LoTA-QAF: Lossless Ternary Adaptation for Quantization-Aware Fine-Tuning

Junyu Chen[1,2,3], **Junzhuo Li**[3], **Zhen Peng**[4], **Wenjie Wang**[1,2], **Yuxiang Ren**[5], **Long Shi**[1,2,*], **Xuming Hu**[3,*]

[1] Southwestern University of Finance and Economics
[2] Artificial Intelligence and Digital Finance Key Laboratory of Sichuan Province
[3] The Hong Kong University of Science and Technology (Guangzhou)
[4] Sun Yat-sen University [5] Nanjing University

223081200039@smail.swufe.edu.cn
shilong@swufe.edu.cn xuminghu@hkust-gz.edu.cn

Abstract

Quantization and fine-tuning are crucial for deploying large language models (LLMs) on resource-constrained edge devices. However, fine-tuning quantized models presents significant challenges, primarily stemming from: First, the mismatch in data types between the low-precision quantized weights (e.g., 4-bit) and the high-precision adaptation weights (e.g., 16-bit). This mismatch limits the computational efficiency advantage offered by quantized weights during inference. Second, potential accuracy degradation when merging these high-precision adaptation weights into the low-precision quantized weights, as the adaptation weights often necessitate approximation or truncation. Third, as far as we know, no existing methods support the lossless merging of adaptation while adjusting all quantized weights. To address these challenges, we introduce lossless ternary adaptation for quantization-aware fine-tuning (LoTA-QAF). This is a novel fine-tuning method specifically designed for quantized LLMs, enabling the lossless merging of ternary adaptation weights into quantized weights and the adjustment of all quantized weights. LoTA-QAF operates through a combination of: i) A custom-designed ternary adaptation (TA) that aligns ternary weights with the quantization grid and uses these ternary weights to adjust quantized weights. ii) A TA-based mechanism that enables the lossless merging of adaptation weights. iii) Ternary signed gradient descent (t-SignSGD) for updating the TA weights. We apply LoTA-QAF to Llama-3.1/3.3 and Qwen-2.5 model families and validate its effectiveness on several downstream tasks. On the MMLU benchmark, our method effectively recovers performance for quantized models, surpassing 16-bit LoRA by up to 5.14%. For task-specific fine-tuning, 16-bit LoRA achieves superior results, but LoTA-QAF still outperforms other methods. Code is available in github.com/KingdalfGoodman/LoTA-QAF.

1 Introduction

Large language models (LLMs) have showcased exceptional proficiency in natural language processing, driving advancements in applications such as complex reasoning (Hadi et al., 2023), code generation (Jiang et al., 2024), and conversational AI systems (McTear and Ashurkina, 2024). However, their immense computational costs present significant challenges, particularly when deploying

* Corresponding authors.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

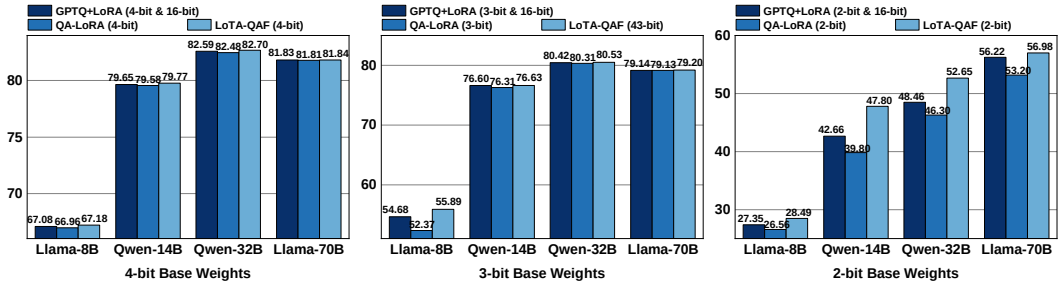

Figure 1: Comparison of 5-shot MMLU accuracy (%), fine-tuned on the Alpaca dataset. GPTQ+LoRA (4-bit & 16-bit) uses 4-bit base and 16-bit adapter weights. QA-LoRA and LoTA-QAF operate at 4-bit after merging adapters into quantized weights. Details in Table 1.

