Quantization without Tears

Minghao Fu Hao Yu Jie Shao Junjie Zhou Ke Zhu Jianxin Wu*
National Key Laboratory for Novel Software Technology, Nanjing University, China
School of Artificial Intelligence, Nanjing University, China
{fumh, yuh, shaoj, zhoujj, zhuk}@lamda.nju.edu.cn, wujx2001@gmail.com

Abstract

Deep neural networks, while achieving remarkable success across diverse tasks, demand significant resources, including computation, GPU memory, bandwidth, storage, and energy. Network quantization, as a standard compression and acceleration technique, reduces storage costs and enables potential inference acceleration by discretizing network weights and activations into a finite set of integer values. However, current quantization methods are often complex and sensitive, requiring extensive task-specific hyperparameters, where even a single misconfiguration can impair model performance, limiting generality across different models and tasks. In this paper, we propose Quantization without Tears (QwT), a method that simultaneously achieves quantization speed, accuracy, simplicity, and generality. The key insight of QwT is to incorporate a lightweight additional structure into the quantized network to mitigate information loss during quantization. This structure consists solely of a small set of linear layers, keeping the method simple and efficient. More importantly, it provides a closed-form solution, allowing us to improve accuracy effortlessly under 2 minutes. Extensive experiments across various vision, language, and multimodal tasks demonstrate that QwT is both highly effective and versatile. In fact, our approach offers a robust solution for network quantization that combines simplicity, accuracy, and adaptability, which provides new insights for the design of novel quantization paradigms.

1. Introduction

Along with their extraordinary breakthroughs in various vision [18], language [10] and multimodal [41] tasks, deep neural networks [11, 17] also exhibit ferocious greed for various resources: compute, GPU memory, bandwidth, storage, energy, *etc*. Hence, compressing and accelerating deep nets have not only attracted interests in academia, but are also an urgent need in real-world deployments and applications.

Among various research efforts in this direction, network quantization [28] is arguably the most practical one. Different from unstructured pruning [16], it is well supported by existing hardware. Compared to structured pruning [19, 22], its compression ratio is higher and its loss is relatively smaller. For example, the INT8 quantization of both FP32 weights and activations leads to roughly 4× reduction in network size and 4× speedup with almost zero accuracy loss in many applications [26], which far exceeds structured pruning. Existing methods are often categorized as Post-Training Quantization (PTQ) [27, 31, 32, 37, 46, 52] or Quantization-Aware Training (QAT) [13, 21, 25, 33, 54], where the difference is whether training is required ('no' for PTQ and 'yes' for QAT).

Quantization, however, is not as perfect as it seems to be. There are also obvious drawbacks and pitfalls in existing quantization methods.

- **The speed-accuracy dilemma**: PTQ can be thousands of times faster than QAT during the quantization process, but QAT may well be 10 percentage points higher than PTQ in accuracy during inference.
- **Complexity**: Quantization methods are often delicate and tricky. They often have tons of hyperparameters to tune for each specific task, and even one improperly set hyperparameter value may ruin the quantized model.
- **Missing generality**: Relevant to their complexities, an existing method is often geared toward a specific model and/or task. Different models/tasks require different quantization methods.

Given the status quo, at this moment it does not seem unreasonable to treat the act of network quantization as an art rather than an established engineering tool.

In this paper, we propose a Quantization without Tears (QwT) method to address these drawbacks, which achieves quantization speed, accuracy, simplicity and generality *simultaneously*.

The key to achieve these goals simultaneously is to slightly change the quantization paradigm. Suppose a net-

*J. Wu is the corresponding author.

work M has the network structure S and parameters θ. Current quantization methods will quantize it into a model $M^{\mathbb{Z}}$ with the same structure S ($S^{\mathbb{Z}} = S$) and quantized parameters $\theta^{\mathbb{Z}}$ in the integer format.

Our key argument is that the quantized structure does *not* need to be strictly S. In our QwT, it becomes $S \cup S_c$, where some extra modules S_c are added to the network structure to *compensate for the information loss due to quantized parameters and activations*. The extra S_c thus help us achieve high accuracy.

S_c does lead to extra overheads. But, it is also obvious that so long as the size and computation of S_c is small or even negligible when compared to $S^{\mathbb{Z}}$, we achieve *both speed and accuracy*. In our QwT, S_c has very simple structures: only few linear layers, which renders it *both simple and general*.

To be more specific, the parameters in S_c can be set in *closed-form with a small calibration set*, which in almost all cases leads to significantly higher accuracy than PTQ methods.

To sum up, the contributions of this paper are:

- Proposing a new paradigm for network quantization by lifting the restriction that the quantized network structure $S^{\mathbb{Z}}$ has to be exactly the same as that of the original network structure S.
- Proposing QwT, a simple and general quantization method without tears in this new paradigm. QwT achieves speed, accuracy, simplicity and generality simultaneously.

Extensive experiments have been carried out, which show that QwT has the following properties:

- **Fast and accurate**. For example, QwT quantizes a ViT network in roughly 2 minutes. During inference, its throughput is almost the same as models quantized by existing quantization methods. QwT is *significantly more accurate* than existing PTQ methods *even without any back-propagation*. On top of that, if higher accuracy is requested, QwT requires *only 1 epoch of training* to approach the accuracy of QAT methods. In contrast, QAT often requires a large number of epochs (*e.g.*, 200 epochs).
- **Simple**. There is *zero (0) hyperparameters* to tune, and the parameters in S_c can be found in *closed-form*.
- **General**. *Exactly the same* QwT method has been successfully applied to various networks and applications, including both CNNs [17] and Transformers (ViT [11] and Swin [34]), object recognition, detection (with both Mask R-CNN [18] and DETR [4]) and segmentation, multimodal models (CLIP [41]), generative models (DiT [40]), and large language models (LLaMA [12]).
- **Practical**. In addition to quantizing to low-bits in simulators, the same QwT method can quantize a model that is able to run directly on GPUs with minimum efforts (*i.e.*, quantization 'without tears'): Simply obtain $\theta^{\mathbb{Z}}$ using TensorRT, then add S_c using QwT. The resulting quantized model is then ready to be deployed on GPUs that support quantized fix-point inference.

