index n in N **do**
3: $\quad \tilde{\boldsymbol{B}}_n = \{\phi_{prep}(\boldsymbol{x}_k) : \boldsymbol{x}_k \in \boldsymbol{B}_n\}$
4: $\quad$ Compute and store
5: $\quad\quad \{\bar{\boldsymbol{\mu}}_l(\tilde{\boldsymbol{B}}_n)\}_{l=1}^L, \{\bar{\boldsymbol{M}}_l^{(2)}(\tilde{\boldsymbol{B}}_n)\}_{l=1}^L$.
6: **end for**
7: **for** $t \leftarrow 1$ to T iterations **do**
8: $\quad$ **for** each index n in N **do**
9: $\quad\quad \tilde{\boldsymbol{B}}_n = \{\phi_{prep}(\boldsymbol{x}_k) : \boldsymbol{x}_k \in \boldsymbol{B}_n\}$
10: $\quad\quad$ Run model $\mathcal{M}$ and compute batch statistics
11: $\quad\quad\quad \{\bar{\boldsymbol{\mu}}_l(\tilde{\boldsymbol{B}}_n)\}_{l=1}^L, \{\bar{\boldsymbol{M}}_l^{(2)}(\tilde{\boldsymbol{B}}_n)\}_{l=1}^L$.
12: $\quad\quad$ Re-compute the global statistics according to
13: $\quad\quad\quad$ Eq. (5).
14: $\quad\quad$ Compute $\mathcal{L}_{BNS}$ by Eq. (4).
15: $\quad\quad$ Compute $\mathcal{L}_{ODSL}$ by Eq. (7).
16: $\quad\quad$ Backpropagate (8) and update $\boldsymbol{B}_n$.
17: $\quad\quad \tilde{\boldsymbol{B}}_n = \{\phi_{prep}(\boldsymbol{x}_k) : \boldsymbol{x}_k \in \boldsymbol{B}_n\}$
18: $\quad\quad$ Run model $\mathcal{M}$ and compute batch statistics
19: $\quad\quad\quad \{\bar{\boldsymbol{\mu}}_l(\tilde{\boldsymbol{B}}_n)\}_{l=1}^L, \{\bar{\boldsymbol{M}}_l^{(2)}(\tilde{\boldsymbol{B}}_n)\}_{l=1}^L$.
20: $\quad$ **end for**
21: **end for**

---

the global BN statistics, the generated data more closely matches the statistical properties of the training data. This, in turn, allows the quantization algorithm to select parameters that better reflect the real data statistics.

5.2. Leveraging Data Augmentations and Image Priors

We introduce a pre-processing pipeline that leverages data augmentations and incorporates image priors to enhance the quality of generated images. We address a previously overlooked aspect regarding the BN statistics where the BN mean and standard deviation saved in the model were derived from an augmented training set. To handle this, we propose incorporating augmentations at the start of each optimization step. Our enhancement strategy contains several carefully crafted preprocessing steps. This step is incorporated using $\phi_{prep} : \mathcal{I} \to \mathcal{I}$ in Algorithm 1.

We further enhance the process by applying a smoothing filter to add natural properties to the images. This manages two core image generation challenges: 1) preserving the inherent smoothness of natural images without adding complex loss functions like total variation; 2) mitigating checkerboard artifacts, which are crucial for minimizing in-

formation loss. Prior studies, such as [25, 45], have highlighted the importance of these issues. This approach addresses artifact issues similarly to swing-convolutions [25]. However, since the smoothing operation is applied directly to the images, it is not confined to a specific layer, such as convolution. Instead, it mitigates artifact issues across all layers in the model, including those caused by operations like max pooling.