these models on resource-constrained edge devices (Wan et al., 2023). To overcome this hurdle, quantization techniques have emerged as a critical approach to model compression, effectively reducing memory demands and computational complexity by quantizing model weights to lower bit precision. Furthermore, deploying on edge devices not only necessitates model compression but also requires the models to handle specialized tasks, which require the integration of domain-specific knowledge. Consequently, there is growing attention on fine-tuning LLMs to align with specific tasks under quantization constraints. One prominent approach designed for this scenario is QLoRA (Dettmers et al., 2023), which integrates low-rank adaptation with quantization. It quantizes the pretrained weights to 4-bit precision, significantly reducing the static memory footprint. Subsequently, 16-bit precision LoRA adapters are trained on top of these quantized weights, achieving efficient and low-cost fine-tuning.

Nevertheless, directly applying adapters to fine-tuning quantized models introduces notable challenges that can hinder efficiency and performance. The first challenge arises during inference, where the interaction between low-precision weights (e.g., 4-bit) and high-precision adapter weights (e.g., 16-bit) can impair computational efficiency. This mismatch introduces computational overhead that can diminish the inference speedups from quantization (Jeon et al., 2024). The second challenge occurs when merging high-precision LoRA adapters into the low-precision quantized weights. During this process, the adapters must inevitably be quantized or truncated to the lower bit width (Bondarenko et al., 2024; Guo et al., 2024). This reintroduction of quantization error at the adapter level, despite the adapters' aim to correct model quantization errors, leads to fine-tuning accuracy degradation. The third challenge is that existing lossless adapter merging schemes (Xu et al., 2023) only allow for adjustment of quantization parameters, rather than direct modification of the quantized weights. This significantly limits the adapter's capacity for effective fine-tuning.

These challenges highlight the development of fine-tuning methods integrated within a quantization framework, namely quantization-aware fine-tuning (QAF). In this paper, we present LoTA-QAF, a lossless ternary adaptation fine-tuning method for quantized pretrained language models. Specifically, LoTA-QAF operates through a combination of three key components: i) A custom-designed ternary adaptation (TA) that includes trainable ternary adapters, which form an auxiliary matrix to adjust the quantized weights within the target quantization grid, instead of merely adjusting quantization parameters. ii) A TA-based lossless merging mechanism that utilizes the auxiliary matrix to generate a ternary matrix and an offset matrix, subsequently updating the quantized weights and zero factors. This mechanism preserves the low-bit computational efficiency and avoids fine-tuning accuracy degradation. iii) Ternary signed gradient descent (t-SignSGD) is employed to update the trainable ternary adapters during fine-tuning.

We apply LoTA-QAF to the Llama-3.1/3.3 and Qwen-2.5 model families, validating its effectiveness in performance recovery and task-specific alignment for quantized models. Figure 1 illustrates the MMLU benchmark results, demonstrating that LoTA-QAF consistently outperforms other methods, especially at 2-bit. It should be noted that LoRA employs 16-bit adapters (inference with 4-bit model weights and 16-bit adapters), which cannot be losslessly merged into quantized weights. In contrast, both LoTA-QAF and QA-LoRA can achieve lossless merging (inference with only 4-bit model weights). However, QA-LoRA only indirectly compensates for quantization errors by adjusting zero

factors via its adapter, lacking the capability to directly modify the quantized weights. Building on this, our main contributions include:

- We present LoTA-QAF, which adjusts quantized weights within the quantization grid and facilitating the lossless merging of ternary adapters into the quantized weights. As a result, LoTA-QAF preserves low-bit computational efficiency and avoids accuracy degradation.
- We introduce ternary signed gradient descent (t-SignSGD), a novel optimizer for ternary adapters. It leverages sign-based updates with dynamic percentile-based thresholding to selectively adjust ternary adapter weights, thereby effectively optimizing these highly constrained parameters.
- LoTA-QAF demonstrates effectiveness in two key quantization-aware fine-tuning scenarios, achieving strong performance while maintaining the inference efficiency of quantized models. For performance recovery, LoTA-QAF shows up to 5.14% improvement over LoRA on Qwen 2.5 14B 2-bit. In task-specific tuning, while trailing LoRA (16-bit adapters), LoTA-QAF still outperforms other methods. Regarding inference efficiency, LoTA-QAF is 1.7x-2.0x faster than LoRA after adapters are merged into quantized weights.