2. Related Work

Network quantization [28] aims to reduce the bit-width of weights and activations, enabling the quantized model to be stored more efficiently and to perform faster inference with suitable hardware support. The fundamental principle of network quantization involves approximating full-precision weights and activations by mapping them to a finite set of discrete values, which are subsequently used in forward model computations (*i.e.*, in inference).

One line of research focuses on quantization-aware training (QAT) [13, 21, 25, 33], which integrates quantization into the training process using back-propagation, where the straight-through estimator [1] is commonly employed to approximate gradients for non-differentiable rounding functions.

Another line of research concentrates on post-training quantization (PTQ) [14, 27, 31, 52], which converts a fully trained full-precision model into low-bit format using a small set of calibration samples. AdaRound [38] proposed an adaptive weight-rounding mechanism. BRECQ [24] leveraged block reconstruction for quantization, utilizing the Fisher Information Matrix to guide the process. QDrop [50] randomly dropped the quantization of activations during quantization to achieve flatness of the low-bit model.

While these methods [24, 38, 50] have proven effective on ResNet [17] backbones, applying them directly to ViT [11] often degrades recognition accuracy since the intrinsic structure of the softmax attention is incompatible with these methods. This poses new challenges to design general PTQ methods for the Transformer architecture.

To address this issue, [35] introduced a ranking loss designed to preserve the relative order between quantized and non-quantized attention scores. PTQ4ViT [52] proposed twin uniform quantization for shifted activations and a Hessian-guided metric to generate scaling factors. RepQ-ViT [27] decoupled quantization and inference, employing distinct quantizers to enable precise quantization while simultaneously ensuring efficient inference. IGQ-ViT [37] introduced instance-aware group quantization for ViT to dynamically allocate channels of activation maps to different quantization groups. GPTQ [14] introduced a one-shot weight quantization technique that exploits approximate second-order information.

Different from all of these methods, we propose a new quantization paradigm that introduces a lightweight module to compensate for the information loss caused by quantization. This new paradigm allows our method to be

seamlessly integrated with any state-of-the-art quantization methods as a plugin in a completely black-box fashion. Experiments demonstrate that our method is highly compatible with various PTQ approaches, enabling effortless improvements in recognition accuracy within just 2 minutes.

3. Method

3.1. Preliminaries

We start by outlining key concepts and notations related to network quantization. Given a quantization bit-width b, the quantization function $Q(\cdot|b) : \mathbb{R} \to \mathbb{Z}$ maps a floating-point number x (*e.g.*, weight or activation) into its corresponding fixed-point representation $x^{\mathbb{Z}}$ encoded by b bits. Among various quantization approaches, uniform quantization is particularly favored thanks to its simplicity and compatibility with hardware deployment. The uniform quantization procedure is formalized as:

$$x^{\mathbb{Z}} = \text{clip}\left(\lfloor\frac{x}{s}\rceil + z, 0, 2^b - 1\right), \tag{1}$$

in which $s \in \mathbb{R}^+$ represents the quantization scale, and $z \in \mathbb{Z}$ denotes the zero-point offset. These parameters are determined as follows:

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

$$z = \text{clip}\left(\lfloor -\frac{\min(x)}{s}\rceil, 0, 2^b - 1\right). \tag{3}$$

In these equations, $\lfloor . \rceil$ denotes the rounding function, and the clip$(\cdot, a, b)$ operation constrains the input value into the range $[a, b]$. The reconstructed quantized output can then be formulated as:

$$\hat{x} = s \times (x^{\mathbb{Z}} - z). \tag{4}$$

Beyond the naive uniform quantizer, a range of more sophisticated quantization techniques [27, 38, 51, 52] have been proposed and extensively studied by the community. In the literature, quantization methods typically quantize both the model weights and activations.

Quantization significantly reduces the storage requirements by enabling models to be stored in lower bit formats. Additionally, thanks to hardware support for integer-only computations, operations involving the quantized representations $x^{\mathbb{Z}}$, such as matrix multiplications between quantized weights and activations, can be performed with substantial improvements in computational efficiency.

3.2. Compensation: The Key Insight

But, obviously there is significant information loss between x and $x^{\mathbb{Z}}$, and it grows very fast when many layers of computation and quantization are stacked together. To recover from the resulting accuracy loss, QAT methods resort to many epochs of training, which leads to the speed-accuracy dilemma and complex, ad hoc quantization methods.

Given a model $M = (S, \theta)$, where S and θ denote its structure and weights, respectively, existing quantization techniques transform M into a quantized version $M^{\mathbb{Z}} = (S^{\mathbb{Z}}, \theta^{\mathbb{Z}})$, which maintains the same network structure (*i.e.*, $S = S^{\mathbb{Z}}$) while modifying the original parameters θ to their quantized counterparts $\theta^{\mathbb{Z}}$.

Our key insight is to challenge this structural rigidity: the quantized model *does not necessarily need to retain the exactly same structural configuration*, *i.e.*, it is legitimate to allow $S^{\mathbb{Z}} \neq S$. We argue that some extra modules S_c can be added to the quantized model, such that its structure $S^{\mathbb{Z}} = S \cup S_c$. The extra modules in S_c can *compensate for the information loss caused by quantization*.

