Fixing Quantization with Lightweight Adapters

Mohammadreza Mohammadi, Matthew Grenier, Ramtin Zand
University of South Carolina
Columbia, SC, USA
MOHAMMM@email.sc.edu, MGRENIER@email.sc.edu, RAMTIN@cse.sc.edu

Abstract

Quantization is an effective strategy for compressing deep neural networks by reducing numerical precision. However, post-training quantization (PTQ) often suffers from significant accuracy degradation at low bit-widths, while quantization-aware training (QAT) is computationally expensive and requires full retraining. We propose a lightweight, sublayer-aware compensation framework that inserts small low-rank adapters into transformer blocks to correct for quantization errors. The adapters are optimized using a hybrid objective combining supervised learning, knowledge distillation, and feature reconstruction to align the quantized model with its full-precision counterpart. Our approach requires only minimal tuning on a small subset of training data and introduces less than 1% overhead for large models and less than 5% for tiny models, while effectively recovering accuracy lost to low-bit quantization. Extensive experiments across vision and language models, including ViT, Swin, BERT, and GPT-2, demonstrate state-of-the-art performance under aggressive precision settings, and in some cases even surpass the original FP32 accuracy. These results provide a practical pathway for deploying highly quantized models with near- or above-full-precision performance.

1. Introduction

Deep neural networks (DNNs) have become the foundation of modern artificial intelligence (AI), achieving state-of-the-art performance across a wide range of domains such as computer vision [9, 13] and natural language processing [5, 30]. These advances, however, have been accompanied by rapidly growing computational and memory demands. As model sizes and complexities continue to increase, deploying DNNs on resource-constrained edge devices has become increasingly challenging, with many existing edge-AI platforms becoming unable to support inference for large-scale models [33].

Addressing this issue, Quantization [12] offers a path towards efficient model deployment by reducing numerical precision of model weights and activations. In quantized networks, weights and activations are typically represented using low-bit integers (e.g., 8-bit or 6-bit) instead of 32-bit or 16-bit floating-point values, leading to both memory and computational savings. Quantization is well supported across modern AI accelerators and can deliver high compression ratio. Among existing techniques, Post-Training Quantization (PTQ) [1, 11, 18, 20, 21, 26, 27, 40] provides a fast and convenient workflow as it avoids any additional fine-tuning, whereas Quantization-Aware Training (QAT) [2, 8, 10, 19] can achieve higher accuracy under aggressive bitwidth reduction at the cost of additional training time. Despite these advantages, quantization remains a challenging problem. PTQ often incurs a noticeable loss in accuracy, especially under aggressive bitwidth reduction [1, 28], while QAT, though more accurate, is computationally expensive and requires retraining the network. Furthermore, many existing quantization frameworks involve intricate optimization procedures or are heavily customized for specific architectures, limiting their general applicability and ease of adoption in practical workflows [24].

To address these limitations, we propose *Quantization with Lightweight Adapters (QLA)*, a simple yet highly effective framework that enhances the accuracy of quantized neural networks with minimal overhead. QLA preserves the efficiency of PTQ while providing a significant accuracy boost, enabling a fast and reliable quantization pipeline. The central idea of QLA is to augment a quantized model with small, low-rank adapter modules that compensate for quantization-induced errors. These lightweight adapters aim to offset quantization error without significantly increasing model size or training cost. As a result, QLA produces quantized models whose accuracy closely matches, or even surpasses, that of their full-precision counterparts, achieving a balance between compactness and performance. The key contributions of this work are as follows:

- We propose a post-training quantization recovery method based on lightweight low-rank adapters that can be seamlessly applied to a wide range of DNNs without modify-

ing the original model parameters.

- We introduce a unified loss formulation that combines supervised learning, knowledge distillation, and feature matching. Unlike prior approaches constrained by full-precision teacher guidance, our hybrid objective enables the quantized model to recover, and in some cases, exceed FP32 accuracy after compensation.
- We validate QLA across diverse architectures and domains, including both vision and language models. Extensive ablations show that very low-rank adapters (e.g., $r = 1$–8) are sufficient to recover accuracy, introducing $\sim 1\%$ parameter overhead for large models and $\sim 5\%$ for smaller models.

2. Related Works

Quantization reduces the bit-width of weights and activations by mapping continuous, full-precision values to a smaller discrete set [12]. To maintain accuracy under reduced precision, many approaches employ QAT, which retrains models to adapt to lower bit widths [2, 8, 10, 19]. Although QAT yields strong accuracy, it is computationally demanding and time-consuming, motivating the use of PTQ as a more practical alternative. However, naive PTQ often suffers from severe accuracy degradation, particularly under aggressive quantization (e.g., 4-bit or lower) [1, 28]. This issue is exacerbated in attention-based architectures such as vision transformers and large language models (LLMs), where activations exhibit high variance and heavy-tailed distributions with numerous outliers [10, 40].

To address these challenges, various PTQ techniques have been proposed. AdaRound [27] introduces learnable rounding through continuous relaxation of discrete weights, while RepQ-ViT [21] tailors quantization to transformer models by applying complex quantizers during calibration and later reparameterizing them into hardware-friendly forms. Data-free methods such as CLAMP-ViT [32] leverage contragstive synthetic data generation to improve calibration without access to training data. More recently, Quantization without Tears (QwT) [11] proposed adding auxiliary compensation modules to recover information lost in PTQ, removing the need for the quantized model to mirror the full-precision structure. While effective, QwT considerably increase model parameters, limiting its efficiency. A more recent variant, QwT-v2 [36], attempts to address this issue by replacing the full compensation matrices with lightweight channel-wise affine corrections. Although this design significantly reduces parameter overhead, it also limits the expressiveness of the compensation and, in many cases, results in lower accuracy compared to QwT.