2 Related Work

Quantization of LLMs. Quantization techniques fundamentally reduce memory requirements and improve computational efficiency by lowering the bit precision of model weights. However, the precision loss inherent in quantization inevitably leads to a decline in the performance of quantized models (Kumar et al., 2024). Consequently, recovering the performance of quantized models is a crucial research topic in the field. Post-training quantization (PTQ) offers a fast compression solution without requiring retraining. One approach, GPTQ (Frantar et al., 2022), employs approximate second-order information during the quantization process, compensating for the quantization errors progressively. In addition, other approaches aim to address outliers, such as AWQ (Lin et al., 2024), which identifies and protects salient weights by scaling channels based on activation statistics. QuaRot (Ashkboos et al., 2024) uses randomized Hadamard transformations to remove outliers from the hidden state. Nevertheless, PTQ cannot adapt models to downstream tasks because it is applied after training.

Quantization-aware training (QAT), which integrates quantization simulation during training (Chen et al., 2024; Liu et al., 2023), not only helps performance recovery but also aligns quantized models with specific tasks. LLM-QAT (Liu et al., 2023) introduces QAT for LLMs using a data-free distillation approach to preserve the original model's output distribution. Efficient-QAT (Chen et al., 2024) proposes a two-stage strategy involving block-wise training of all parameters and end-to-end training of quantization parameters. HALO (Ashkboos et al., 2025) introduces Hadamard rotations during both forward and backward passes to mitigate outliers in low-precision training. However, the computational cost of QAT remains a significant challenge.

Quantization-Aware Fine-Tuning of LLMs. To maintain the low computational cost of quantization, while achieving performance recovery and alignment with specific tasks through fine-tuning, Bondarenko et al. (2024); Dettmers et al. (2023); Xu et al. (2023) explore quantization-aware fine-tuning (QAF) of large language models (LLMs). Specifically, the pivotal work, QLoRA (Dettmers et al., 2023), employs fine-tuning of LoRA adapters over a frozen quantized model. LoftQ (Li et al., 2023b) finds an optimized low-rank initialization for adapters. RoLoRA (Huang et al., 2024) incorporates rotation mechanisms during LoRA fine-tuning over quantized weights to mitigate outlier. LR-QAT (Bondarenko et al., 2024) trains adapters within the QAT framework, making the adapters quantization-aware. QA-LoRA (Xu et al., 2023) focuses on enabling a lossless merge of the trained adapter into the quantized weights. It achieves this by structuring the adapter to align with group-wise quantization parameters, allowing the adapter's influence to be absorbed into the zero factors.

Based on our literature review, the closest work to ours is QA-LoRA, which achieves lossless merging of adapters. However, QA-LoRA only adjusts the zero factors in group-wise quantization using the adapter, which restricts its fine-tuning capability. In contrast, our LoTA-QAF allows fine-tuning of all quantized weights within the quantization grid while maintaining the lossless merge property.

