LARQ: Lightweight Adapter-based Recovery from Quantization in Large Language Models

Mohammadreza Mohammadi Matthew Grenier Jinendra Malekar Ramtin Zand
Department of Computer Science and Engineering, University of South Carolina, Columbia, SC
{MOHAMMM, MGRENIER, JMALEKAR}@email.sc.edu ,{RAMTIN}@cse.sc.edu

***Abstract*—Post-training quantization (PTQ) is a widely used compression technique for reducing the memory footprint and computational cost of large language models (LLMs) during inference on resource-constrained hardware. However, aggressive quantization can introduce non-trivial accuracy degradation. Existing recovery methods either require costly retraining, depend on labeled data, or incur significant parameter overhead. We propose a Lightweight Adapter-based Recovery method for Quantization (LARQ) in LLMs. LARQ is an architecture-agnostic framework that restores quantization accuracy without task-specific supervision and is applicable across different quantization precisions. LARQ augments a frozen quantized LLM with small parallel low-rank adapters inserted into both attention and feed-forward sub-layers, achieving over $60\times$ lower parameter overhead than prior compensation approaches. The adapters are trained using knowledge distillation and intermediate feature matching from a full-precision teacher, eliminating the need for labeled data. We evaluate LARQ across nine LLM architectures from the LLaMA, Mistral, and Qwen2 families under bitsandbytes NF4 and GPTQ 4-bit quantization. LARQ consistently improves perplexity across all 18 configurations and recovers accuracy in nearly all cases with negligible overhead ($< 1\%$ of model parameters).**

I. Introduction

Large Language Models (LLMs) have transformed the landscape of natural language processing, demonstrating remarkable capabilities across a wide range of tasks, including question answering [2], commonsense reasoning [18], and code generation [17]. Models such as LLaMA [18], Mistral [12], and Qwen [1] have achieved state-of-the-art performance across diverse benchmarks, yet their deployment on resource-constrained hardware remains a significant challenge. Modern LLMs typically contain billions of parameters, requiring tens of gigabytes of memory for inference and substantial computational resources that exceed the capacity of most edge and consumer-grade hardware [3], [5].

Quantization has emerged as one of the most practical approaches to reducing the memory footprint and inference cost of LLMs. By representing model weights and activations in low-bit formats, typically 4-bit or 8-bit integers instead of 16-bit or 32-bit floating-point values, quantization can reduce memory consumption by up to $8\times$ while preserving much of the model's capability [8]. Among quantization strategies, Post-Training Quantization (PTQ) is particularly attractive for LLMs because it requires no retraining of the original model, meaning large models with billions of parameters can be quantized in a time-efficient manner.

Despite these advances, PTQ for LLMs is not without limitations. Aggressive quantization, particularly at 4-bit precision, introduces quantization error that degrades model accuracy, with larger degradations observed in smaller models [4], [5]. Quantization-Aware Training (QAT) can mitigate this accuracy loss but is computationally prohibitive for billion-parameter models [13]. Parameter-efficient fine-tuning methods such as LoRA [11] and QLoRA [4] offer a middle ground by training small adapter modules on top of quantized models, but they require labeled task-specific data and are fundamentally designed for fine-tuning rather than quantization error recovery.

Recent methods, principally Quantization without Tears (QwT) [6], have worked to address this issue by introducing compensation layers to each transformer block. Despite effectively reducing quantization error, each compensation layer consist of a full $d \times d$ linear layer, resulting in parameter overhead that scales quadratically with hidden dimension and becomes non-negligible for large models. Furthermore, QwT relies solely on closed-form calibration without leveraging teacher guidance, limiting its ability to recover task-relevant output distributions.

To address this gap, we propose *LARQ* (*Lightweight Adapter Recovery from Quantization*), a lightweight and architecture-agnostic framework to recover quantization-induced accuracy degradation in LLMs. LARQ augments a quantized LLM by inserting small, parallel, low-rank adapter modules of rank $r \ll d$ at each layer's attention and feed-forward blocks, reducing the per-block parameter cost from $\mathcal{O}(d^2)$ to $\mathcal{O}(rd)$. In practice, this results in a reduction of over $100\times$ for typical LLM hidden dimensions (e.g., $r = 8$, $d = 4096$). These adapters are trained using a combination of knowledge distillation (KD) and intermediate feature matching (FM), using the full-precision model as a teacher to guide the quantized student toward recovering its original output distribution. Since LARQ requires no supervision and operates on top of a quantized model's frozen weights, it is applicable to any PTQ-quantized LLM regardless of architecture or quantization scheme. Optionally, when task-specific labeled data is available, LARQ's training objective can be extended with a supervised loss, providing further capacity for error recovery.