More specifically, modern deep nets typically compose of many blocks, *e.g.*, bottleneck blocks in ResNet [17] or Transformer blocks in ViT [11]. Let l_i denote the i-th block in a model, and let $x_i \in \mathbb{R}^{d_{in}}$ and $y_i \in \mathbb{R}^{d_{out}}$ be the input and output of this block l_i, respectively, such that $y_i = l_i(x_i)$. We argue that we can add a compensation module c_i for this block. Then, $S_c = \bigcup_i c_i$.

For notational simplicity, we omit the subscript i from now on, *i.e.*, we represent a block as $y = l(x)$ and the compensation module is simply denoted as c. After quantization, the input activations and weights of the block l are modified into the quantized version $x^{\mathbb{Z}}$ and $l^{\mathbb{Z}}$, respectively. The quantized computation becomes $y^{\mathbb{Z}} = l^{\mathbb{Z}}(x^{\mathbb{Z}})$.

Clearly, there is information loss in all 3 quantization pairs: $l \mapsto l^{\mathbb{Z}}$, $x \mapsto x^{\mathbb{Z}}$ and $y \mapsto y^{\mathbb{Z}}$. What is intriguing is that the information loss is obviously highly non-linear in all 3 pairs. To implement the compensation idea, we have to answer the following questions:

1. How to measure the information losses in all 3 pairs that interplay with each other in a complex manner?
2. How to design the compensation module c that accounts for these highly non-linear information losses?

3.3. QwT: Quantization without Tears

We propose a QwT (quantization without tears) method, which answers both questions in the simplest possible form.

First, the information loss is measured by $\|y - y^{\mathbb{Z}}\|^2$. Because y is the output of l, it naturally takes care of information losses in $x^{\mathbb{Z}}$ and $l^{\mathbb{Z}}$—when $y^{\mathbb{Z}} = y$, intuitively there is absolutely zero information loss even if $x \neq x^{\mathbb{Z}}$ and $l \neq l^{\mathbb{Z}}$. Note that $\|y - y^{\mathbb{Z}}\|^2$ also accounts for *cumulative* information losses. That is, in the i-th block, c_i compensates information losses accumulated in all previous blocks that have not yet been corrected by $c_1, c_2, \ldots, c_{i-1}$.

Second, because of this cumulative nature of our choice, in QwT we choose to implement c (with the index i omitted) as *a simple linear layer*. Although it is impossible to accurately compensate the non-linear information loss in

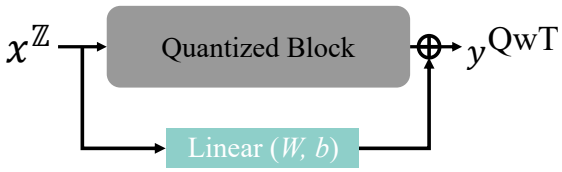

Figure 1. Illustration of QwT in one block. QwT adds a simple linear layer to any model block, compensating for information loss using the block input $x^{\mathbb{Z}}$. This approach is straightforward and compatible with almost all types of backbones [11, 17, 34].

one block via a linear layer, we have many chances to repeatedly apply linear corrections. The entire compensation formed by all extra modules is in fact non-linear because it interacts with the quantized network in every block.

To be concrete, we define $c(x) = Wx + b$ and then

$$y^{\text{QwT}} = l^{\mathbb{Z}}(x^{\mathbb{Z}}) + c(x^{\mathbb{Z}}), \tag{5}$$

where $W \in \mathbb{R}^{d_{out} \times d_{in}}$ and $b \in \mathbb{R}^{d_{out}}$ are the weight matrix and bias vector of the linear layer c, respectively. The QwT structure is illustrated in Figure 1.

These choices are deliberate. They not only make QwT conceptually the simplest, but also ensures a *closed-form* solution. To minimize the difference between y and $y^{\mathbb{Z}}$, we select a small set of training examples (512 images) from the training set. Using these samples, QwT collects all inputs for the block $l^{\mathbb{Z}}$ to form a matrix $X^{\mathbb{Z}} \in \mathbb{R}^{d_{in} \times N}$, where N is the total number of features or tokens. We feed $X^{\mathbb{Z}}$ into the quantized block $l^{\mathbb{Z}}$ to obtain $Y^{\mathbb{Z}} \in \mathbb{R}^{d_{out} \times N}$. Next, we feed $X^{\mathbb{Z}}$ into the non-quantized block l to obtain $Y \in \mathbb{R}^{d_{out} \times N}$. Correspondingly, Y contains output of the matching block in the original model.

Our task is then to *estimate* $Y - Y^{\mathbb{Z}}$ *using* $X^{\mathbb{Z}}$. This is a classic linear regression problem, and has a closed-form solution:

$$W^* = (Y - Y^{\mathbb{Z}}){X^{\mathbb{Z}}}^{\top}(X^{\mathbb{Z}}{X^{\mathbb{Z}}}^{\top})^{-1}. \tag{6}$$

In Equation 6, to simplify the solution, the bias b is absorbed into W and a row vector of $\mathbf{1}^{\top}$ are concatenated to $X^{\mathbb{Z}}$. It is worth mentioning that theoretically QwT will *not* make the quantized network getting worse—by setting W and b to all zeros, QwT will not alter the quantized network.

Note that after the QwT module for $l^{\mathbb{Z}}$ is inserted, the compensation in the next block depends on all the previous QwT modules. Consequently, the information loss from y to $y^{\mathbb{Z}}$ is gradually compensated block by block, allowing c to account for the accumulated loss from all preceding blocks that remain uncorrected.