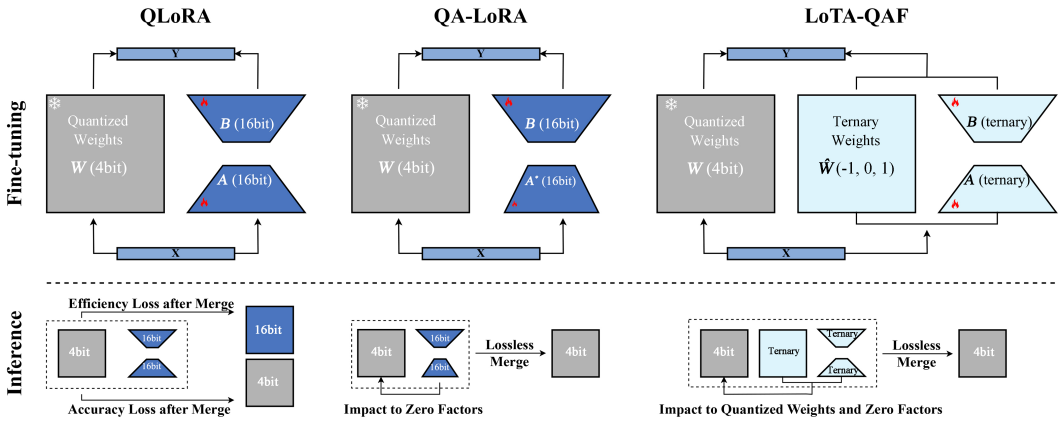

Figure 2: Comparison of prior adaptation methods with LoTA-QAF. A merge can result in efficiency loss, where model weights are de-quantized to preserve the adapter's precision but sacrifice inference efficiency, or accuracy loss, where the adapter's precision is sacrificed to maintain low-bit inference. A lossless merge avoids this trade-off because adapter weights are utilized identically during both the training and inference (after merge), ensuring the adapter's learned effect is fully preserved.

3 Method

3.1 Preliminaries

Low-Rank Adaptation. The pre-trained weight matrix $\mathbf{W} \in \mathbb{R}^{D_{\text{in}} \times D_{\text{out}}}$ is frozen during the fine-tuning process, where D_{in} and D_{out} represent the input and output dimensions, respectively. To enable efficient fine-tuning, trainable adapters $\mathbf{A} \in \mathbb{R}^{D_{\text{in}} \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times D_{\text{out}}}$ are introduced, where the rank $r \ll \min(D_{\text{in}}, D_{\text{out}})$. These adapters are multiplied to form a low-rank matrix $\mathbf{AB}$, which fine-tuning model with significantly fewer trainable parameters. The modified forward pass is computed as

$$\mathbf{y} = (\mathbf{W} + \frac{\alpha}{r}\mathbf{AB})^T\mathbf{x}, \tag{1}$$

where $\mathbf{A}$ is initialized with random Gaussian distribution, while $\mathbf{B}$ is initialized to zero. Here, α is a constant scaling factor that adjusts the magnitude of the low-rank update.

Asymmetric Affine Quantization. We perform N-bit affine quantization on the weight matrix $\mathbf{W}$, the quantization process maps $\mathbf{W}$ to an integer matrix $\mathbf{W}_{\text{int}}$. The dequantized approximation $\mathbf{W}_{\text{q}}$ as follows:

$$\mathbf{W}_{\text{q}} = s\mathbf{W}_{\text{int}} + z = s\left\lfloor \frac{\mathbf{W} - z}{s} \right\rceil + z, \tag{2}$$

where the scaling factor $s = (\max(\mathbf{W}) - \min(\mathbf{W}))/(2^N - 1)$ and the zero factor $z = \min(\mathbf{W})$. The symbol $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. All elements in the integer matrix $\mathbf{W}_{\text{int}}$ belong to the set $\{0, 1, \ldots, 2^N - 1\}$.

3.2 Lossless Ternary Adaptation

When fine-tuning quantized models using low-rank adaptation to compensate for quantization loss or adapt to a specific task, the key aspect is adjusting quantized weights $\mathbf{W}_{\text{int}}$. Moreover, the merge process should not quantize or truncate the adapter weights, which would reintroduce quantization errors at the adapter level. Consequently, we design the ternary adaptation, which losslessly merges the ternary adaptation into the quantized weights. This adaptation utilizes trainable ternary adapters $\mathbf{A}_{\text{T}} \in \{-1, 0, 1\}^{D_{\text{in}} \times r}$ and $\mathbf{B}_{\text{T}} \in \{-1, 0, 1\}^{r \times D_{\text{out}}}$. Consistent with low-rank adaptation principles, the rank $r \ll \min(D_{\text{in}}, D_{\text{out}})$. For initialization, we apply Kaiming normal initialization (He et al.,