We conduct a comprehensive evaluation across nine LLM architectures spanning the LLaMA, Mistral, and Qwen2 families under two quantization schemes (bitsandbytes [4] NF4 and GPTQ [5] 4-bit) and two benchmarks (WikiText-2 [14] and MMLU [10]). Across these settings, LARQ consistently

improves performance over quantized baselines, achieving recovery in 17 out of 18 MMLU configurations. The results further reveal that model size plays a key role in commonsense reasoning recovery, with larger models exhibiting more stable improvements under quantization.

II. Approach

A. 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 quantization errors accumulate across layers, leading to noticeable degradation in overall model performance.

B. SOTA Limitations

Recently, QwT [6] introduced an approach to compensate for the discrepancy between quantized and full-precision models without retraining the entire network. QwT augments the quantized model with compensation modules that operate in parallel with existing blocks, where each module is implemented as a linear layer receiving the same input and producing an output of identical dimensionality. The parameters of these 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, $M_{\text{FP}}(\cdot)$ and $M_{\text{int}}(\cdot)$ denote the functions implemented by the floating-point and quantized models, respectively, and the norm is computed at a chosen alignment point.

Although conceptually simple, QwT performs block-level compensation using full $d \times d$ transformation matrices for each transformer block, introducing substantial parameter overhead that scales quadratically with hidden dimension. Moreover, the parameters of these matrices are estimated using linear regression, requiring solving large systems whose size grows quickly with the number of calibration samples, leading to significant memory and computational costs. Finally, QwT relies solely on closed-form calibration without leveraging teacher guidance, limiting its ability to recover task-relevant output distributions. In contrast, LARQ employs low-rank adapter modules embedded within transformer sub-layers, avoiding the scalability issues associated with large full-rank weight matrices while substantially reducing parameter overhead.

C. Implementation

Considering the limitations of existing methods, LARQ introduces lightweight low-rank adapter modules within each transformer sub-layer. As shown in Figure 1a, each transformer block is augmented with two adapters: one placed in parallel to the attention sub-layer and one placed in parallel to the feed-forward network (FFN). These modules learn corrections for quantization-induced error while keeping the original block structure and data flow intact.

Figure 1b illustrates how the adapters interact with the transformer pathways. Two LoRA blocks [11] are inserted in parallel within each decoder layer: one receives the block input and operates alongside the self-attention sub-layer, and one receives the post-attention hidden states and operates alongside the FFN sub-layer. In both cases, the adapter output is added directly to the corresponding sub-layer output before the residual connection and layer normalization, preserving the original data flow.

The LoRA blocks are not merged with the internal modules of each sub-layer and do not modify any weights of the quantized backbone. They operate purely as parallel correction paths that receive the same input and produce outputs with the same dimensionality as the original sub-layers, making LARQ fully model-agnostic and applicable to any decoder-only architecture without architectural modifications.

To formalize the adapter operation, we describe its effect at the level of a single transformer sub-layer. 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 $\mathcal{F}(X) \in \mathbb{R}^{d \times \ell}$. Each adapter receives the same input and computes a low-rank correction:

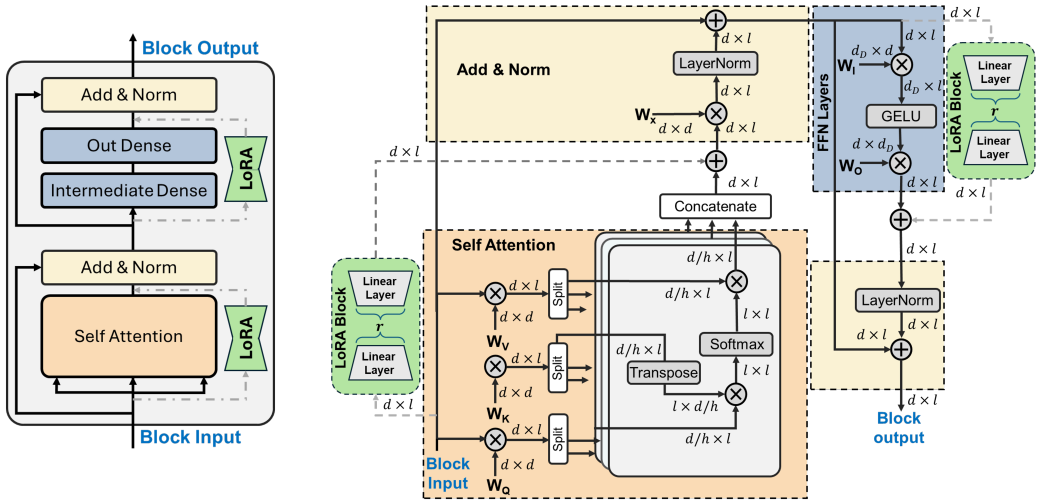
$$\Delta(X) = \alpha BAX, \tag{5}$$

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 remains frozen and only the adapter parameters are optimized. Unlike the original LoRA [11], which is designed for task-specific fine-tuning, LARQ treats the quantized model as a fixed backbone and trains the adapters to compensate for quantization-induced error.

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, and since two adapters are inserted per decoder layer,



(a) (b)

Fig. 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.

the total additional parameters per layer is $4dr$. 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 achieving substantial accuracy recovery. This overhead is vastly smaller than the full-rank compensation matrices used by QwT [6], which require d^2 additional parameters per block.

Loss Function. In its primary form, LARQ trains adapters using a combination of knowledge distillation and feature matching, requiring no labeled data:

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

The two loss components are defined as:

$$\mathcal{L}_{\text{KD}} = T^2 \cdot \text{KL}\left(\text{softmax}\left(\frac{z_t}{T}\right) \,\middle\|\, \text{softmax}\left(\frac{z_s}{T}\right)\right), \tag{8}$$

$$\mathcal{L}_{\text{FM}} = \frac{1}{L} \sum_{\ell=1}^{L} \left\| f_t^{(\ell)} - f_s^{(\ell)} \right\|_2^2, \tag{9}$$

where z_t and z_s denote the teacher and student logits, T is the distillation temperature, and $f_t^{(\ell)}$ and $f_s^{(\ell)}$ are the intermediate feature maps from layer ℓ of the teacher and student respectively. $\mathcal{L}_{\text{KD}}$ aligns the output distribution of the quantized model with its full-precision teacher, while $\mathcal{L}_{\text{FM}}$ promotes local feature reconstruction between matched layers.

When task-specific labeled data is available and the model is aligned with the target task (e.g., through prior fine-tuning or compatibility with the task formulation), the objective can optionally be extended with a supervised term:

$$\mathcal{L} = \lambda_{\text{sup}} \mathcal{L}_{\text{sup}} + \lambda_{\text{kd}} \mathcal{L}_{\text{KD}} + \lambda_{\text{feat}} \mathcal{L}_{\text{FM}}, \tag{10}$$

where $\mathcal{L}_{\text{sup}}$ denotes a task-specific supervised loss computed over labeled data. Including the supervised term allows the adapter to incorporate task-specific signals beyond those provided by the full-precision teacher, and can enable the quantized model to surpass FP32 performance on the target task, as observed in our experiments.

III. Experiments

A. Experimental Setup

Models. We evaluate LARQ across nine decoder-only LLM architectures spanning three families: LLaMA [9], [18] (Llama-2-7B, Llama-3.2-1B, Llama-3.2-3B, Meta-Llama-3-8B), Mistral [12] (Mistral-7B-v0.1, Mistral-7B-v0.3), and Qwen2 [1] (Qwen2-0.5B, Qwen2-1.5B, Qwen2-7B). This selection covers a wide range of model sizes (0.5B to 8B parameters) and architectures, enabling a comprehensive assessment of LARQ across diverse LLM families.

Quantization Schemes. We evaluate under two widely adopted 4-bit PTQ schemes: bitsandbytes (BnB) NF4 with double quantization [4] and GPTQ [5] with group size 128. Both schemes quantize all linear layers in the backbone while keeping the adapter parameters in full precision.

Adapter Configuration. Adapter parameters are initialized such that $B = 0$ and A is randomly initialized, ensuring that adapters contribute zero correction at the start of training. Additionally, all experiments use a fixed adapter rank of $r = 8$.