5.3. Output Distribution Stretching Loss

To address the output distribution discrepancies described in Section 4.3, we propose the output distribution stretching loss (ODSL). This loss enables the generation of data that cover the full support of the model's output and can be applied to any task. The primary objective of the output distribution stretching loss is to maximize the difference between the minimum and maximum values of the model's output for each image. In addition, to prevent significant deviations from the model's typical output values, we impose a constraint per image leveraging the last BN layer. This layer provides a point of information about the activations produced by the model in the training dataset. The objective is given by:

$$\max_{\boldsymbol{x}_k} \| \max_i [f_{\text{out}}(\boldsymbol{x}_k)]_i - \min_i [f_{\text{out}}(\boldsymbol{x}_k)]_i \|_2^2, \quad (6)$$
$$\text{s.t.} \quad \|\bar{\boldsymbol{\mu}}_L^k - \boldsymbol{\mu}_L\|_2^2 \le \delta, \quad \|\bar{\boldsymbol{\sigma}}_L^k - \boldsymbol{\sigma}_L\|_2^2 \le \delta,$$

where $\bar{\boldsymbol{\mu}}_L^k$ and $\bar{\boldsymbol{\sigma}}_L^k$ represent the mean and standard deviation of the activations at the last BN layer for the k-th image. Additionally, $f_{\text{out}}(\boldsymbol{x}_k)$ denotes the output activations of the model for the k-th image. Note that $f_{\text{out}}(\boldsymbol{x}_k)$ is a vector, and if the output is not a vector, we flatten it. The parameter δ defines the allowable deviation, ensuring that the output of each image remains within a controlled range of the typical output statistics of the model. This constraint resembles the alignment of the slack distribution introduced in [56]. The loss function defined as $\mathcal{L}_{\text{ODSL}}(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{k=1}^{|\mathcal{D}|} \ell(\boldsymbol{x}_k)$:

$$\ell(\boldsymbol{x}_k) = -\| \max_i [f_{\text{out}}(\boldsymbol{x}_k)]_i - \min_i [f_{\text{out}}(\boldsymbol{x}_k)]_i \|_2^2$$
$$+ \max\left(\|\bar{\boldsymbol{\mu}}_L^k - \boldsymbol{\mu}_L\|_2^2 - \delta, 0\right)$$
$$+ \max\left(\|\bar{\boldsymbol{\sigma}}_L^k - \boldsymbol{\sigma}_L\|_2^2 - \delta, 0\right). \quad (7)$$

Subsequently, the ODSL is incorporated into the total loss function (see Algorithm 1), with the hyperparameter λ:

$$\mathcal{L}(\mathcal{D}) = \mathcal{L}_{BNS}(\mathcal{D}) + \lambda \cdot \mathcal{L}_{\text{ODSL}}(\mathcal{D}). \quad (8)$$

6. Experimental Results

This section provides a comprehensive evaluation of DGH, emphasizing its ability to generate high-quality synthetic data that improves quantization performance across models and PTQ algorithms. We compare several ZSQ

Table 1. Top-1 accuracy on ImageNet-1k validation set with models fully quantized using BRECQ [30]. The models are quantized using four different data generation algorithms and real data.

Method	ResNet-18 71.06	ResNet-50 77.0	MBV2 72.49
	W4A4 Fully Quantized		
ZeroQ [4]	$0.44_{\pm0.04}$	$5.47_{\pm1.66}$	$1.31_{\pm0.29}$
IntraQ [60]	$\mathbf{47.44}_{\pm2.44}$	$25.6_{\pm5.58}$	$\mathbf{63.45}_{\pm0.26}$
GENIE [25]	$24.86_{\pm4.81}$	$\mathbf{49.84}_{\pm3.55}$	$25.15_{\pm0.98}$
DGH (Ours)	$\mathbf{65.64}_{\pm0.19}$	$\mathbf{72.35}_{\pm0.1}$	$\mathbf{65.0}_{\pm0.15}$
Real Data	$62.5_{\pm2.01}$	$71.65_{\pm0.15}$	$60.62_{\pm0.46}$
	W2A4 Fully Quantized		
ZeroQ [4]	$4.35_{\pm0.57}$	$21.98_{\pm1.36}$	$0.98_{\pm0.23}$
IntraQ [60]	$8.25_{\pm2.41}$	$7.41_{\pm0.97}$	$7.97_{\pm1.33}$
GENIE [25]	$\mathbf{23.55}_{\pm0.79}$	$\mathbf{40.89}_{\pm2.84}$	$\mathbf{23.5}_{\pm2.72}$
DGH (Ours)	$\mathbf{56.55}_{\pm0.64}$	$\mathbf{58.87}_{\pm0.29}$	$\mathbf{32.9}_{\pm1.14}$
Real Data	$56.7_{\pm1.71}$	$60.67_{\pm0.47}$	$42.02_{\pm1.52}$

Table 2. Top-1 accuracy on ImageNet-1k validation set with models fully quantized to W8A8 using using hardware-friendly PTQ [16]. The models are quantized using four different data generation algorithms and real data.