In our experiments, we observed that the coefficient of determination (R^2) [43] for a small subset (<5%) of QwT modules was notably low, adversely hurting recognition accuracy. Consequently, we apply the initialization using Equation 6 only when $R^2 > 0$; Otherwise, the W and b of the QwT module are set to zero.

Finally, the QwT method has a simple pipeline: first quantize a model M using any PTQ method, then add the compensation module c_i to every block i and set the parameters in c_i using Equation 6.

A notable advantage of QwT lies in its inherent simplicity. This simplicity ensures that the initialization process of QwT modules is highly efficient, which requires *roughly 2 minutes* in practice to compensate for the information loss during quantization, thereby enhancing recognition accuracy. Experimental results demonstrated that QwT exhibits significant versatility and efficiency across various vision and language tasks.

4. Experiments

In this section, we begin by evaluating our QwT method on a range of discriminative tasks, including image classification, object detection, instance segmentation, and multimodal recognition. Subsequently, we extend our analysis to generative tasks, such as image generation using diffusion models [40] and text generation with large language models [12].

4.1. Experiments on Image Classification

Settings. We evaluated our method on image classification tasks using the ImageNet dataset [8], leveraging various backbone architectures including ViT [11], DeiT [48], Swin [34], and ResNet [17]. We randomly sampled 512 images from the training set to initialize the parameters of the QwT modules using Equation 6. In all networks, the affine transformation matrix W in QwT is implemented in FP16 format to reduce model size. In ResNet, W is further simplified as a group-wise convolution using a kernel size of 1 and 64 channels per group, achieving additional efficiency in storage and computation. Note that a group-wise convolution is still a linear operator, which can be perfectly encoded by the pair (W, b). Other details were consistent with prior work [27]. Please refer to the appendix for more information.

Results on different backbones. Table 1 summarizes the quantization results when applying QwT across different backbone architectures. Specifically, we selected RepQ-ViT [27] and Percentile [23] as the baseline methods for the Transformer family [11, 34, 48] and ResNet [17], respectively. The results show that incorporating the QwT module consistently boosts the recognition accuracy, leading to an average increase of approximately 2.6%, and even up to 5% for 4-bit quantization, highlighting that QwT is particularly effective for low-bit scenarios. Additionally, after the QwT modules are integrated, the accuracy in 6-bit quantization cases aligns closely with prior state-of-the-art approaches [37, 46].

Table 1. Quantization results on the ImageNet dataset [8]. '#Bits' indicates the bit-width of weights/activations. 'Size' (MB) represents the storage cost of the model on the hard disk. '*' denotes QwT modules and classification head are finetuned for one epoch. '†' indicates that the previous state-of-the-art results are directly sourced from the papers [37, 46] due to the unavailability of their official code implementations.

Network	Method	#Bits	Size	Top-1
DeiT-T	Full-precision	32/32	22.9	72.2
	IGQ-ViT† [37]	4/4	-	62.5
	RepQ-ViT [27]	4/4	3.3	58.2
	RepQ-ViT + QwT	4/4	4.2	61.4
	RepQ-ViT + QwT*	4/4	4.2	**64.8**
	IGQ-ViT† [37]	6/6	-	71.2
	RepQ-ViT [27]	6/6	4.6	71.0
	RepQ-ViT + QwT	6/6	5.5	71.2
	RepQ-ViT + QwT*	6/6	5.5	**71.6**
Swin-T	Full-precision	32/32	113.2	81.4
	IGQ-ViT† [37]	4/4	-	77.8
	RepQ-ViT [27]	4/4	14.9	73.0
	RepQ-ViT + QwT	4/4	19.2	75.5
	RepQ-ViT + QwT*	4/4	19.2	**79.3**
	IGQ-ViT† [37]	6/6	-	80.9
	RepQ-ViT [27]	6/6	21.7	80.6
	RepQ-ViT + QwT	6/6	26.0	80.7
	RepQ-ViT + QwT*	6/6	26.0	**80.9**
ViT-B	Full-precision	32/32	346.3	84.5
	IGQ-ViT† [37]	4/4	-	**79.3**
	RepQ-ViT [27]	4/4	44.9	68.5
	RepQ-ViT + QwT	4/4	59.1	76.3
	RepQ-ViT + QwT*	4/4	59.1	78.5
	IGQ-ViT† [37]	6/6	-	83.8
	RepQ-ViT [27]	6/6	66.2	83.6
	RepQ-ViT + QwT	6/6	80.4	83.9
	RepQ-ViT + QwT*	6/6	80.4	**84.0**
ResNet-50	Full-precision	32/32	102.2	76.6
	CL-Calib† [46]	4/4	-	75.4
	Percentile[23]	4/4	14.0	68.4
	Percentile + QwT	4/4	16.0	74.5
	Percentile + QwT*	4/4	16.0	**75.8**
	CL-Calib† [46]	6/6	-	-
	Percentile[23]	6/6	19.9	76.0
	Percentile + QwT	6/6	21.9	76.8
	Percentile + QwT*	6/6	21.9	**76.8**

The potential of our QwT method can be further unlocked through finetuning. By jointly optimizing the QwT modules and the classification head for only one (1) additional epoch (results marked with *), more gains in accuracy are achieved, enabling our method to surpass previous state-of-the-art results in nearly all cases.

These results are even closer to those produced by QAT methods, which typically require extensive training (e.g., 200 epochs). In contrast, our QwT* achieves similar performance with only one epoch of finetuning. Our approach not only substantially improves training efficiency but also keeps the backbone parameters unchanged, making it more suitable for hardware deployment.

Table 2. Results of 8-bit quantization, using tensor-wise Percentile [23] as the baseline PTQ method. 'Latency' (ms) is measured on a single RTX 3090 GPU with a batch size of 64, utilizing Nvidia's TensorRT [39] toolkit for deployment.