Knowledge distillation (KD) transfers information from a large teacher model to a smaller student model to enhance accuracy without increasing inference cost. Two main KD paradigms are commonly used: response-based methods, which train the student to match the teacher's output predictions [14, 25, 35], and feature-based methods, which align intermediate representations between teacher and student [16, 34]. Recent works [38, 41] show that combining these approaches provides complementary benefits, leading to stronger performance than either alone.

Low-rank adaptation (LoRA) is a parameter-efficient fine-tuning technique that adds small trainable low-rank matrices to a frozen backbone model, substantially reducing the number of trainable parameters and thus minimizing both memory and computational costs [15]. LoRA has been successfully extended to a range of architectures, including Vision Transformers (ViTs) [42], convolutional neural networks (CNNs) [6], and diffusion models [39]. Although our method makes use of low-rank adapters, our aim is not to fine-tune a given model on a new task as in a traditional LoRA approach. Instead, we train adapters to minimize quantization error on the *same* task, leading to increased accuracy when compared to a PTQ baseline.

3. Methods

3.1. Preliminaries

In PTQ, a quantization function (F_q) maps floating-point values (x) (representing either weights or activations) to their corresponding fixed-point representations (x_{int}), i.e., ($F_q(x) \rightarrow x_{\text{int}}$). Given a quantization bit-width (b), the uniform PTQ procedure is formulated as:

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

where ($\lfloor \cdot \rceil$) denotes the rounding operation, s represents the scaling factor, and z denotes the zero-point. The clamp function constrains the quantized value to the valid range [0, $2^b - 1$]. The quantization parameters s and z are determined as follows:

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

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

Due to the clamp and rounding functions, quantization inherently introduces information loss when mapping floating-point values x to their quantized counterparts x_{int}. This loss becomes more pronounced in deeper models, where the quantization errors accumulate across layers, leading to a noticeable degradation in overall model performance. Recently, QwT [11] introduced an alternative approach to compensate for the discrepancy between the quantized and full-precision models without retraining the entire network. Given the structure of the floating-point

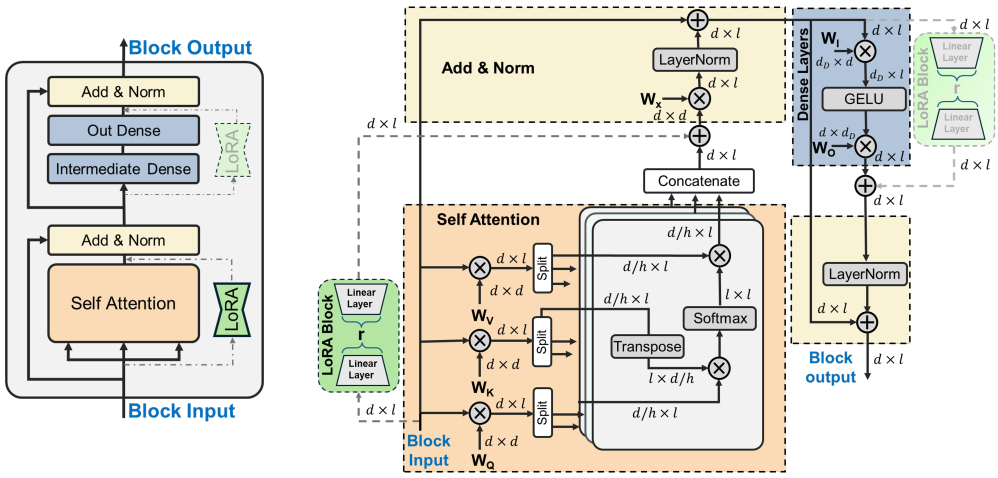

Figure 1. (a) High-level transformer block with parallel lightweight low-rank adapters. (b) Detailed design showing sub-layer-specific compensation correcting noise induced by quantization, with less than 1% overhead.

model S_{FP} and its quantized counterpart S_{int}, QwT proposes that the quantized model does not need to strictly preserve the original architecture, i.e., $S_{\text{FP}} \neq S_{\text{int}}$.

To mitigate information loss introduced by quantization, QwT augments the quantized structure with compensation modules S_{comp} that operate in parallel with existing blocks. Each compensation module is implemented as a linear layer placed in parallel to a model block, receives the same input, and produces an output with identical dimensionality. Let $M_{\text{FP}}(\cdot)$ and $M_{\text{int}}(\cdot)$ denote the functions implemented by the floating-point and quantized models, respectively. The parameters of the compensation modules are obtained by minimizing the following regression objective:

$$\mathcal{L}_{\text{comp}} = \|M_{\text{FP}}(x) - M_{\text{int}}(x)\|_2^2, \tag{4}$$

where x is an input sample and the norm is computed at the chosen alignment point (e.g., block or network output).

3.2. SOTA Limitations

Although conceptually simple, QwT [11], the current state-of-the-art (SOTA) approach, suffers from several inefficiencies. QwT performs block-level compensation using large transformation matrices for each transformer block, which introduces substantial parameter overhead. Moreover, the parameters of these matrices are estimated using linear regression, requiring solving large systems whose size grows with the number of calibration samples, leading to significant memory and computational costs. In contrast, our method employs low-rank adapter modules embedded within transformer sub-modules (*i.e.* attention or FFN layers), avoiding the scalability issues associated with large, full-rank weight matrices, substantially reducing parameter overhead.

3.3. Implementation

Considering the limitations of existing methods, we add lightweight compensation modules that attach within transformer sub-layers. As shown in Figure 1a, the transformer block is augmented with one, or optionally two, lightweight adapters: one parallel to the attention block and another to the feed-forward network. These modules learn corrections for quantization error while keeping the original block structure and data flow intact.

Figure 1b illustrates how these adapters interact with the transformer pathways. The LoRA [15] blocks are not merged with the internal modules of each sub-layer. Instead, they operate as parallel correction paths that receive the same input and produce outputs with the same dimensionality as the original sub-layers, making our method model-agnostic.