Training. Adapters are trained for one epoch using the AdamW optimizer with learning rate 1×10^{-3}. The knowledge distillation temperature is set to $T = 2$, with loss weights $\lambda_{\text{kd}} = 1.0$ and $\lambda_{\text{feat}} = 1 \times 10^{-3}$. The full-precision teacher model is loaded in bf16 on CPU during training to preserve GPU memory for the quantized student. Training uses a block size of 256 tokens with batch size 1.

Evaluation. We evaluate on two complementary benchmarks. WikiText-2 [14] perplexity measures general language modeling quality using standard strided evaluation. MMLU

[10] measures knowledge retrieval and reasoning accuracy using 5-shot log-likelihood scoring following the lm-evaluation-harness protocol [7]. For each model and quantization scheme, we report results for the full-precision teacher (FP), the quantized baseline without adapters (Q), and the quantized model with LARQ adapters (Q+LARQ).

B. WikiText-2 Perplexity

Table I reports WikiText-2 perplexity results across all nine architectures under both quantization schemes. For each model, BnB and GPTQ rows share the same full-precision baseline, as the teacher remains unchanged. Across all configurations, LARQ consistently reduces perplexity compared to the corresponding 4-bit quantized models. For instance, for Llama-3-8B, perplexity is reduced from 6.66 to 6.52 under BnB and from 6.69 to 6.61 under GPTQ. Similar improvements are observed for smaller models such as Llama-3.2-1B, where perplexity decreases from 11.25 to 10.70 (BnB) and from 11.18 to 10.94 (GPTQ). These results indicate that the benefits of LARQ are consistent across different model scales and are not limited to large architectures.

Importantly, these gains are achieved with negligible memory overhead relative to the quantized backbone. As shown in Table I, incorporating LARQ increases the model size by only $\sim$0.01–0.02 GB across all architectures, corresponding to less than 0.5% additional memory even for smaller models. For example, in Llama-3-8B under GPTQ, the model size increases marginally from 5.73 GB to 5.75 GB while achieving a clear reduction in perplexity. Similar trends are observed across all models, including compact architectures such as Qwen-2-0.5B, where the increase in memory footprint is below the reporting precision in GB (i.e., < 0.01 GB) and thus appears unchanged in the table.

Comparison with QwT. A direct comparison with QwT on the same Meta-Llama-3-8B model under GPTQ 4-bit quantization on WikiText-2 further highlights the effectiveness of our approach (see Table II). While QwT improves the GPTQ baseline perplexity from 6.65 to 6.63, our method achieves a larger reduction to 6.61, indicating stronger compensation for quantization-induced errors. Moreover, this improvement is achieved with substantially lower memory cost: QwT introduces approximately 536.9M additional parameters (about 1.07 GB in FP16), whereas our method adds only 4.46M parameters, corresponding to roughly 0.02 GB in FP32, over 60$\times$ smaller in memory footprint.

C. MMLU Evaluation

We further evaluate the proposed method on the MMLU (Massive Multitask Language Understanding) benchmark using the standard 5-shot setting. MMLU is a widely used evaluation suite consisting of multiple-choice questions spanning diverse domains, including STEM, humanities, social sciences, and professional topics. Performance is measured using top-1 accuracy, where the model must select the correct answer among four candidates. This benchmark provides a

TABLE I: WikiText-2 perplexity results across nine LLM architectures under BnB NF4 and GPTQ 4-bit quantization. LARQ uses rank $r = 8$ adapters applied in parallel to both the attention and FFN sub-layers of every decoder layer. Size (GB) denotes model memory footprint; LARQ rows include fp32 adapter overhead ($\leq$0.02 GB across all models). Lower perplexity is better. Best quantized result per model in **bold**.