Network	Method	Size	Latency	Top-1
DeiT-T	Full-precision	22.9	11.6	72.2
	Percentile [23]	5.9	2.8	71.2
	Percentile + QwT	6.8	3.2	71.5
Swin-T	Full-precision	113.2	34.5	81.4
	Percentile [23]	28.6	9.5	80.8
	Percentile + QwT	32.9	10.9	81.0
Swin-S	Full-precision	198.4	61.0	83.2
	Percentile [23]	50.1	16.0	82.1
	Percentile + QwT	58.0	17.9	83.0
ViT-S	Full-precision	88.2	28.3	81.4
	Percentile [23]	22.5	5.8	79.2
	Percentile + QwT	26.0	6.6	80.1
ViT-B	Full-precision	346.3	85.3	84.5
	Percentile [23]	87.4	15.5	75.8
	Percentile + QwT	101.6	17.5	82.8

In Table 2, we additionally report the inference latency of different models directly deployed on a GPU. Compared to full-precision models, naive quantized models achieve an average reduction of 77% in inference latency and 75% in model size. When QwT modules are incorporated, these reductions slightly decrease to 74% and 71%, respectively, with an overhead of only 3%. This minimal additional cost is offset by an average 1.9 percentage points improvement in recognition accuracy, demonstrating the strong practicality of the QwT method.

Results across various PTQ methods. We extended our experiments to evaluate the versatility of QwT by applying it to various PTQ methods. As shown in Table 3, we integrated QwT into PTQ4ViT [52], RepQ-ViT [27], and Percentile [23], using ViT-B as the backbone.

We observe that QwT consistently enhances top-1 accuracy across all baseline PTQ methods. Notably, in 4-bit scenarios, PTQ4ViT demonstrates an improvement of approximately 40%, while RepQ-ViT shows an 8% increase. Compared to modern PTQ methods [27, 37, 52], which often involve complex and tedious procedures, our method demonstrates high simplicity and, most importantly, is compatible with all these approaches, too. The significant improvement in accuracy narrows the performance gap between different PTQ methods, and offers new insights into the design of new paradigms for network quantization.

Extension to QAT methods. We further investigated the potential of adapting QwT to QAT methods. Specifically,

Table 3. Quantization results among different PTQ methods on the ImageNet dataset [8] using ViT-B [11] as the backbone.

Method	#Bits	Size	Top-1
Full-precision	32/32	346.3	84.5
PTQ4ViT [52]	4/4	44.9	30.7
PTQ4ViT+QwT	4/4	59.1	**70.0**
RepQ-ViT [27]	4/4	44.9	68.5
RepQ-ViT + QwT	4/4	59.1	**76.3**
Percentile [23]	6/6	66.2	56.7
Percentile+QwT	6/6	80.4	**79.8**
PTQ4ViT [52]	6/6	66.2	81.7
PTQ4ViT+QwT	6/6	80.4	**83.2**
RepQ-ViT [27]	6/6	66.2	83.6
RepQ-ViT + QwT	6/6	80.4	**83.9**
Percentile [23]	8/8	87.4	75.8
Percentile+QwT	8/8	101.6	**82.8**

Table 4. Quantization results of applying QwT finetuning schema on QAT methods.

Network	Method	#Bits	Top-1
DeiT-S	Q-ViT [25]	2/2	72.1
	Q-ViT + QwT*	2/2	**72.5**
	Q-ViT [25]	3/3	79.0
	Q-ViT + QwT*	3/3	**79.1**

we applied QwT modules to QAT models after completing QAT training to assess whether QwT can further enhance recognition accuracy.

We preliminarily found that for QAT models, the initialization process described by Equation 6 is no longer effective. Applying it directly to QAT models significantly degrades accuracy. We attribute this to the fact that, unlike full-precision models, the optimization state of a QAT-trained model is sufficiently converged, resulting in almost no information loss from y to $y^{\mathbb{Z}}$. In fact, $y^{\mathbb{Z}}$ may even outperform y, as QAT models sometimes surpass their full-precision counterparts in evaluation accuracy.

To integrate QwT into QAT methods, we therefore initialize W and b to zero as a compromise. We then explore whether fine-tuning QwT can still improve recognition accuracy. For this study, we use Q-ViT [25], a representative QAT method for ViT backbones, as the baseline. The results in Table 4 demonstrate that, even without using the initialization from Equation 6, fine-tuning the QwT modules consistently enhances QAT models, confirming the generalizability of our approach.

4.2. Experiments on Object Detection & Instance Segmentation

Settings. We evaluated our method on object detection and instance segmentation tasks using the COCO 2017 [30] dataset. ResNet50 [17] with DETR [4], Swin-S [34] with Mask R-CNN [18], and Swin-S/B [34] with Cascade Mask R-CNN [3] were used as detectors. The evaluation metric was Average Precision (AP). Similar to image classification, we randomly selected 512 images from the training set to initialize the QwT weights and biases. For ResNet, the QwT was implemented using group-wise convolution with a kernel size of 1 and 64 channels per group to balance model size and AP. For DETR, we used MinMax as the PTQ baseline, a classic method that quantizes the model based on the range between the minimum and maximum values of weights or activations. For the other detectors, RepQ-ViT [27] was chosen as the baseline PTQ method.

Main results. Table 5 presents the results of applying QwT to object detection and instance segmentation tasks. We observe that QwT consistently enhances both AP^{box} and AP^{mask} across *all* cases *without finetuning*, achieving an average improvement of 0.4% with individual gains ranging from 0.1% to 0.7%. The consistent improvement underscores the robustness of our method for both object detection and instance segmentation tasks. Notably, in certain 6-bit scenarios, such as on Cascade Mask R-CNN, QwT even achieves AP comparable to full-precision models.