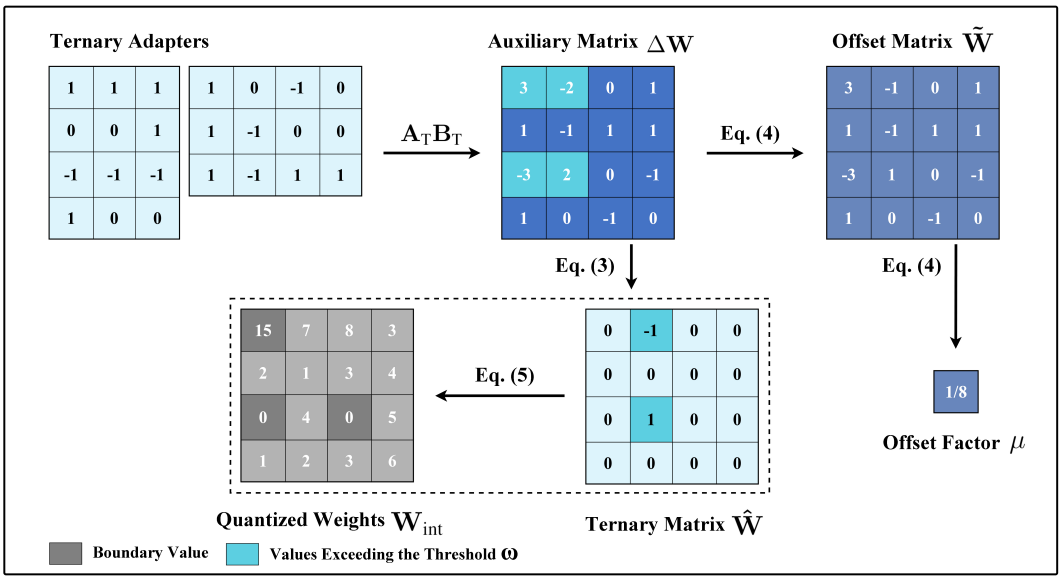

Figure 3: Illustration of LoTA-QAF. The process is demonstrated using a small 4x4 matrix under 4-bit quantization, with adapter rank r set to 3 and threshold ω to 1. Notably, quantized weights have boundary values (e.g., 0 and 15 for 4-bit, where 15 cannot be incremented and 0 cannot be decremented), requiring overflow prevention.

2015) to $\mathbf{A}_T$ and then ternarize its elements using a threshold, calculated as 0.75 times the mean absolute value of the sampled weights (Li et al., 2016), while $\mathbf{B}_T$ is initialized to zeros.

Notably, the product of the ternary adapters $\mathbf{A}_T$ and $\mathbf{B}_T$ forms the auxiliary matrix $\Delta\mathbf{W} = \mathbf{A}_T\mathbf{B}_T$. Its elements $\Delta\mathbf{W}_{ij}$ are integers within the range $[-r, r]$, where $i \in \{1, \ldots, D_{\text{in}}\}$ and $j \in \{1, \ldots, D_{\text{out}}\}$. The auxiliary matrix $\Delta\mathbf{W}$ is mapped to the ternary matrix $\hat{\mathbf{W}} \in \{-1, 0, 1\}^{D_{\text{in}} \times D_{\text{out}}}$ using a threshold $\omega \in (0, r)$, as follows:

$$\hat{\mathbf{W}}_{ij} = \text{sign}(\Delta\mathbf{W}_{ij}) \cdot \mathbb{I}_{|\Delta\mathbf{W}_{ij}| > \omega}, \tag{3}$$