Method	ResNet-18 71.06	ResNet-50 77.0	MBV2 72.49
ZeroQ [4]	$2.21_{\pm0.00}$	$9.34_{\pm0.02}$	$1.79_{\pm0.01}$
IntraQ [60]	$\mathbf{70.60}_{\pm0.05}$	$\mathbf{75.95}_{\pm0.02}$	$\mathbf{72.05}_{\pm0.03}$
GENIE [25]	$64.21_{\pm0.10}$	$74.18_{\pm0.03}$	$65.70_{\pm0.04}$
DGH (Ours)	$\mathbf{70.91}_{\pm0.02}$	$\mathbf{76.54}_{\pm0.03}$	$\mathbf{72.47}_{\pm0.03}$
Real Data	$70.91_{\pm0.03}$	$76.58_{\pm0.01}$	$72.46_{\pm0.03}$

algorithms, including ZeroQ [4], IntraQ [60], and GENIE [25], using real data as an additional reference. Examples of synthetic images generated using DGH are presented in Appendix A.3.3

We begin by comparing the Top-1 accuracy on the ImageNet-1k [46] validation set for ResNets [19] and MobileNetV2 [47] using BRECQ [30] and HPTQ [16] with pre-trained models from [29]. The models are quantized in both hardware-friendly and academic quantization settings. Additionally, we evaluate the proposed method on the COCO validation set for object detection tasks, using models such as RetinaNet [33], SSD [37] and FCOS [49] from Torchvision [39]. Finally, we conduct an ablation study to show the contribution of each component of DGH. In all tables, the best results are shaded in gray and highlighted in bold (x), and the second-best results are highlighted in bold (**x**). The implementation details for our experiments can be found in Appendix A.1.

6.1. Classification Results

Here, we examine the effectiveness of image generation for hardware-friendly quantization, where all model layers are quantized, including the output. First, we employ the BRECQ algorithm [30] tailored for this setting. We quantize all weights and activations to WwAa, where w is the number of bits for the weights and a for the activations. This differs from the original BRECQ, which assigns 8 bits to specific layers and activations.

In Tab. 1, we present the Top-1 accuracy on the ImageNet-1k validation set using BRECQ algorithm [30] with activation bit-width set to 4 bits and weights bit-width is set to 2 and 4 bits. These results demonstrate that DGH significantly outperforms all other approaches and is comparable to, and occasionally surpasses, the results obtained

using real data.

Second, we employ the HPTQ method [16], which focuses on quantization of efficient[1] hardware platforms. The quantization bit-width for all layers is set to 8-bit for both weights and activations, ensuring compatibility with widely used hardware accelerators. In Tab. 2, we present the Top-1 accuracy on the ImageNet-1k validation set using the HPTQ quantization method. These results show a smaller performance gap due to the less challenging 8-bit configuration. We further analyze the effect of varying bit widths, validating this observation, with detailed results provided in Section A.3.2 of the supplementation materials. Additionally, IntraQ [60] demonstrates substantial improvement benefiting from its CE-like loss function. Overall, as seen in Tab. 1 and Tab. 2, these results highlight the advantages of DGH, achieving superior quantization performance on real-world deployment scenarios.

In addition to the hardware-friendly quantization scheme, we compare DGH to ZeroQ [4], IntraQ [60], and GENIE [25] on the academic quantization scheme where DGH has shown competitive results. The results are shown in Appendix A.3.1.

