Enhancing Diversity for Data-free Quantization

Supplementary Material

6. Comparison to existing works

Previous works [5, 6, 24, 46] utilize data augmentation. Here we compare the differences of our method with previous works. IntraQ [46] uses local reinforcement to conduct crop or resize to augment the generated images. MixMix [24] employs the most basic Mixup strategy, blending two generated samples as input and using the interpolated labels as target labels. Qimera [6] propose to Mixup the class embeddings, then generate images containing more classes and use the interpolated labels as target labels. As such randomly mixed unrelated images will result in inaccurate interpolated labels, which will confuse the full-precision model, TexQ [5] proposes to only Mixup generated images, without using interpolated labels.

Instead of using inaccurate label $y = \sum \lambda_i y_i$ with random weights λ_i which Mixup unrelated images as existing works do [5, 6, 24], we devise a multi-layer features mixer where our weights σ_i of the target label is the attention scores which Mixup label embeddings and learn $\mathcal{O}^0$ to capture relations among classes and enhance inter-class diversity. Besides, our class embeddings are extracted from multiple layers of the full-precision model, and we devise normalization flow based attention to focus on features of different levels and enhance intra-class diversity for generating data. We also observe the activation feature values in the activation layers collapse to a sharp peak and exhibit a long-tail distribution under the mode collapse problem, which is bad for quantizing the activation layers. Thus, we propose $SimLoss_1$ to help generate images exhibiting more diverse features to decentralize this distribution. Then, we use learnable parameters on the clip ranges in the quantized model, and propose $SimLoss_2$ to adaptively align the distribution of activation values between the quantized model and the full-precision model, for the quantized model adaptive to our generated diverse data.

On the other hand, HAST [23] proposed to generate images with larger losses, which are the hard samples that the full-precision model finds hard to classify correctly. This differs from our motivation, which emphasizes the diversity of images. Our method significantly outperforms HAST under the same benchmarks and settings. Genie [17] uses a quantization framework different from our method and our baselines, which will make the comparison unfair; thus, we did not compare Genie following our baselines. Specifically, all the methods in our paper use a fixed scale factor s, but Genie uses LSQ [10] to adjust s. Using LSQ will further improve the performance.

7. Experimental details

In this paper, all the quantization algorithms employ symmetric per-layer quantization which follows previous works [5, 6, 32, 33, 41, 47]. For a given weight parameter W_θ and quantization bit k, the quantization weight W_q is calculated as follows:

$$\begin{aligned} s &= \frac{u-l}{2^k-1}, \quad z = \frac{l}{s} + 2^{k-1}, \\ w_q &= clip(round(\frac{w_\theta}{s} - z), -2^{k-1}, 2^{k-1}-1), \end{aligned} \tag{14}$$

where u, l denote the maximum and minimum values of the weights respectively. Transformer architectures do not have BN layers and we did not use the batch normalization layers statistics (BNS) loss for ViT models. We followed the CNN-target quantization methods and used the BNS loss when quantizing CNN models.

Following previous works [5, 6], the max calibrating training epoch is 400 using an early stop strategy with the validation datasets, and the warm-up epoch for the generator is set to 50 for ImageNet and to 20 for CIFAR-100. We evenly divide the full-precision model into J layers without changing its original network structure as shown in Figure 3 and J is tuned from 2, 3 and 4. The training epoch of our classifier MLP^j is set to one-fifth of the warm-up epoch. After training our classifiers, we use the weights of all MLP^j to obtain the label embeddings. The attention heads are tuned from 2, 4 and 6, and the dimensions are tuned from 64, 128 and 256. The number of Transformer blocks in Eq.(8) is tuned from 2 and 4. All the experiments are conducted with PyTorch 1.2.0 and V100 GPU. For all baselines, we report the experimental results from their corresponding papers. For the rest quantization bit settings not covered in the original papers, we conduct experiments using their officially open codes and carefully tune the hyper-parameters based on the recommendations from the original papers.

- GDFQ https://github.com/xushoukai/GDFQ
- AIT https://github.com/iamkanghyunchoi/ait
- PSAQ-ViT https://github.com/zkkli/PSAQ-ViT
- ARC https://github.com/iamkanghyunchoi/ait/tree/main
- Qimera https://github.com/iamkanghyunchoi/qimera
- AdaSG https://github.com/hfutqian/AdaSG
- AdaDFQ https://github.com/hfutqian/AdaDFQ

Our code can be found at anonymous repo.