Let $X \in \mathbb{R}^{d\times\ell}$ denote the input to a given sub-layer, where d is the embedding dimension and ℓ is the sequence length. The main sub-layer (attention or FFN) produces an output $\mathcal{F}(X) \in \mathbb{R}^{d\times\ell}$. The adapter receives the same input and computes a low-rank correction

$$\Delta(X) = \alpha BAX, \tag{5}$$

where $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{d\times r}$ with $r \ll d$ define a low-rank subspace, and α scales the update magnitude. The final sub-layer output becomes

$$\hat{Y} = \mathcal{F}(X) + \Delta(X). \tag{6}$$

During training, the quantized backbone model remains frozen, and only the parameters of the LoRA blocks are optimized. Unlike the original LoRA [15], our goal is not to adapt the model to a new task or dataset. Instead, the

quantized network is treated as a fixed backbone, and the adapters learn corrections that compensate for quantization-induced error. Since the backbone parameters are not modified, the proposed QLA methodology can be applied on top of any PTQ method.

As illustrated in Figure 1b, the LoRA block attached to the FFN sub-layer is shown transparently to reflect a design choice motivated by our ablation study. In many cases, the adapter on the attention sub-layer already recovers a large portion of the lost accuracy, while the FFN adapter provides additional but smaller improvements. Therefore, when memory is limited, using only the attention adapter can already achieve strong recovery, while including the FFN adapter can further improve performance.

Parameter Overhead. Let d denote the embedding dimension and r the adapter rank. Each low-rank module introduces $P_{\text{LoRA}} = 2dr$ additional parameters per insertion point, corresponding to the matrices $A \in \mathbb{R}^{r\times d}$ and $B \in \mathbb{R}^{d\times r}$. In practice, r is selected from a small range (e.g., $r \in \{1, 2, 4, 8, 16\}$), resulting in negligible parameter overhead (often below 1% of the backbone parameters) while still achieving substantial accuracy recovery. Notably, this overhead is dramatically smaller than full rank compensation matrices, such as those of QwT [11], which require d^2 additional parameters per block.

Loss Function Formulation. The optimization objective combines supervised learning, knowledge distillation, and feature reconstruction:

$$\mathcal{L} = \lambda_{\text{ce}}\mathcal{L}_{\text{CE}} + \lambda_{\text{kd}}\mathcal{L}_{\text{KD}} + \lambda_{\text{feat}}\mathcal{L}_{\text{FM}}. \tag{7}$$

The three loss components are defined as follows:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log p_i, \tag{8}$$

$$\mathcal{L}_{\text{KD}} = T^2 \cdot \text{KL}\left(\text{softmax}\left(\frac{z_t}{T}\right) \,\middle\|\, \text{softmax}\left(\frac{z_s}{T}\right)\right), \tag{9}$$

$$\mathcal{L}_{\text{FM}} = \frac{1}{L}\sum_{\ell=1}^{L} \left\| f_t^{(\ell)} - f_s^{(\ell)} \right\|_2^2, \tag{10}$$

where p_i is the predicted probability for class i, z_t and z_s denote the teacher and student logits, T is the distillation temperature, and $f_t^{(\ell)}$, $f_s^{(\ell)}$ are the intermediate feature maps from layer ℓ. $\mathcal{L}_{\text{CE}}$ ensures task consistency, $\mathcal{L}_{\text{KD}}$ aligns the quantized model with its full-precision teacher, and $\mathcal{L}_{\text{FM}}$ promotes local feature reconstruction between matched layers. For language models, the same formulation applies directly, with $\mathcal{L}_{\text{CE}}$ replaced by the next-token prediction loss. We include the supervised term to avoid being limited by the FP32 teacher, allowing the quantized model to potentially exceed it, as confirmed in our results.

During training, the quantized backbone remains frozen, and gradients are propagated only through the adapters, resulting in negligible computational overhead. The adapters converge rapidly, since they learn low-dimensional corrections guided by both ground-truth labels and teacher activations. In turn, this targeted fine-tuning achieves far greater performance than basic PTQ without substantial training overhead.

4. Experiments

We evaluate QLA across a diverse set of transformer-based architectures spanning both vision and language domains. Our experiments are designed to assess the generality, compatibility, and effectiveness of the proposed framework under multiple PTQ pipelines and precision settings.

For each architecture, we first obtain a quantized backbone using a chosen PTQ method. We then attach the proposed low-rank corrective modules *without modifying the underlying quantization procedure*. This setup enables us to isolate the contribution of QLA from the PTQ algorithm itself and evaluate its ability to compensate for quantization-induced error across different model families and tasks.

4.1. Vision Models

We conduct large-scale image classification experiments on ImageNet-1K [3]. All evaluations are performed using transformer-based vision architectures, including representative Vision Transformers (ViT) [9], DeiT [37], and Swin Transformers [23]. Following quantization via a chosen PTQ pipeline, the backbone parameters remain frozen, and only the low-rank modules are optimized.

To train the lightweight adapters, we use a modest subset of the ImageNet training split consisting of 40K labeled images. This subset is substantially larger than typical PTQ calibration sets, yet significantly smaller than full-dataset retraining. All adapter parameters are initialized so that the initial model behavior matches the quantized backbone. Specifically, the low-rank matrices A and B are initialized to zero, while the scaling coefficient α is initialized to 0.1. Training is conducted for a single epoch using AdamW with a learning rate of 1×10^{-3} and batch size 16. Unlike QwT, which stores the compensation matrices in FP16 to reduce model size, the proposed lightweight adapters introduce only a negligible number of additional parameters. Therefore, we store the adapter parameters in FP32 to better preserve accuracy while keeping the overall model size nearly unchanged.

For fairness, we attempted to apply the same 40,000 sample training subset used for QLA when running the public QwT implementation. Unsurprisingly though, increasing the number of samples beyond 512 did not lead to accuracy improvements for QwT-adapted models, hence we utilized 512 samples for QwT compensation as in the original work.