Model	Method	#Bits	Size (GB)	PPL ↓
Llama-3 8B	Full-precision	16	16.06	6.23
	Quantized (BnB)	4	5.59	6.66
	BnB + LARQ	4	5.61	**6.52**
	Quantized (GPTQ)	4	5.73	6.65
	GPTQ + LARQ	4	5.75	**6.61**
Llama-3.2 1B	Full-precision	16	2.31	10.16
	Quantized (BnB)	4	1.01	11.25
	BnB + LARQ	4	1.02	**10.70**
	Quantized (GPTQ)	4	1.03	11.18
	GPTQ + LARQ	4	1.04	**10.94**
Llama-3.2 3B	Full-precision	16	5.98	8.09
	Quantized (BnB)	4	2.20	8.59
	BnB + LARQ	4	2.21	**8.33**
	Quantized (GPTQ)	4	2.26	8.55
	GPTQ + LARQ	4	2.27	**8.42**
Llama-2 7B	Full-precision	16	13.48	5.72
	Quantized (BnB)	4	3.76	5.87
	BnB + LARQ	4	3.78	**5.81**
	Quantized (GPTQ)	4	3.89	5.87
	GPTQ + LARQ	4	3.91	**5.83**
Mistral 7B-v0.1	Full-precision	16	13.97	5.46
	Quantized (BnB)	4	4.01	5.59
	BnB + LARQ	4	4.03	**5.53**
	Quantized (GPTQ)	4	4.16	5.62
	GPTQ + LARQ	4	4.17	**5.57**
Mistral 7B-v0.3	Full-precision	16	14.00	5.55
	Quantized (BnB)	4	4.03	5.68
	BnB + LARQ	4	4.04	**5.63**
	Quantized (GPTQ)	4	4.17	5.72
	GPTQ + LARQ	4	4.19	**5.65**
Qwen-2 0.5B	Full-precision	16	0.94	14.06
	Quantized (BnB)	4	0.45	15.19
	BnB + LARQ	4	0.45	**14.61**
	Quantized (GPTQ)	4	0.46	15.23
	GPTQ + LARQ	4	0.46	**14.77**
Qwen-2 1.5B	Full-precision	16	2.88	9.91
	Quantized (BnB)	4	1.12	10.39
	BnB + LARQ	4	1.13	**10.18**
	Quantized (GPTQ)	4	1.15	10.33
	GPTQ + LARQ	4	1.16	**10.22**
Qwen-2 7B	Full-precision	16	14.19	7.35
	Quantized (BnB)	4	5.44	7.54
	BnB + LARQ	4	5.46	**7.46**
	Quantized (GPTQ)	4	5.58	7.54
	GPTQ + LARQ	4	5.59	**7.52**

TABLE II: Comparison with QwT on Meta-Llama-3-8B under GPTQ 4-bit quantization on WikiText-2.

Method	Size (GB)	Overhead (M)	PPL ↓
Full-precision	16.06	–	6.24
GPTQ	5.73	–	6.65
GPTQ + QwT	6.80	536.9	6.63
GPTQ + LARQ	5.75	4.46	**6.61**

comprehensive assessment of a model's general reasoning and knowledge capabilities across a broad range of tasks.

Table III summarizes the results across nine architectures under both quantization schemes. In almost all cases, augmenting the quantized models with LARQ leads to higher accuracy. For example, on Llama-3-8B, accuracy increases from 62.46% to 63.21% under GPTQ, while under BnB it improves from 63.59% to 63.98%. Notably, the gains are often more pronounced for smaller models: Qwen-2-0.5B improves by over 3% absolute accuracy under GPTQ (39.49% to 42.74%), highlighting the effectiveness of the method in low-capacity regimes. Across all configurations, only a single case (Mistral-7B-v0.1 under BnB) shows no measurable change, while all others benefit from the proposed adaptation.

Beyond individual examples, a consistent trend emerges across both quantization backends: adapted models effectively reduce quantization-induced error. This behavior is observed across both large and compact architectures, suggesting that the proposed method is largely insensitive to model scale and backend choice.

D. Compatibility with Supervised Learning

While LARQ is designed to operate without labeled data, its formulation can be extended to supervised settings when task-specific annotations are available. To demonstrate this flexibility, we evaluate LARQ on SQuAD v1.1 [16] using GPT-2 [15] small and large under BnB NF4 4-bit quantization, augmenting the KD and FM objectives with a supervised next-token prediction loss over labeled question-answer pairs.

In earlier sections, the evaluated models were not fine-tuned on the corresponding downstream datasets, and no task-specific supervision was used, restricting LARQ to unsupervised objectives (KD and FM). In contrast, the GPT-2 models used here are already fine-tuned on SQuAD-style question answering tasks, enabling the use of supervised loss without introducing a task mismatch. This allows us to evaluate LARQ in a fully supervised regime. This provides a complementary setting to our main experiments and allows us to evaluate the maximum performance achievable by LARQ under task-specific supervision.