6.2. Object Detection Results

Here, we present the results of applying DGH to object detection models, showcasing its versatility across tasks beyond classification. We selected three widely used object detection architectures for our experiments: RetinaNet [33], SSD [37] and FCOS [49] from Torchvision [39]. The models were fully quantized using AdaRound [40], with 4-bit weights and 8-bit activations, and data generated using DGH, GENIE [25], and ZeroQ [4]. Since IntraQ [60] requires a CE loss, it was not suitable for this task. We evaluated the quantized models on the COCO validation dataset [34] and present the mean average precision (mAP). As shown in Tab. 3, DGH significantly outperforms all other methods and matches the accuracy achieved with real data.

---

[1]By efficient we mean the use of power-of-two thresholds.

Table 3. mAP on the COCO validation dataset using AdaRound with four different data generation algorithms and real data.

Method	SSDLite MBV3 Large 21.3	FCOS ResNet50 FPN V1 39.2	RetinaNet ResNet50 FPN V2 41.5
ZeroQ [4]	$14.58_{\pm 0.0}$	$37.54_{\pm 0.0}$	$2.60_{\pm 0.01}$
GENIE [25]	$13.71_{\pm 0.01}$	$28.08_{\pm 0.01}$	$12.89_{\pm 0.02}$
DGH (Ours)	$17.20_{\pm 0.01}$	$37.67_{\pm 0.0}$	$32.28_{\pm 0.01}$
Real Data	$17.16_{\pm 0.01}$	$37.98_{\pm 0.0}$	$32.36_{\pm 0.0}$

Table 4. Ablation study evaluating the impact of DGH's components on the ImageNet-1k validation set Top-1 accuracy using the BRECQ quantization algorithm applied to ResNet18 and MobileNet-V2. The study focuses on analyzing the effects of three key components: Statistics Aggregation Scope (SAS), Image Pre-processing (IP), and Output Distribution Stretching Loss (ODSL).

SAS	IP	ODSL	ResNet18 W4A4	ResNet18 W2A4	MBV2 W4A4	MBV2 W2A4
			15.92	15.4	18.53	20.79
✓			20.37	24.73	24.58	20.61
	✓		17.0	18.0	18.97	24.02
		✓	25.15	24.42	31.15	23.42
✓	✓		22.58	22.79	24.91	24.83
✓		✓	64.86	50.74	62.44	25.46
	✓	✓	19.54	23.51	26.47	25.43
✓	✓	✓	65.64	56.55	65.0	32.90

6.3. Ablation Study of DGH

In these experiments, we conduct an ablation study to validate the necessity of each component of DGH, as presented in Tab. 4. We systematically modify individual components to assess their impact on quantization accuracy, using the hardware-friendly quantization setting described in the previous subsection. The results demonstrate that while each component individually contributes to improvements over the baseline, the most significant gains arise when combining distribution alignment and ODSL.

To further investigate this effect, we conducted an additional experiment to assess how quantization performance benefits from ODSL and increasing statistical aggregation scope. We performed ZSQ using images generated with and without ODSL, varying the aggregation scope sizes, and evaluated the resulting accuracy. As shown in Fig. 5, the advantage of ODSL becomes evident with as few as four images, and the performance gap widens as the optimization scope expands. A larger statistical aggregation scope provides an additional degree of freedom for each individual image to deviate from the global statistics of the image set.
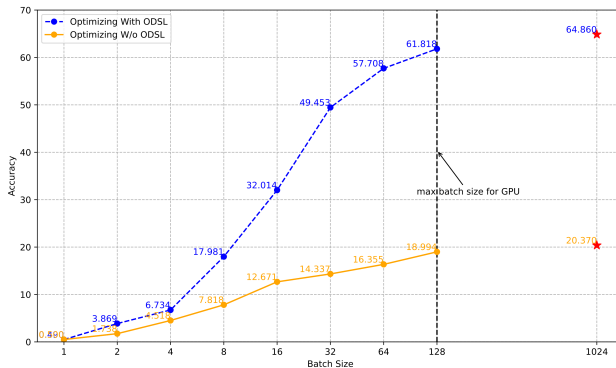

Figure 5. The figure represents the Top-1 accuracy on the ImageNet-1k validation set of ResNet-18 quantized to W4A4-bit precision using AdaRound with 1024 generated images. The blue curve represents optimizing with ODSL, while the orange curve represents optimizing without ODSL. The x-axis denotes the statistics aggregation scope (batch size) used in the image generation process, where each batch is optimized separately according to Eq. (2). The red stars indicate the results of using DGH's aggregation algorithm.