Table 1 reports the Top-1 accuracy and model size of QLA when applied to RepQ-ViT [21] for various vi-

Table 1. Comparison of PTQ methods on transformer-based vision models. QLA is applied to backbones quantized by RepQ-ViT [21] using a *fixed configuration* consisting of rank $r = 8$ adapters applied only to the attention sub-layers. Results for IGQ-ViT and QwT-v2 are taken from the original paper due to unavailable code, while QwT results are obtained using the authors' public implementation.

Net.	Method	#Bits	Size (MB)	Top-1
DeiT-T	Full-precision	32/32	22.90	72.2
	IGQ-ViT [26]	4/4	–	62.5
	RepQ-ViT [21]	4/4	3.27	58.2
	RepQ-ViT + QwT [11]	4/4	4.17	61.4
	RepQ-ViT + QwT-v2 [36]	4/4	3.40	59.9
	RepQ-ViT + QLA	4/4	3.46	**63.8**
	IGQ-ViT [26]	6/6	–	71.2
	RepQ-ViT [21]	6/6	4.60	71.0
	RepQ-ViT + QwT [11]	6/6	5.49	71.2
	RepQ-ViT + QwT-v2 [36]	6/6	4.70	71.2
	RepQ-ViT + QLA	6/6	4.76	**71.5**
Swin-T	Full-precision	32/32	113.20	81.4
	IGQ-ViT [26]	4/4	–	77.8
	RepQ-ViT [21]	4/4	14.84	73.0
	RepQ-ViT + QwT [11]	4/4	19.17	75.5
	RepQ-ViT + QwT-v2 [36]	4/4	15.20	77.1
	RepQ-ViT + QLA	4/4	15.20	**78.6**
	IGQ-ViT [26]	6/6	–	80.9
	RepQ-ViT [21]	6/6	21.84	80.6
	RepQ-ViT + QwT [11]	6/6	26.03	80.7
	RepQ-ViT + QwT-v2 [36]	6/6	22.00	80.9
	RepQ-ViT + QLA	6/6	22.00	**80.9**
ViT-B	Full-precision	32/32	346.30	84.5
	IGQ-ViT [26]	4/4	–	**79.3**
	RepQ-ViT [21]	4/4	44.93	68.5
	RepQ-ViT + QwT [11]	4/4	59.12	76.3
	RepQ-ViT + QwT-v2 [36]	4/4	45.60	75.6
	RepQ-ViT + QLA	4/4	45.55	77.1
	IGQ-ViT [26]	6/6	–	83.8
	RepQ-ViT [21]	6/6	66.16	83.6
	RepQ-ViT + QwT [11]	6/6	80.35	83.9
	RepQ-ViT + QwT-v2 [36]	6/6	66.90	83.8
	RepQ-ViT + QLA	6/6	66.85	**84.1**

sion transformer architectures. We compare our approach (RepQ-ViT + QLA) against baseline RepQ-ViT, IGQ-ViT [26], as well as the compensation methods QwT [11] and QwT-v2 [36] across three representative transformer architectures: DeiT-T [37], Swin-T [23], and ViT-B [9].

To ensure a fair comparison across models, QLA is evaluated using a fixed configuration consisting of rank $r = 8$ adapters inserted only in the attention sub-layers. Although different ranks and alternative placement strategies can provide additional accuracy improvements, we deliberately avoid architecture-specific tuning here. A comprehensive study of the impact of adapter rank and placement is provided in Section 4.3.

This configuration introduces a very small memory overhead, ranging from approximately 0.16 MB to 0.69 MB depending on the model size. This corresponds to approximately 5% additional parameters for smaller models and 1% for larger models. Despite this minimal overhead, QLA consistently surpasses prior compensation-based PTQ methods across most evaluated settings.

For DeiT-T [37], applying QLA to RepQ-ViT improves the 4-bit accuracy from 58.2% to **63.8%**. This result surpasses RepQ-ViT + QwT (61.4%), RepQ-ViT + QwT-v2 (59.9%), and IGQ-ViT (62.5%), while increasing the model size by only 0.19 MB compared to the RepQ-ViT baseline. In addition to achieving the highest accuracy, QLA introduces substantially less memory overhead than QwT, while remaining comparable to QwT-v2 with only about 0.06 MB additional overhead. At 6-bit precision, QLA further improves accuracy to **71.5%**, surpassing RepQ-ViT (71.0%), RepQ-ViT + QwT (71.2%), RepQ-ViT + QwT-v2 (71.2%), and IGQ-ViT (71.2%), while approaching the full-precision accuracy of 72.2%.

For Swin-T [23], QLA provides even larger improvements in the aggressive 4-bit regime, increasing the RepQ-ViT accuracy from 73.0% to **78.6%**. This result surpasses RepQ-ViT + QwT (75.5%), RepQ-ViT + QwT-v2 (77.1%), and IGQ-ViT (77.8%). Importantly, this improvement is achieved with only 0.36 MB additional memory ($\sim 2\%$ of the 14.84 MB baseline), matching the overhead of QwT-v2 while remaining significantly smaller than the 4.33 MB overhead introduced by QwT ($\sim 29\%$). At 6-bit precision, QLA reaches **80.9%** accuracy, matching the best reported result from IGQ-ViT and QwT-v2 while maintaining negligible memory overhead.

For the larger ViT-B [9] model, QLA significantly improves the RepQ-ViT baseline in the 4-bit setting, raising accuracy from 68.5% to 77.1%, a gain of +8.6%. While IGQ-ViT achieves the highest accuracy in this configuration (79.3%), QLA still surpasses the previous compensation methods RepQ-ViT + QwT (76.3%) and RepQ-ViT + QwT-v2 (75.6%). Importantly, this improvement is achieved with only 0.62 MB additional parameters (less than 1.5% of the 44.93 MB baseline model size), compared to the 14.2 MB overhead introduced by QwT ($\sim 31.6\%$) and the 0.7 MB overhead of QwT-v2. At 6-bit precision, QLA achieves **84.1%** accuracy, surpassing RepQ-ViT + QwT (83.9%), RepQ-ViT + QwT-v2 (83.8%), and IGQ-ViT (83.8%), and falling within only 0.4% of the full-precision model.