where $\text{sign}(\cdot)$ denotes the sign function, and $\mathbb{I}_{|\Delta\mathbf{W}_{ij}|>\omega}$ represents the indicator function, which evaluates to 1 if $|\Delta\mathbf{W}_{ij}| > \omega$ and 0 otherwise. The resulting ternary matrix $\hat{\mathbf{W}}$ has the same dimensions as the quantized weight matrix $\mathbf{W}_{\text{int}}$. Consequently, $\hat{\mathbf{W}}$ is used to adjust the quantized weights $\mathbf{W}_{\text{int}}$, compensating for quantization errors or adapting the model to specific tasks. Additionally, $\hat{\mathbf{W}}$ can be losslessly merged into $\mathbf{W}_{\text{int}}$ after fine-tuning, preserving the inference efficiency of the quantized weights. We provide further analysis of the threshold ω in Section 4.3.

Furthermore, an offset factor μ is introduced, which represents an average offset relative to $\Delta\mathbf{W}$, and ultimately absorbed into the zero factor. The calculation of μ relies on the offset matrix $\tilde{\mathbf{W}}$. This offset matrix $\tilde{\mathbf{W}}$ and the offset factor μ are then computed as:

$$\begin{aligned} \tilde{\mathbf{W}}_{ij} &= \Delta\mathbf{W}_{ij} - \omega\hat{\mathbf{W}}_{ij} \\ \mu &= \frac{\sum_{i=1}^{D_{\text{in}}} \sum_{j=1}^{D_{\text{out}}} \tilde{\mathbf{W}}_{ij}}{D_{\text{in}} D_{\text{out}}}, \end{aligned} \tag{4}$$

where offset factor μ can be performed at different granularity, such as per-tensor, per-group, or per-channel, depending on the specific quantization method employed.

Following the fine-tuning stage, the lossless ternary adaptation merge mechanism is defined as:

$$\begin{aligned} \mathbf{W}'_{\text{int}} &= \mathbf{W}_{\text{int}} + \hat{\mathbf{W}} \\ z' &= z + s\mu \end{aligned} \tag{5}$$

where the final weights $\mathbf{W}'_{\text{int}}$ belong to the set $\{0, 1, \ldots, 2^N - 1\}$. Equivalently, the forward pass of the unmerged module is defined as $\mathbf{y} = (s \cdot \mathbf{W}'_{\text{int}} + z')^T\mathbf{x}$. Therefore, the lossless ternary adaptation

preserves low-bit computational efficiency and avoids the reintroduction of quantization loss at the adapter level.

3.3 Ternary Signed Gradient Descent

Inspired by the application of signed gradient descent (SignSGD) for quantization-related parameter updates in AutoRound (Cheng et al., 2023), we propose t-SignSGD for optimizing ternary adapters. Indeed, Balles et al. (2020); Li et al. (2024, 2023a); Safaryan and Richtárik (2021) demonstrate that SignSGD is beneficially applied to updates within discrete and constrained domains. This characteristic is highly relevant to our ternary adapters, which are constrained to the discrete values $\{-1, 0, 1\}$. Specifically, our proposed t-SignSGD performs updates on the ternary adapters (e.g., $\mathbf{A}_{\mathrm{T}}$). Let $g_t = \nabla_{\mathbf{A}_{\mathrm{T}}} \mathcal{L}|_{\mathbf{A}_{\mathrm{T},t}}$ denote the gradient at iteration t. To focus updates on prominent gradients, we employ a fixed minimum gradient threshold τ (e.g., 1×10^{-9}) and a dynamic percentile-based threshold σ_t. The threshold σ_t is determined at each iteration t based on a specific percentile of the gradient magnitudes, selecting a certain proportion of the largest gradients for update. This proportion of selected gradients is initially the top 5% and then linearly decays to 0.01% during training. The t-SignSGD update is performed as follows:

$$\mathbf{A}_{\mathrm{T},t+1} = \mathrm{clip}\left(\mathbf{A}_{\mathrm{T},t} - \mathrm{sign}(g_t) \cdot \mathbb{I}_{|g_t| > \max(\tau, \sigma_t)}, -1, 1\right), \tag{6}$$