7. Conclusion

In this paper, we present DGH, a novel data generation method designed for hardware-friendly ZSQ. Our approach addresses three key limitations of existing methods: the statistics calculation scope of generated images, not accounting for data augmentations in the generation process, and feature maps missing BN comparison points. To resolve the statistics alignment issue, we introduce an aggregation algorithm that compares the statistics of the entire image set to the BN statistics during each iteration. For data augmentation, we add a preprocessing stage that applies image augmentations and a smoothing filter at the start of each iteration. Lastly, to address the missing optimization points, we introduce an output distribution stretching loss, applicable to any task. The experimental results demonstrate that DGH achieves state-of-the-art results for hardware-friendly ZSQ, where the models are fully quantized, enhancing the practicality of deploying quantized models on resource-constrained hardware. Additionally, the versatility of our method is validated through its successful application to various tasks, including image classification and object detection, and quantization algorithms such as BRECQ, AdaRound, and HPTQ, highlighting its robustness and efficiency in real-world scenarios. Future research will explore the extension of DGH to transformer models, investigating the potential benefits and optimizations that DGH can bring to the quantization of these architectures.

References

[1] Ron Banner, Yury Nahshan, and Daniel Soudry. Post training 4-bit quantization of convolutional networks for rapid-deployment. *Advances in Neural Information Processing Systems*, 32, 2019. 1

[2] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 696–697, 2020. 2

[3] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. *Advances in neural information processing systems*, 31, 2018. 3

[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*, pages 13169–13178, 2020. 1, 2, 3, 7, 8, 12, 13

[5] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017. 1

[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*, 34:14835–14847, 2021. 2

[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*, pages 8311–8321, 2022. 2

[8] Yoojin Choi, Jihwan Choi, Mostafa El-Khamy, and Jungwon Lee. Data-free network quantization with adversarial knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 710–711, 2020. 2

[9] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE, 2019. 1

[10] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015. 2

[11] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020. 1, 2

[12] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. Training with quantization noise for extreme model compression. *arXiv preprint arXiv:2004.07320*, 2020. 2

[13] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022. 2

[14] Cong Guo, Yuxian Qiu, Jingwen Leng, Xiaotian Gao, Chen Zhang, Yunxin Liu, Fan Yang, Yuhao Zhu, and Minyi Guo. SQuant: On-the-fly data-free quantization via diagonal hessian approximation. In *International Conference on Learning Representations (ICLR)*, 2022. 2

[15] Hai Victor Habi, Roy H Jennings, and Arnon Netzer. Hmq: Hardware friendly mixed precision quantization block for cnns. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVI 16*, pages 448–463. Springer, 2020. 2

[16] Hai Victor Habi, Reuven Peretz, Elad Cohen, Lior Dikstein, Oranit Dror, Idit Diamant, Roy H Jennings, and Arnon Netzer. Hptq: Hardware-friendly post training quantization. *arXiv preprint arXiv:2109.09113*, 2021. 1, 2, 7

[17] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1

[18] Matan Haroush, Itay Hubara, Elad Hoffer, and Daniel Soudry. The knowledge within: Methods for data-free model compression. *CoRR*, abs/1912.01274, 2019. 1, 2, 3, 5

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[20] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30, 2018. 1, 2

[21] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Accurate post training quantization with small calibration sets. In *International Conference on Machine Learning*, pages 4466–4475. PMLR, 2021. 1, 2

[22] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 1, 3

[23] 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*, pages 2704–2713, 2018. 1, 2

[24] 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*, pages 12329–12338, 2022. 2

[25] Yongkweon Jeon, Chungman Lee, and Ho-young Kim. Genie: Show me the data for quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12064–12073, 2023. 1, 3, 6, 7, 8, 12, 13

[26] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018. 1

[27] En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, 19(1):447–457, 2019. 1