Overall, these results show that very low-rank adapters are sufficient to recover most of the accuracy loss caused by quantization in transformer architectures. With a fixed rank $r = 8$, QLA consistently improves or matches prior PTQ methods while introducing negligible parameter overhead, generally outperforming both QwT and QwT-v2.

Table 2. Compatibility of QLA with different PTQ methods and precision settings on DeiT-S. ⋆For CLAMP-ViT, the original paper reports the average quantization precision but does not provide the model size. Therefore, we estimate the model size based on the reported average weight precision.

Method	# Bits	Size(MB)	Acc (%)
Full-precision	32/32	88.20	79.9
RepQ [21]	4/4	11.85	69.0
RepQ-ViT + QLA	4/4	12.16	74.1
RepQ [21]	6/6	17.16	78.9
RepQ-ViT + QLA	6/6	17.47	79.1
Percentile [17]	6/6	17.16	65.4
Percentile + QLA	6/6	17.47	75.8
Percentile [17]	8/8	22.47	74.5
Percentile + QLA	8/8	22.78	79.0
CLAMP-Vit⋆ [32]	4.7/5.9	13.00	79.1
CLAMP-Vit⋆ + QLA	4.7/5.9	13.31	79.5

4.1.1. Compatibility with Different PTQ Methods

To evaluate the generality of QLA, we apply it to several PTQ methods operating under different quantization strategies and precisions. Table 2 reports results on DeiT-S [9] using uniform and mixed-precision quantization schemes.

First, we apply QLA to RepQ [21] under aggressive 4-bit quantization. While the baseline RepQ-ViT model achieves 69.0% accuracy, adding QLA improves the accuracy to 74.1%, recovering a large portion of the performance loss caused by quantization with only a small increase in model size (from 11.85 MB to 12.16 MB). At 6-bit precision, the RepQ-ViT baseline already achieves 78.9% accuracy, and QLA further improves it slightly to 79.1%, approaching the full-precision accuracy of 79.9%.

We also apply QLA to Percentile [17]. In the 6-bit setting, QLA greatly improves the baseline accuracy of 65.4% to 75.8% while introducing only 0.31 MB of additinal parameters. In the 8-bit setting, applying QLA improves the Percentile baseline significantly from 74.5% to 79.0% accuracy, demonstrating that QLA can effectively compensate for quantization errors even in higher precision regimes.

Finally, we apply QLA to CLAMP-ViT [32], which uses mixed-precision quantization with an average precision of 4.7 bits for weights and 5.9 bits for activations. Even in this mixed-precision setting, QLA remains compatible and further improves the accuracy from 79.1% to 79.5% while increasing the model size by only 0.32 MB.

4.1.2. Latency, Power, and Energy

Table 3 reports the performance of QLA and QwT [11] when deployed on a Jetson AGX Orin platform. All models are optimized using the NVIDIA TensorRT runtime and quantized to INT8 precision. Latency is measured using the `trtexec` benchmarking tool, while power consumption is monitored using the built in `tegrastats` utility. Latency metrics were not reported for other bit-widths and quantization methods as the Jetson AGX Orin does not support smaller than INT8 precision. Additionally, it should be made clear that these accuracy measurements represent on-device measurements, not "fake quantization" results.

Table 3. Comparison of QLA and QwT when applied to Percentile [17] using INT8 weights and activations. Latency, power, and energy measurements are obtained by deploying the models on a Jetson AGX Orin 64GB platform using NVIDIA's TensorRT [29] inference toolkit with batch size 64.

Metric	Method	DeiT-T	DeiT-S	ViT-S	ViT-B	Swin-T	Swin-S
Latency (ms)	Full-precision	42.75	91.60	91.67	235.95	124.61	206.87
	Percentile [17]	14.41	28.65	28.63	70.65	48.30	76.75
	Percentile + QwT [11]	**15.61**	**31.41**	**31.32**	76.84	**52.67**	**83.72**
	Percentile + QLA	16.17	31.90	31.74	**76.28**	54.64	87.12
Power (W)	Full-precision	39.2	44.6	44.7	47.5	40.9	41.6
	Percentile [17]	38.0	43.0	42.9	46.8	37.9	38.6
	Percentile + QwT [11]	38.4	43.0	43.1	47.1	38.2	39.1
	Percentile + QLA	**36.5**	**41.1**	**41.3**	**45.0**	**36.6**	**37.4**
Energy (J)	Full-precision	1.68	4.09	4.10	11.22	5.10	8.60
	Percentile [17]	0.55	1.23	1.23	3.31	1.83	2.96
	Percentile + QwT [11]	0.60	1.35	1.35	3.62	2.01	3.27
	Percentile + QLA	**0.59**	**1.31**	**1.31**	**3.43**	**2.00**	**3.26**
Accuracy (%)	Full-precision	72.1	79.8	81.4	84.5	81.4	83.2
	Percentile [17]	71.1	74.5	79.3	75.7	81.0	82.1
	Percentile + QwT [11]	71.7	77.7	80.2	82.8	79.8	**83.1**
	Percentile + QLA	**71.9**	**79.0**	**80.9**	**83.6**	**81.1**	82.8
Model Size (MB)	Full-precision	22.87	88.20	88.20	346.27	113.15	198.43
	Percentile [17]	5.93	22.47	22.47	87.39	28.56	50.13
	Percentile + QwT [11]	6.82	26.02	26.02	101.59	32.89	58.01
	Percentile + QLA	**6.08**	**22.78**	**22.78**	**88.02**	**28.86**	**50.74**

As expected, all quantized models significantly reduce latency compared to their full-precision counterparts. Compared to QwT, QLA does achieve slightly higher latency in general, although it performs superiorly on ViT-B. We suspect QLA generally achieves marginally higher latency than QwT due to the structure of the low-rank adapters. While the low-rank adapters clearly introduce less parameter and FLOP overhead, they still require two matrix multiplication operations which must necessarily be computed sequentially, contrasted with the single full-rank matrix of QwT. In the case that latency optimized to this degree is required, the low-rank adapters can be multiplied out to produce a full $d \times d$ matrix at the cost of increased parameters. With these new dimensions, latency should be identical to that of QwT, while keeping accuracy intact.