Additionally, a clear trend emerges where the AP gains introduced by QwT increases along with model size. For instance, in AP^{box}, the average improvement achieved by QwT rises from 0.3% in ResNet-50+DETR to 0.5% in Swin-B+Cascade Mask R-CNN, indicating the method's enhanced effectiveness in larger models.

Compared to full-precision models, baseline PTQ methods yield an average storage reduction of approximately 80%. The introduction of QwT modules slightly reduces this savings to around 78% (-2%), which demonstrates that QwT enhances AP metrics with negligible overhead.

4.3. Experiments on Multimodal Recognition

Settings. We conducted experiments using OpenAI's CLIP model [41]. Known for its exceptional zero-shot performance on the ImageNet [8] classification task, CLIP serves as an ideal benchmark for assessing the effectiveness on multimodal recognition tasks. We selected the variant of CLIP that includes a ViT-B/32 [11] as the visual encoder and a 12-block Transformer [49] as the text encoder. Since, to the best of our knowledge, no publicly available PTQ implementation exists for CLIP, we developed a baseline using RepQ-ViT [27]. We randomly selected 512 image-text pairs from the training data, both for PTQ model calibration and QwT initialization. Thanks to the simplicity and efficiency of our method, it achieved significant improvements under 30 seconds, as detailed in Table 6.

Main results. We conducted experiments with two quantization strategies: quantizing 1) only the visual encoder and 2) both visual and text encoders. As shown in Table 6, baseline PTQ methods showed significant drop in top-

Table 5. Quantization results on the COCO dataset [8]. We use box average precision (AP^{box}) and mask average precision (AP^{mask}) to assess object detection and instance segmentation accuracy, respectively.

Network	Method	#Bits	Size	AP^{box}	AP^{mask}
ResNet-50 + DETR	Full-precision	32/32	164.5	42.0	-
	MinMax	6/6	47.4	39.5	-
	MinMax + QwT	6/6	49.4	**40.0**	-
	MinMax	8/8	56.4	41.6	-
	MinMax + QwT	8/8	58.4	**41.7**	-
Swin-S + Mask R-CNN	Full-precision	32/32	276.5	48.5	43.3
	RepQ-ViT [27]	4/4	36.1	42.6	40.0
	RepQ-ViT + QwT	4/4	44.0	**43.1**	**40.4**
	RepQ-ViT [27]	6/6	53.3	47.6	42.9
	RepQ-ViT + QwT	6/6	61.2	**48.0**	**43.1**
Swin-S + Cascade Mask R-CNN	Full-precision	32/32	427.8	51.9	45.0
	RepQ-ViT [27]	4/4	56.9	49.3	43.1
	RepQ-ViT + QwT	4/4	64.8	**49.9**	**43.4**
	RepQ-ViT [27]	6/6	83.4	51.4	44.6
	RepQ-ViT + QwT	6/6	91.3	**51.7**	**44.8**
Swin-B + Cascade Mask R-CNN	Full-precision	32/32	579.9	51.9	45.0
	RepQ-ViT [27]	4/4	76.1	49.3	43.1
	RepQ-ViT + QwT	4/4	90.1	**50.0**	**43.7**
	RepQ-ViT [27]	6/6	112.1	51.5	44.8
	RepQ-ViT + QwT	6/6	126.1	**51.8**	**45.0**

1 accuracy compared to their full-precision counterparts, struggling to effectively represent a low-bit CLIP model. The reduction in performance is especially obvious when both the visual and text encoders are quantized.

In contrast, our QwT method enhanced top-1 accuracy across all cases, significantly bridging the accuracy gap between low-bit and full-precision models. Specifically, in vision-only quantization, QwT increased top-1 accuracy by an average of 0.6%, with only a modest 4% increase in model size compared to baseline PTQ methods.

When both the visual and text encoders are quantized, baseline PTQ methods exhibited an average accuracy drop of 29.2%. In contrast, QwT provided a significant accuracy improvement, with an average increase of 14.8%. These findings highlight QwT's effectiveness in preserving high accuracy while substantially reducing model size for multimodal recognition tasks.

4.4. Experiments on Image Generation

QwT has also demonstrated efficacy in generative models, notably enhancing the performance of quantized diffusion models. Unlike classifiers or detectors, which require a single forward pass, diffusion models involve multiple forward passes to generate the final images, presenting a unique prototype. Under these circumstances, QwT has proven itself highly effective, underscoring its general applicability and robustness.

Table 6. Quantization results of CLIP for zero-shot classification tasks on ImageNet. The 'Quant Setup' column differentiates between two strategies: *quantizing only the vision encoder* and *quantizing both the vision and text encoders concurrently*.

Quant Setup	Method	#Bits	Size (MB)	Top-1
Vision	Full-precision	32/32	607.2	63.4
	RepQ-ViT [27]	6/6	323.5	59.2
	RepQ-ViT + QwT	6/6	336.8	**60.3**
	RepQ-ViT [27]	8/8	345.3	62.9
	RepQ-ViT + QwT	8/8	359.5	**63.0**
Vision & Text	Full-precision	32/32	607.2	63.4
	RepQ-ViT [27]	6/6	200.8	29.8
	RepQ-ViT + QwT	6/6	221.3	**43.5**
	RepQ-ViT [27]	8/8	232.1	38.7
	RepQ-ViT + QwT	8/8	252.6	**54.6**

Settings. For our experiments, we selected the influential DiT [40] (Diffusion Transformer) architecture, following the experimental setup of Q-DiT [5]. Specifically, we employed pretrained DiT-XL/2 models at a resolution of 256×256. For rapid and precise sampling, we utilized the DDIM sampler with 50 sampling steps and applied classifier-free guidance (cfg) of 1.5, abbreviated as DiT-XL/2 (steps = 50, cfg = 1.5). Our experiments included two quantization configurations: W8A8 and W4A8. Additional results involving various model sizes, steps, and cfg values are available in the appendix.