Table IV reports Exact Match (EM) scores for the full-precision baseline, the quantized model, and LARQ with the combined supervised objective. LARQ consistently improves over the quantized baseline in both model sizes. For GPT-2 Small, LARQ achieves 60.2 EM, surpassing the full-precision baseline of 59.1 by 1.1% despite operating at 4-bit precision. A similar trend is observed for GPT-2 Large, where LARQ

TABLE III: MMLU 5-shot accuracy (%) across nine LLM architectures under BnB NF4 and GPTQ 4-bit quantization. LARQ uses rank $r = 8$ adapters applied in parallel to both the attention and FFN sub-layers of every decoder layer. Size (GB) denotes model memory footprint in gigabytes (10^9 bytes); LARQ rows include fp32 adapter overhead ($\leq$0.02 GB across all models). Higher accuracy is better. Best quantized result per model in **bold**.

Model	Method	#Bits	Size (GB)	Acc (%) ↑
Llama-3 8B	Full-precision	16	16.06	65.55
	Quantized (BnB)	4	5.59	63.59
	BnB + LARQ	4	5.61	**63.98**
	Quantized (GPTQ)	4	5.73	62.46
	GPTQ + LARQ	4	5.75	**63.21**
Llama-3.2 1B	Full-precision	16	2.31	31.55
	Quantized (BnB)	4	1.01	28.64
	BnB + LARQ	4	1.02	**29.37**
	Quantized (GPTQ)	4	1.03	29.93
	GPTQ + LARQ	4	1.04	**30.27**
Llama-3.2 3B	Full-precision	16	5.98	56.42
	Quantized (BnB)	4	2.20	54.78
	BnB + LARQ	4	2.21	**55.26**
	Quantized (GPTQ)	4	2.26	53.58
	GPTQ + LARQ	4	2.27	**54.22**
Llama-2 7B	Full-precision	16	13.48	45.81
	Quantized (BnB)	4	3.76	44.51
	BnB + LARQ	4	3.78	**44.92**
	Quantized (GPTQ)	4	3.89	43.39
	GPTQ + LARQ	4	3.91	**45.19**
Mistral 7B-v0.1	Full-precision	16	13.97	62.56
	Quantized (BnB)	4	4.01	61.28
	BnB + LARQ	4	4.03	61.28
	Quantized (GPTQ)	4	4.16	61.08
	GPTQ + LARQ	4	4.17	**61.33**
Mistral 7B-v0.3	Full-precision	16	14.00	62.39
	Quantized (BnB)	4	4.03	60.81
	BnB + LARQ	4	4.04	**61.46**
	Quantized (GPTQ)	4	4.17	60.95
	GPTQ + LARQ	4	4.19	**61.12**
Qwen-2 0.5B	Full-precision	16	0.94	44.22
	Quantized (BnB)	4	0.45	41.75
	BnB + LARQ	4	0.45	**42.91**
	Quantized (GPTQ)	4	0.46	39.49
	GPTQ + LARQ	4	0.46	**42.74**
Qwen-2 1.5B	Full-precision	16	2.88	55.83
	Quantized (BnB)	4	1.12	54.95
	BnB + LARQ	4	1.13	**55.05**
	Quantized (GPTQ)	4	1.15	53.37
	GPTQ + LARQ	4	1.16	**54.18**
Qwen-2 7B	Full-precision	16	14.19	70.47
	Quantized (BnB)	4	5.44	69.88
	BnB + LARQ	4	5.46	**70.15**
	Quantized (GPTQ)	4	5.58	68.42
	GPTQ + LARQ	4	5.59	**68.82**

TABLE IV: Performance of GPT-2 models on the SQuAD v1.1 [16] dataset under BnB NF4 4-bit quantization. We report model size in gigabytes and Exact Match (EM) scores for the full-precision baseline, the quantized baseline, and LARQ with the full supervised objective including next-token prediction loss. LoRA adapters of rank $r = 8$ are applied in parallel to both the attention and FFN sub-layers.

Model	Method	#Bits	Size (MB)	EM ↑
GPT-2 Small	Full-precision	32	497.8	59.1
	BnB [4]	4	62.6	56.0
	BnB + LARQ	4	63.3	**60.2**
GPT-2 Large	Full-precision	32	3096	71.8
	BnB [4]	4	389.1	74.1
	BnB + LARQ	4	392.3	**74.7**

reaches 74.7 EM compared to the full-precision score of 71.8, yielding a gain of 2.9%.

These results highlight two key properties of LARQ in the supervised setting. First, the adapter modules are expressive enough to incorporate task-specific signals on top of a frozen quantized backbone. Second, because the supervised loss is not constrained by the full-precision teacher, the combined objective enables the quantized model to exceed FP32 accuracy, indicating that LARQ does not impose a strict upper bound tied to the teacher model.