where the indicator function $\mathbb{I}_{(\cdot)}$ equals 1 if the condition ($|g_t| > \max(\tau, \sigma_t)$) is satisfied, and 0 otherwise. The $\mathrm{clip}(\cdot, -1, 1)$ function ensures that the updated adapter weights $\mathbf{A}_{\mathrm{T},t+1}$ are strictly constrained to ternary set $\{-1, 0, 1\}$. Notably, Eq. (6) does not use a learning rate to scale the sign of the gradient. Instead, if a ternary adapter weight $\mathbf{A}_{\mathrm{T}}$ is selected for an update (i.e., its gradient magnitude $|g_t|$ exceeds the threshold $\max(\tau, \sigma_t)$), its value is adjusted by $-\mathrm{sign}(g_t)$. Therefore, the threshold σ_t play a crucial role in determining the *selectivity* of updates. By targeting only the top percentile of gradient magnitudes (e.g., the top 5%), σ_t acts as an adaptive mechanism that focuses updates on the most salient ternary adapter weights at each iteration t.

The convergence of t-SignSGD is primarily governed by its update rule, which is designed for the discrete and bounded nature of the ternary adapters. The key component is the indicator function $\mathbb{I}_{|g_t| > \max(\tau, \sigma_t)}$ in Eq. (6), which acts as an adaptive and dynamic selection mechanism. It ensures stable convergence in two ways: i) Stability via noise filtering (preventing oscillation). In discrete optimization, small, noisy gradients can cause parameters to oscillate unstably between states (e.g., flipping between 0 and 1). Our thresholding mechanism filters out these low-magnitude gradients, ensuring that updates are only performed when the gradient signal is strong and reliable. This addresses the core challenge in sign-based methods related to low signal-to-noise ratios, effectively preventing unstable "jitter" and promoting a smoother convergence path. ii) Convergence via annealing-like dynamics. The decaying percentile threshold σ_t introduces an annealing-like, coarse-to-fine search strategy. Early in training (σ_t is high), the top percentile of gradients triggers updates. This focuses the optimization on the most impactful parameters, allowing the model to quickly perform "broad-stroke" adjustments and find a promising region in the vast solution space. Later in training (σ_t is low), allowing for more fine-grained adjustments. This enables the model to refine its solution within the promising region, analogous to the exploitation phase in classic search algorithms. This dynamic balance between exploration and exploitation is a well-known strategy for improving convergence in complex landscapes. Further analysis of parameters and convergence is provided in Section 4.3.

4 Experiments

4.1 Experimental setup

Models and Quantization. We conduct experiments on several large language models: Llama 3.1 8B, Qwen 2.5 14B, Qwen 2.5 32B, and Llama 3.3 70B. GPTQ (Frantar et al., 2022) asymmetric quantization is applied to all these models, with a group size of 64 for Llama 3.1 8B and Qwen 2.5 14B, and 128 for Qwen 2.5 32B and Llama 3.3 70B. For calibration, we use 1024 samples from the C4 dataset (Raffel et al., 2019).

Tasks. We focus on quantization-aware fine-tuning (QAF) and implement two fine-tuning paradigms: performance-recovery and task-specific. For performance-recovery fine-tuning, we

Table 1: Accuracy (%) of performance-recovery and task-specific. An em dash (–) indicates unobtainable results, as the evaluation script requires strict format alignment with fine-tuning data.