8. Time complexity analysis

Multi-layer Features: The main extra time complexity in our paper is the attention mechanism which has the com-

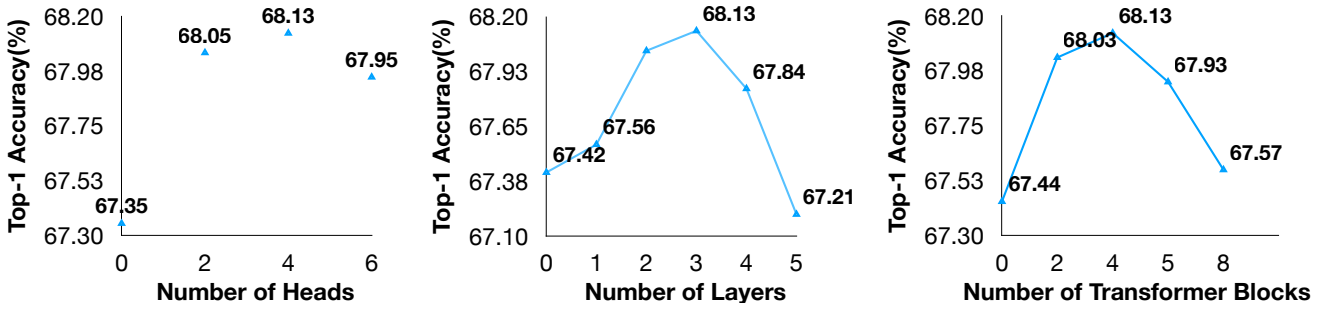

Figure 8. Parameter sensitivity.

plexity of $\mathcal{O}(c^2D)$ where c, D denote the number of Mixup classes and embedding dimension, respectively.

Loss function: For Sim_loss_1, the total complexity is $\mathcal{O}(nD)$ for $cosine$ operation where n and D denotes the number of generated samples in a training iteration and feature dimension in the full-precision, respectively. For Sim_loss_2, the complexity is $\mathcal{O}(JD)$ where J denotes the number of activation layers.

Backbone: We use the same backbone for the generator following all baselines, which consists of stacking Conv2D layers. The complexity of each Conv2D layer is $\mathcal{O}(C^2K^2HW)$, where C, K, H, W denotes the channel size, kernel size, the height and the width of the feature map, respectively.

Time complexity: The order of magnitude of complexity for the loss function is much smaller than the backbone, so it can safely be negligible. The time complexity of the attention mechanism is $\mathcal{O}(c^2D)$ where $c^2 \sim HW$ and $D \ll C^2K^2$ for the experiments on Cifar-100 and ImageNet. Therefore, our extra complexity $\mathcal{O}(c^2D)$ can safely be negligible as the complexity of generator $\mathcal{O}(C^2K^2HW)$ is the main time complexity. Thus, our overall complexity has the same order as the baselines.

9. Parameter sensitivity

We evaluate the hyperparameters, *i.e.* J the total number of layers of the full-precision model where we extract multi-layer features to obtain class embeddings, the number of Transformer blocks in Eq.(8), and the number of heads for attention in Eq.(6) and Transformer in Eq.(8). The hyperparameters setting to 0 denotes that we do not extract multi-layer features to obtain class embeddings or we do not use attention or Transformer blocks. The experimental results are shown in Figure 8 for ResNet-18 under the 4w4a setting. We can observe that: (1) If we increase the number of heads or the number of Transformer blocks up to 4, it will improve the performance as the generator could learn the relations among different classes and minutia features of different levels better. However, too many heads or Transformer blocks could result in more complex models. Since we lack the original training data, optimizing complex models becomes very challenging and thus we get worse performance. (2) Using more layers from the full-precision model up to 3 could help extract minutia features of more levels, thus enhancing the diversity of generated data and improving the final quantization performance. However, too many layers from the full-precision model will result in more complexity in Eq.(8) when focusing on minutia features of all J levels, and then we will get worse performance.

Table 5. Results on ResNet-50

	GDFQ		Qimera	
bit	original	+plugin	original	+plugin
3w3a	0.31	**17.23 (+16.92)**	1.82	**20.81 (+18.99)**
4w4a	52.12	**68.08 (+15.96)**	66.25	**68.22 (+1.97)**

10. Loss functions as plugin

Our loss functions help to generate diverse data with more complex feature patterns from the perspective of activated feature values and learn appropriate clip ranges adaptive to the generated diverse data in the activation layers. As our loss functions are model-agnostic, they can be integrated into existing data-free quantization methods. We use our loss functions as a plugin to existing methods to assess the effectiveness. We report the results in Table 5 for baselines GDFQ, which does not utilize augmentation, and Qimera, which utilizes latent representation to augment. We can see that our loss functions further improve the quantization performance by making the generator generate diverse data with more complex feature patterns from the perspective of activated feature values. The improvements are much more significant for the method where the mode collapse problem is more serious [6, 41].