Despite this marginally inferior runtime behavior however, QLA consistently achieves lower power consumption and lower energy usage than QwT across all evaluated models. In addition, QLA improves the Top-1 accuracy in most cases, outperforming QwT on five out of six architectures while once again introducing far fewer additional parameters. These results demonstrate that QLA delivers higher accuracy and improved energy efficiency compared to pre-

Table 4. Performance of GPT-2 models on the SQuAD v1.1 [31] dataset under different quantization settings. We report model size and Exact Match (EM) scores for BNB, RepQ, QwT, and the proposed method across multiple bit-widths. For QLA, LoRA adapters with rank $r = 8$ are applied only to the attention layers.

Net.	Method	#Bits	Size (MB)	EM
GPT-2 S	Full-precision	32/32	497.8	59.1
	BNB [4]	4/16	62.6	56.0
	BNB + QwT [11]	4/16	76.8	57.7
	BNB + QLA	4/16	63.3	**60.2**
	RepQ [21]	6/6	93.7	49.6
	RepQ + QwT [11]	6/6	107.9	51.8
	RepQ + QLA	6/6	94.4	**57.9**
	RepQ [21]	8/8	124.8	53.6
	RepQ + QwT [11]	8/8	139.0	57.5
	RepQ + QLA	8/8	125.5	**62.0**
GPT-2 L	Full-precision	32/32	3096	71.8
	BNB [4]	4/16	389.1	74.1
	BNB + QwT [11]	4/16	403.3	72.7
	BNB + QLA	4/16	392.3	**74.7**
	RepQ [21]	6/6	582.5	73.7
	RepQ + QwT [11]	6/6	601.9	68.5
	RepQ + QLA	6/6	583.7	**74.8**
	RepQ [21]	8/8	775.8	73.6
	RepQ + QwT [11]	8/8	795.2	72.2
	RepQ + QLA	8/8	779.3	**74.4**

vious methods without introducing noticeable latency.

4.2. Language Models

To evaluate the generalization of our approach beyond vision architectures, we extend the QLA framework to language models, including GPT-2 (autoregressive) and BERT/RoBERTa (encoder-only). Evaluations are conducted on SQuAD v1.1 [31] and MRPC [7], representing question-answering and semantic similarity tasks, respectively. To maintain consistency with the vision models evaluated in this paper, QLA for language models is implemented using LoRA adapters with rank $r = 8$ applied only to the attention layers. QwT [11] employs 512 calibration samples for vision models. Implementing QwT for language models, this setting proved insufficient. With this in mind, for QwT we used the full training split to obtain richer activation statistics and better performance. Since the code for QwT-v2 is not publicly available, we are unable to include comparisons with this method for language models.

4.2.1. Generative Models

In our experiments, RepQ [21] did not yield stable results at 4-bit precision for any model tested, hence RepQ results are reported only for the 6 and 8-bit configurations. As RepQ was originally designed for vision transformers and does not quantize embedding layers, we extend its implementation to include full embedding-layer quantization, enabling end-to-end low-precision inference. Additionally, to test the performance of QLA on weight only and floating point quantization, we adopt the BitsAndBytes (BNB) NF4 quantization backend from QLoRA [4], using only the NF4 quantization path with LoRA disabled.

As shown in Table 4, GPT-2 Small [30], BNB compresses the model from 497.8 MB to 62.6 MB with an Exact Match (EM) of 56.0%. Adding QwT [11] increases the size of the model to 76.8 MB and slightly improves the performance to 57.7%. In comparison, our method achieves 60.2% EM with only 0.7 MB memory overhead. At the 6-bit setting, RepQ reaches 49.6% EM with a footprint of 93.7 MB, while adding QwT improves accuracy to 51.8% but increases memory to 107.9 MB. QLA achieves 57.9% EM with 94.4 MB, improving accuracy while maintaining memory efficiency. At 8-bit precision, RepQ achieves 53.6% EM, and RepQ + QwT improves it to 57.5% with a size of 139.0 MB. Our method in contrast reaches 62.0% EM with only 0.7 MB overhead.

For GPT-2 Large [30], BNB improves EM from 71.8% in full precision to 74.1% with a compressed model size of 389.1 MB. Adding QwT reduces accuracy to 72.7% despite increasing model size to 403.3 MB. In contrast, our approach reaches 74.7% EM with a comparable footprint (392.3 MB), surpassing full-precision and BNB-quantized performance while keeping the memory overhead negligible. At 6-bit precision, RepQ achieves 73.7% EM with a size of 582.5 MB, and RepQ+QwT reduces accuracy to 68.5% while increasing storage requirements to 601.9 MB. Our method attains 74.7% EM with only 0.2% memory overhead. At 8-bit precision, RepQ reaches 73.6% EM at 775.8 MB, while adding QwT (72.2% at 795.2 MB) provides no benefit despite higher storage. Our method obtains 74.7% EM at 779.3 MB, with 0.45% memory overhead.

We believe many of these improvements stem from our joint loss formulation, which integrates distillation, feature reconstruction, and supervised objectives. Unlike prior methods such as QwT that depend solely on full-precision teacher guidance, QLA is not limited by the teacher's performance ceiling, allowing it to surpass the full-precision model's accuracy under aggressive quantization.