We applied QwT directly to the quantized diffusion model using Q-DiT. A key consideration is that the model performs T forward passes per inference, with notable variation in the activation distribution and range across steps. A key assumption is that quantization error is primarily dependent on the input x, with minimal influence from elements like the time step or class condition. Accordingly, we set $t = 0$ to initialize the compensation module. The results are presented in Table 7.

Main results. Our method was compared with three representative quantization techniques: RepQ-ViT, GPTQ and Q-DiT designed for diffusion models. For both W8A8 and W4A8 settings, QwT significantly enhanced the performance of the quantized models, yielding improvements of 0.10 and 0.69 in FID, which illustrates the efficacy of QwT with minimal increase in model size.

We visualize the images generated by our model alongside those from compared models in Figure 2. The three rows represent the original images, quantized images with Q-DiT, and quantized images with QwT, respectively. All models are based on DiT-XL/2 (steps = 50, cfg = 1.5). To enable a fair comparison, we ensure that the initial Gaussian noise and the noise added during inference are identical across all methods. The images produced by our method show a closer visual resemblance to the original model,

Figure 2. Qualitative visualization results of quantizing DiT-XL/2.

Table 7. Quantitative results of quantizing DiT-XL/2. ↓ (↑) means smaller (larger) is better.

Method	#Bits	Size (MB)	FID (↓)	IS (↑)
Full-precision	16/16	1349	5.32	236.17
RepQ-ViT	8/8	677	5.46	234.74
GPTQ	8/8	690	5.90	218.90
Q-DiT	8/8	683	5.45	236.52
Q-DiT + QwT	8/8	707	**5.35**	**236.91**
RepQ-ViT	4/8	339	319.68	2.20
GPTQ	4/8	351	9.94	166.35
Q-DiT	4/8	347	6.75	208.38
Q-DiT + QwT	4/8	361	**6.06**	**215.70**

which aligns with the quantitative results.

4.5. Experiments on Large Language Models

Settings. We evaluated our framework on the LLaMA3-8B [12] model. For PTQ methods, we adopted GPTQ [14] with INT4 weight quantization. Our approach is also compatible with other PTQ methods such as AWQ [29] and SPQR [9]. We conducted a group-wise asymmetric quantization with a group size of 128 and apply activation reordering. In particular, GPTQ take 128 samples from the C4 dataset as calibration sets, and each sample is 2048 tokens long. We use the same calibration set when performing QwT after GPTQ algorithm.

Evaluation metrics. Following the settings of GPTQ, we evaluated the perplexity on the WikiText2 [47] and C4 [42] datasets. We further assessed the zero-shot commonsense question answering (QA) ability on eight tasks covering SIQA [45], HellaSwag [53], PIQA [2], WinoGrande [44], ARC [7], BoolQ [6], and OpenBookQA [36]. We also evaluated both the zero-shot and five-shot performance of the LLMs on Massively Multitask Language Understanding (MMLU) benchmark [20]. It consists of 57 language tasks including humanities, STEM, social science, *etc*. We adopted lm-eval-harness [15] to produce the accuracy results.

Table 8. Quantization results among WikiText2, C4 and eight zero-shot commonsense QA datasets using LLaMA3-8B as the backbone. ↓ (↑) means smaller (larger) is better.

Method	#Bits	Size (GB)	W2 (↓)	C4 (↓)	QA. Avg (↑)
Full-precision	16	16.06	6.24	8.96	66.10
GPTQ	4	5.73	6.65	9.44	64.90
GPTQ + QwT	4	6.80	**6.63**	**9.38**	**65.18**

Results. Table 8 summarizes the perplexity in WikiText2, C4 and the average accuracy in eight common sense reasoning datasets. More results are shown in the appendix. Note that we abbreviate WikiText2 to W2. As the results show, our optimized models will not overfit the calibration dataset and consistently outperform the original PTQ models. These results reveal the effectiveness of our QwT.

5. Conclusions

In this paper, we proposed Quantization without Tears (QwT), a novel approach that incorporates a lightweight structure into quantized models to compensate for the information loss during network quantization. The QwT modules, implemented as a tiny set of linear layers and seamlessly integrated into backbone blocks, achieved accuracy, simplicity, and generality simultaneously. Notably, QwT provides a closed-form solution to complete the compensation process in under 2 minutes and enables effortless integration with existing quantization techniques. Extensive experiments demonstrated QwT's exceptional effectiveness and versatility across a wide range of tasks, models, and quantization methods, advancing a streamlined and flexible paradigm for network quantization.

Acknowledgments

This research was partly supported by the National Natural Science Foundation of China under Grant 62276123.

J.W. designed the compensation insight and the QwT framework mathematically. M.F. made them into algorithms and codes that work well in practice, and carried out the main empirical validations. H.Y., J.S. and J.Z. carried out experiments and validations on LLM, AIGC and multimodal tasks, respectively. K.Z. engaged in discussions. All authors contributed to paper writing.