It is important to note that this supervised setup is not directly comparable to the unsupervised experiments presented earlier, as it leverages labeled data. Instead, it provides a complementary evaluation of LARQ under task-specific supervision, demonstrating that the same framework can be effectively applied in both unsupervised and supervised scenarios without any architectural modifications.

IV. Limitations

While LARQ demonstrates consistent recovery across multiple LLM families and quantization backends, several limitations remain. First, the current formulation applies identical adapter ranks across all transformer layers and sub-layers, despite quantization sensitivity often varying significantly across attention and FFN blocks. More adaptive layer-aware rank allocation strategies may further improve recovery efficiency. Second, LARQ relies on mixed-precision computation, where the backbone remains quantized while the adapters operate in full precision. Current hardware accelerators and inference runtimes often provide limited support for efficient mixed-precision execution, potentially reducing some practical deployment benefits. Finally, compared to quantization-aware training, LARQ requires substantially less training computation because only the small low-rank adapters are updated while the quantized backbone remains frozen. However, for extremely large-scale models, even training these lightweight adapters for a single epoch can still be time-consuming and computationally expensive.

V. Societal Impact

From a societal perspective, this work aims to improve the efficiency and accessibility of LLM deployment on resource-constrained hardware. However, improving deployment efficiency may also indirectly increase access to systems capable of generating inaccurate, biased, or misleading content. LARQ itself does not introduce new generative capabilities or alter model alignment behavior beyond recovering quantization-induced degradation, and therefore inherits the broader societal risks associated with pretrained language models.

VI. Conclusion

We presented LARQ, a lightweight and architecture-agnostic framework for recovering quantization-induced accuracy degradation in LLMs. By augmenting frozen 4-bit quantized LLMs with small parallel low-rank adapters, LARQ enables effective recovery using knowledge distillation and intermediate feature matching, without requiring any labeled data and with less than 1% parameter overhead. Extensive evaluation across nine architectures, two quantization schemes, and multiple benchmarks demonstrates that LARQ consistently improves both perplexity and downstream accuracy. In particular, LARQ improves WikiText-2 perplexity across all evaluated configurations and recovers MMLU accuracy in nearly all cases, while achieving a significantly better accuracy–efficiency trade-off compared to prior approaches such as QwT, with over $60\times$ lower parameter overhead. Finally, we show that LARQ can be extended to supervised settings, enabling task-aware adaptation under quantization without architectural changes. Overall, LARQ provides a unified and efficient solution for improving the accuracy of quantized LLMs, making it well-suited for deployment in resource-constrained environments.

VII. Code Availability

The implementation of LARQ is available at: https://github.com/iCAS-Lab/LARQ.

Acknowledgements

This work is supported by the National Science Foundation (NSF) under grant numbers 2340249 and 2409697.

References

[1] J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[3] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Llm.int8(): 8-bit matrix multiplication for transformers at scale," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 318–30 332, 2022.

[4] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," *Advances in neural information processing systems*, vol. 36, pp. 10 088–10 115, 2023.

[5] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," *arXiv preprint arXiv:2210.17323*, 2022.

[6] M. Fu, H. Yu, J. Shao, J. Zhou, K. Zhu, and J. Wu, "Quantization without tears," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 4462–4472.

[7] L. Gao, J. Tow, B. Abbasi, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu, A. Le Noac'h, H. Li, K. McDonell, N. Muennighoff, C. Ociepa, J. Phang, L. Reynolds, H. Schoelkopf, A. Skowron, L. Sutawika, E. Tang, A. Thite, B. Wang, K. Wang, and A. Zou, "A framework for few-shot language model evaluation," 2023. [Online]. Available: https://zenodo.org/records/10256836

[8] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-power computer vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[9] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.

[10] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," *arXiv preprint arXiv:2009.03300*, 2021.

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[12] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[13] Z. Liu, B. Oguz, C. Zhao, E. Chang, P. Stock, Y. Mehdad, Y. Shi, R. Krishnamoorthi, and V. Chandra, "Llm-qat: Data-free quantization aware training for large language models," *arXiv preprint arXiv:2305.17888*, 2023.

[14] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.

[15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[16] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[17] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, T. Alon, H. Bisber, G. Lima, C. Szegedy, G. Synnaeve *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.

[18] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.