[28] Jixing Li, Xiaozhou Guo, Benzhe Dai, Guoliang Gong, Min Jin, Gang Chen, Wenyu Mao, and Huaxiang Lu. Acq: Improving generative data-free quantization via attention correction. *Pattern Recognition*, 152:110444, 2024. 2

[29] Yuhang Li. Brecq: Towards accurate post-training quantization via bit-split and rectified clipping. https://github.com/yhhhli/BRECQ, 2024. Accessed: 2024-07-11. 7

[30] 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*, 2021. 1, 2, 7, 12

[31] Yuhang Li, Mingzhu Shen, Jian Ma, Yan Ren, Mingxin Zhao, Qi Zhang, Ruihao Gong, Fengwei Yu, and Junjie Yan. MQBench: Towards reproducible and deployable model quantization benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 2

[32] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021. 2

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 7

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 7

[35] Jiawei Liu, Lin Niu, Zhihang Yuan, Dawei Yang, Xinggang Wang, and Wenyu Liu. Pd-quant: Post-training quantization based on prediction difference metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24427–24437, 2023. 1

[36] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019. 12

[37] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016. 7

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

[39] TorchVision maintainers and contributors. TorchVision: PyTorch's Computer Vision library, Nov. 2016. 7

[40] 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*, pages 7197–7206. PMLR, 2020. 1, 2, 4, 7

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

[42] 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. 1, 2

[43] Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. Overcoming oscillations in quantization-aware training. In *International Conference on Machine Learning*, pages 16318–16330. PMLR, 2022. 2

[44] Yury Nahshan, Brian Chmiel, Chaim Baskin, Evgenii Zheltonozhskii, Ron Banner, Alex M Bronstein, and Avi Mendelson. Loss aware post-training quantization. *Machine Learning*, 110(11):3245–3262, 2021. 2

[45] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 6

[46] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 7

[47] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 7

[48] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019. 4

[49] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: A simple and strong anchor-free object detector. *IEEE transactions on pattern analysis and machine intelligence*, 44(4):1922–1933, 2020. 7

[50] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 4

[51] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8612–8620, 2019. 1

[52] Xiuying Wei, Ruihao Gong, Yuhang Li, Xianglong Liu, and Fengwei Yu. Qdrop: Randomly dropping quantization for extremely low-bit post-training quantization. *arXiv preprint arXiv:2203.05740*, 2022. 2

[53] Shoukai Xu, Haokun Li, Bohan Zhuang, Jing Liu, Jiezhang Cao, Chuangrun Liang, and Mingkui Tan. Generative low-bitwidth data free quantization. In *Computer Vision–ECCV*

2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16, pages 1–17. Springer, 2020. 1, 2, 5

[54] Hongxu Yin, Pavlo Molchanov, Jose M. Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deep-inversion. In *The IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2, 3, 5

[55] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 365–382, 2018. 1

[56] Xiangguo Zhang, Haotong Qin, Yifu Ding, Ruihao Gong, Qinghua Yan, Renshuai Tao, Yuhang Li, Fengwei Yu, and Xianglong Liu. Diversifying sample generation for accurate data-free quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15658–15667, June 2021. 1, 2, 6

[57] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *International conference on machine learning*, pages 7543–7552. PMLR, 2019. 2

[58] DanDan Zheng, Yuanliu Liu, Liang Li, et al. Leveraging inter-layer dependency for post-training quantization. *Advances in Neural Information Processing Systems*, 35:6666–6679, 2022. 1

[59] Yunshan Zhong, Mingbao Lin, Mengzhao Chen, Ke Li, Yunhang Shen, Fei Chao, Yongjian Wu, and Rongrong Ji. Fine-grained data distribution alignment for post-training quantization. In *European Conference on Computer Vision*, pages 70–86. Springer, 2022. 1

[60] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 12329–12338. IEEE, 2022. 1, 2, 3, 5, 7, 12, 13

[61] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017. 1

[62] Baozhou Zhu, Peter Hofstee, Johan Peltenburg, Jinho Lee, and Zaid Alars. Autorecon: Neural architecture search-based reconstruction for data-free compression. *arXiv preprint arXiv:2105.12151*, 2021. 2