4.2.2. Encoder-only Models

BERT [5]. To further validate the versatility of our across different modalities, we apply it to BERT-Base on the MRPC [7] dataset. As shown in Table 5, QLA provides consistent gains. For the 4-bit BNB baseline, our approach reaches 87.5% accuracy, matching QwT while using substantially fewer additional parameters (55.9 MB vs. 69.3 MB). For RepQ [21], QLA improves 4-bit performance from 31.6% to 73.0%. Similarly, at 6-bit, proposed QLA boosts RepQ from 84.8% to 87.0% and exceeds QwT.

Table 5. Performance of BERT Base on MRPC [7].

Net.	Method	#Bits	Size (MB)	Top-1
BERT-B	Full-precision	32/32	440	87.8
	BNB [4]	4/16	55.20	80.2
	BNB + QwT [11]	4/16	69.30	87.5
	BNB + QLA	4/16	55.9	**87.5**
	RepQ [21]	4/4	55.20	31.6
	RepQ + QwT [11]	4/4	69.30	56.1
	RepQ + QLA	4/4	55.9	**73.0**
	RepQ [21]	6/6	82.50	84.8
	RepQ + QwT [11]	6/6	96.70	80.9
	RepQ + QLA	6/6	83.2	**85.5**

Table 6. Performance of RoBERTa on the SQuAD v1.1 [31].

Net.	Method	#Bits	Size (MB)	EM
RoBERTa-B	Full-precision	32/32	496.2	82.9
	BNB [4]	4/16	62.40	81.5
	BNB + QwT [11]	4/16	76.60	81.4
	BNB + QLA	4/16	63.1	**83.2**
	RepQ [21]	6/6	93.4	75.3
	RepQ + QwT [11]	6/6	107.6	72.9
	RepQ + QLA	6/6	94.1	**80.4**
	RepQ [21]	8/8	124.4	81.5
	RepQ + QwT [11]	8/8	138.6	81.8
	RepQ + QLA	8/8	125.1	**83.5**

RoBERTa [22]. We extend the QLA framework to RoBERTa-Base and evaluate it on SQuAD v1.1 [31]. As shown in Table 6, for RoBERTa-Base, BNB compresses the model from 496.2 MB to 62.4 MB and achieves 81.5% EM. Adding QwT maintains similar accuracy (81.4%) while increasing the model size to 76.6 MB. In contrast, QLA achieves 83.2% EM with just 0.7 MB overhead. At 6-bit precision, RepQ [21] reaches 75.3% EM with 93.4 MB, and RepQ + QwT reduces performance to 72.9% while increasing memory usage to 107.6 MB. Our approach maintains a comparable size to RepQ (94.1 MB) while improving EM to 80.4%, showing better stability under moderate precision quantization. At 8-bit precision, RepQ achieves 81.5% EM with a 124.4 MB footprint, and RepQ + QwT yields 81.8% EM with a 138.6 MB memory footprint. Our QLA method reaches 83.5% EM while keeping memory footprint close to the baseline (125.1 MB), providing the best accuracy-to-memory balance among all methods.

4.3. Ablation Study

Fig. 2 evaluates the effect of the adapter rank r and LoRA placement across multiple architectures and quantization settings. Increasing the rank provides marginal accuracy improvements, with performance saturating at small ranks (e.g., $r \leq 4$). This indicates that ranks smaller than 8 may be sufficient in compensating for quantization error.

Moreover, Fig. 2 provides further insight into the relative performance of different adapter placements. Unsurprisingly, placing adapters on both the attention and FFN sublayers generally achieves the best results. Falling closely behind, adapters on attention layers only provide the next best performance, with FFN only adapters trailing behind. With this in mind, for applications with more relaxed memory and latency constraints, utilizing adapters on both attention and FFN layers could prove appealing. In a setting with tighter latency and memory budgets though, our analysis shows that comparable accuracy and be achieved with adapters solely applied to attention blocks.

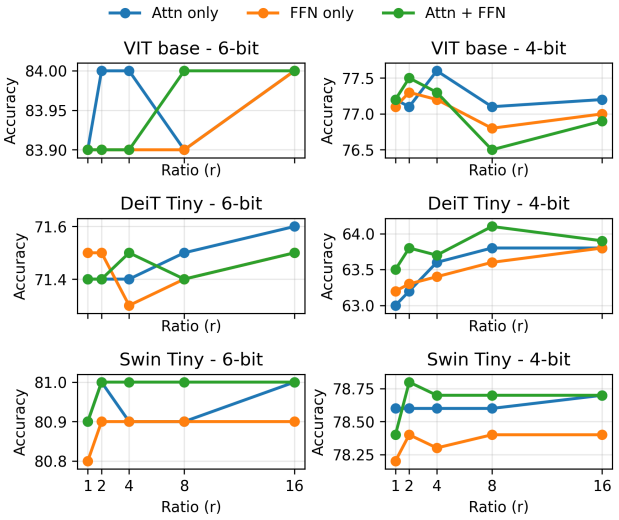

Figure 2. Ablation study on adapter rank r and LoRA placement across three vision architectures under RepQ-ViT quantization settings.

5. Conclusion

In this work, we introduced a compensation framework for transformers that leverages lightweight low-rank adapters to correct quantization-induced errors. These adapters are inserted into the transformer blocks and optimized through supervised learning, knowledge distillation, and feature reconstruction, offsetting quantization error without modifying the backbone or requiring full quantization-aware training. Extensive experiments across vision and language benchmarks demonstrate that the proposed approach consistently improves the accuracy of quantized models while introducing minimal computational and memory overhead, making it a practical solution for deploying highly quantized transformers on resource-constrained hardware.

Acknowledgments

This work is supported by the National Science Foundation (NSF) under grant number 2340249.