References

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

[2] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence*, pages 7432–7439, 2020. 8

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6154–6162, 2018. 6

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with Transformers. In *European Conference on Computer Vision*, pages 213–229, 2020. 2, 6

[5] Lei Chen, Yuan Meng, Chen Tang, et al. Q-DiT: Accurate post-training quantization for diffusion Transformers. *arXiv:2406.17343*, 2024. 7

[6] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2924–2936, 2019. 8

[7] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457*, 2018. 8

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 4, 5, 6, 7

[9] Tim Dettmers, Ruslan A. Svirschevski, Vage Egiazarian, et al. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *International Conference on Learning Representations*, pages 1–13, 2024. 8

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, 2019. 1

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, pages 1–21, 2021. 1, 2, 3, 4, 6

[12] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and others. The llama 3 herd of models. *arXiv:2407.21783*, 2024. 2, 4, 8

[13] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations*, pages 1–10, 2020. 1, 2

[14] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained Transformers. In *International Conference on Learning Representations*, pages 1–12, 2023. 2, 8

[15] Leo Gao, Jonathan Tow, Stella Biderman, et al. A framework for few-shot language model evaluation, 2021. 8

[16] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems*, page 1135–1143, 2015. 1

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 2, 3, 4, 6

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision*, pages 2980–2988, 2017. 1, 2, 6

[19] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision*, pages 1398–1406, 2017. 1

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, pages 1–10, 2021. 8

[21] Junghyup Lee, Dohyung Kim, and Bumsub Ham. Network quantization with element-wise gradient scaling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6444–6453, 2021. 1, 2

[22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations*, pages 1–12, 2017. 1

[23] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2805–2814, 2019. 4, 5, 6

[24] Yuhang Li, Ruihao Gong, Xu Tan, et al. BRECQ: Pushing the limit of post-training quantization by block reconstruction. In *International Conference on Learning Representations*, pages 1–12, 2021. 2

[25] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-ViT: Accurate and fully quantized low-bit Vision Transformer. In *Advances in Neural Information Processing Systems*, pages 34451 – 34463, 2024. 1, 2, 6

[26] Zhikai Li and Qingyi Gu. I-ViT: Integer-only quantization for efficient Vision Transformer inference. In *IEEE/CVF International Conference on Computer Vision*, pages 17019–17029, 2023. 1

[27] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. RepQ-ViT: Scale reparameterization for post-training quantization of Vision Transformers. In *IEEE/CVF International Conference on Computer Vision*, pages 17181–17190, 2023. 1, 2, 3, 4, 5, 6, 7

[28] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461(C): 370–403, 2021. 1, 2

[29] Ji Lin, Jiaming Tang, Haotian Tang, et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration. In *Annual Conference on Machine Learning and Systems*, pages 87–100, 2024. 8

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014. 6

[31] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. FQ-ViT: Post-training quantization for fully quantized Vision Transformer. In *International Joint Conference on Artificial Intelligence*, pages 1173–1179, 2022. 1, 2

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

[33] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. ReActNet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision*, pages 143–159, 2020. 1, 2

[34] Ze Liu, Yutong Lin, Yue Cao, et al. Swin Transformer: Hierarchical Vision Transformer using shifted windows. In *IEEE/CVF International Conference on Computer Vision*, pages 9992–10002, 2021. 2, 4, 6

[35] Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. Post-training quantization for Vision Transformer. In *Advances in Neural Information Processing Systems*, pages 28092 – 28103, 2024. 2

[36] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, 2018. 8

[37] Jaehyeon Moon, Dohyung Kim, Junyong Cheon, and Bumsub Ham. Instance-aware group quantization for Vision Transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16132–16141, 2024. 1, 2, 4, 5

[38] 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, 2020. 2, 3

[39] NVIDIA Corporation. *NVIDIA TensorRT*, 2024. https://developer.nvidia.com/tensorrt. 5

[40] William Peebles and Saining Xie. Scalable diffusion models with Transformers. In *IEEE/CVF International Conference on Computer Vision*, pages 4172–4182, 2023. 2, 4, 7

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763, 2021. 1, 2, 6

[42] Colin Raffel, Noam Shazeer, Adam Roberts, et al. Exploring the limits of transfer learning with a unified text-to-text Transformer. *Journal of Machine Learning Research*, 21 (140):1–67, 2020. 8

[43] John O Rawlings, Sastry G Pantula, and David A Dickey. *Applied regression analysis: a research tool*. Springer, 1998. 4

[44] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: an adversarial winograd schema challenge at scale. *Communications of the ACM*, 64 (9):99–106, 2021. 8

[45] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*, pages 4463–4473, 2019. 8

[46] Yuzhang Shang, Gaowen Liu, Ramana Rao Kompella, and Yan Yan. Enhancing post-training quantization calibration through contrastive learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15921–15930, 2024. 1, 4, 5

[47] Merity Stephen, Xiong Caiming, Bradbury James, et al. Pointer sentinel mixture models. In *International Conference on Learning Representations*, pages 1–11, 2017. 8

[48] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357, 2021. 4

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, page 6000–6010, 2017. 6

[50] 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*, pages 1–12, 2022. 2

[51] Zhuguanyu Wu, Jiaxin Chen, Hanwen Zhong, Di Huang, and Yunhong Wang. AdaLog: Post-training quantization for Vision Transformers with adaptive logarithm quantizer. In *European Conference on Computer Vision*, 2024. 3

[52] Zhihang Yuan, Chenhao Xue, Yiqi Chen, Qiang Wu, and Guangyu Sun. PTQ4ViT: Post-training quantization for Vision Transformers with twin uniform quantization. In *European Conference on Computer Vision*, pages 191–207, 2022. 1, 2, 3, 5, 6

[53] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish

your sentence? In *Annual Meeting of the Association for Computational Linguistics*, page 4791–4800, 2019. 8

[54] Ke Zhu, Yin-Yin He, and Jianxin Wu. Quantized feature distillation for network quantization. In *AAAI Conference on Artificial Intelligence*, pages 11452–11460, 2023. 1