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

[2] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 1, 2

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 4

[4] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36: 10088–10115, 2023. 7, 8

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019. 1, 7

[6] Chuntao Ding, Xu Cao, Jianhang Xie, Linlin Fan, Shangguang Wang, and Zhichao Lu. Lora-c: Parameter-efficient fine-tuning of robust cnn for iot devices. *arXiv preprint arXiv:2410.16954*, 2024. 2

[7] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the third international workshop on paraphrasing (IWP2005)*, 2005. 7, 8

[8] Peiyan Dong, Lei Lu, Chao Wu, Cheng Lyu, Geng Yuan, Hao Tang, and Yanzhi Wang. Packqvit: Faster sub-8-bit vision transformers via full and packed quantization on the mobile. *Advances in Neural Information Processing Systems*, 36:9015–9028, 2023. 1, 2

[9] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 4, 5, 6

[10] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019. 1, 2

[11] Minghao Fu, Hao Yu, Jie Shao, Junjie Zhou, Ke Zhu, and Jianxin Wu. Quantization without tears. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4462–4472, 2025. 1, 2, 3, 4, 5, 6, 7, 8

[12] 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. 1, 2

[13] 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. 1

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 2, 3

[16] Mingi Ji, Byeongho Heo, and Sungrae Park. Show, attend and distill: Knowledge distillation via attention-based feature matching. In *Proceedings of the AAAI conference on artificial intelligence*, pages 7945–7952, 2021. 2

[17] Rundong Li, Yan Wang, Feng Liang, Hongwei Qin, Junjie Yan, and Rui Fan. Fully quantized network for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2810–2819, 2019. 6

[18] 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. *arXiv preprint arXiv:2102.05426*, 2021. 1

[19] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. Q-vit: Accurate and fully quantized low-bit vision transformer. *Advances in neural information processing systems*, 35:34451–34463, 2022. 1, 2

[20] Zhengyi Li, Cong Guo, Zhanda Zhu, Yangjie Zhou, Yuxian Qiu, Xiaotian Gao, Jingwen Leng, and Minyi Guo. Efficient activation quantization via adaptive rounding border for post-training quantization. *CoRR*, 2022. 1

[21] Zhikai Li, Junrui Xiao, Lianwei Yang, and Qingyi Gu. Repq-vit: Scale reparameterization for post-training quantization of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17227–17236, 2023. 1, 2, 4, 5, 6, 7, 8

[22] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 8

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 4, 5

[24] Jinendra Malekar and Ramtin Zand. Amdahl's law for LLMs: A throughput-centric analysis of extreme LLM quantization. *Transactions on Machine Learning Research*, 2025. 1

[25] Zhong Meng, Jinyu Li, Yong Zhao, and Yifan Gong. Conditional teacher-student learning. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6445–6449. IEEE, 2019. 2

[26] Jaehyeon Moon, Dohyung Kim, Junyong Cheon, and Bumsub Ham. Instance-aware group quantization for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16132–16141, 2024. 1, 5

[27] 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

[28] 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

[29] NVIDIA Corporation. Nvidia tensorrt: High-performance deep learning inference optimizer and runtime. https://developer.nvidia.com/tensorrt, 2025. Accessed: 2026-03-05. 6

[30] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1, 7

[31] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. 7, 8

[32] Akshat Ramachandran, Souvik Kundu, and Tushar Krishna. Clamp-vit: Contrastive data-free learning for adaptive post-training quantization of vits. In *European Conference on Computer Vision*, pages 307–325. Springer, 2024. 2, 6

[33] Brendan C Reidy, Mohammadreza Mohammadi, Mohammed E Elbtity, and Ramtin Zand. Efficient deployment of transformer models on edge tpu accelerators: A real system evaluation. In *Architecture and System Support for Transformer Models (ASSYST@ ISCA 2023)*, 2023. 1

[34] Majid Sepahvand, Fardin Abdali-Mohammadi, and Amir Taherkordi. Teacher–student knowledge distillation based on decomposed deep feature representation for intelligent mobile applications. *Expert Systems with Applications*, 202: 117474, 2022. 2

[35] Liangchen Song, Xuan Gong, Helong Zhou, Jiajie Chen, Qian Zhang, David Doermann, and Junsong Yuan. Exploring the knowledge transferred by response-based teacher-student distillation. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 2704–2713, 2023. 2

[36] Ningyuan Tang, Minghao Fu, Hao Yu, and Jianxin Wu. Qwt-v2: Practical, effective and efficient post-training quantization. *arXiv preprint arXiv:2505.20932*, 2025. 2, 5

[37] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 4, 5

[38] Guoliang Yang, Shuaiying Yu, Yangyang Sheng, and Hao Yang. Attention and feature transfer based knowledge distillation. *Scientific Reports*, 13(1):18369, 2023. 2

[39] Yang Yang, Wen Wang, Liang Peng, Chaotian Song, Yao Chen, Hengjia Li, Xiaolong Yang, Qinglin Lu, Deng Cai, Boxi Wu, et al. Lora-composer: Leveraging low-rank adaptation for multi-concept customization in training-free diffusion models. *arXiv preprint arXiv:2403.11627*, 2024. 2

[40] 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. Springer, 2022. 1, 2

[41] Xinyi Zeng, Zhanlin Ji, Haiyang Zhang, Rui Chen, Qinping Liao, Jingkun Wang, Tao Lyu, and Li Zhao. Dsp-kd: dual-stage progressive knowledge distillation for skin disease classification. *Bioengineering*, 11(1):70, 2024. 2

[42] Yitao Zhu, Zhenrong Shen, Zihao Zhao, Sheng Wang, Xin Wang, Xiangyu Zhao, Dinggang Shen, and Qian Wang. Melo: Low-rank adaptation is better than fine-tuning for medical image diagnosis. In *2024 IEEE International Symposium on Biomedical Imaging (ISBI)*, pages 1–5. IEEE, 2024. 2