Method	#Bit	MMLU (5-shot)					Task-Specific (0-shot)		
		Hums.	STEM	Social	Other	Avg.	GSM8K	SQL	ViGGO
LLaMA-70B	16	*80.30*	*75.26*	*88.94*	*84.58*	*82.23*	–	–	–
GPTQ	4	80.09	75.04	88.79	84.39	81.81	–	–	–
GPTQ+LoRA	4+16	80.19	74.98	88.79	84.39	81.83	90.14	82.90	89.72
QA-LoRA	4	80.11	75.01	88.89	84.29	81.81	84.32	77.65	85.35
LoTA-QAF	4	80.12	75.07	88.82	84.36	**81.84**	87.04	78.50	86.46
GPTQ	3	77.62	71.30	86.38	82.17	79.13	–	–	–
GPTQ+LoRA	3+16	77.58	71.27	86.48	82.23	79.14	88.17	80.20	88.09
QA-LoRA	3	77.66	71.27	86.45	82.07	79.13	82.65	75.80	76.38
LoTA-QAF	3	77.68	71.30	86.42	82.36	**79.20**	83.24	77.40	85.38
GPTQ	2	39.00	35.08	46.83	46.64	41.53	–	–	–
GPTQ+LoRA	2+16	52.58	45.86	65.91	62.63	56.22	67.54	72.70	79.35
QA-LoRA	2	50.03	44.08	61.68	58.83	53.20	61.80	70.70	68.44
LoTA-QAF	2	52.96	47.10	67.34	62.83	**56.98**	63.98	73.40	75.38
Qwen-32B	16	*77.95*	*82.43*	*88.72*	*84.96*	*82.92*	–	–	–
GPTQ	4	77.90	82.11	88.56	83.71	82.47	–	–	–
GPTQ+LoRA	4+16	78.13	82.46	88.59	83.52	82.59	83.09	88.70	77.29
QA-LoRA	4	77.98	82.02	88.63	83.68	82.48	76.82	86.80	70.93
LoTA-QAF	4	78.07	82.33	88.76	84.10	**82.70**	78.74	87.60	76.18
GPTQ	3	75.66	79.45	87.36	81.17	80.29	–	–	–
GPTQ+LoRA	3+16	75.75	79.42	87.46	81.53	80.42	82.34	88.90	74.70
QA-LoRA	3	75.58	79.42	87.33	81.43	80.31	75.44	80.60	57.20
LoTA-QAF	3	76.00	79.51	87.59	81.43	**80.53**	76.04	83.20	60.94
GPTQ	2	34.03	32.54	37.60	39.46	35.68	–	–	–
GPTQ+LoRA	2+16	44.06	43.04	54.92	54.23	48.46	57.55	82.40	69.99
QA-LoRA	2	39.34	42.02	53.04	54.49	46.30	54.74	80.40	55.44
LoTA-QAF	2	47.31	45.42	60.87	59.93	**52.65**	57.92	82.20	63.71
Qwen-14B	16	*74.71*	*77.86*	*87.46*	*82.39*	*79.91*	–	–	–
GPTQ	4	74.79	76.91	87.26	81.88	79.57	–	–	–
GPTQ+LoRA	4+16	74.79	77.04	87.42	81.94	79.65	80.06	87.70	74.05
QA-LoRA	4	74.90	76.78	87.36	81.82	79.58	76.10	83.90	68.47
LoTA-QAF	4	74.86	77.20	87.52	82.14	**79.77**	78.37	84.50	71.10
GPTQ	3	71.07	72.41	84.24	79.79	76.19	–	–	–
GPTQ+LoRA	3+16	71.48	73.20	84.37	80.11	76.60	72.18	87.40	71.93
QA-LoRA	3	71.16	72.79	84.21	79.88	76.31	66.49	77.30	57.36
LoTA-QAF	3	71.54	72.79	84.76	80.17	**76.63**	70.36	79.50	64.45
GPTQ	2	30.01	32.03	33.57	32.15	31.72	–	–	–
GPTQ+LoRA	2+16	39.17	41.29	48.13	43.90	42.66	37.23	80.60	61.59
QA-LoRA	2	33.79	36.60	42.18	49.79	39.80	34.69	58.10	43.87
LoTA-QAF	2	45.23	42.85	55.44	49.15	**47.80**	36.25	62.80	52.63
LLaMA-8B	16	*64.17*	*60.39*	*77.64*	*73.09*	*68.25*	–	–	–
GPTQ	4	62.19	58.71	76.37	72.93	66.89	–	–	–
GPTQ+LoRA	4+16	62.57	58.33	76.73	73.22	67.08	70.20	76.70	88.64
QA-LoRA	4	62.06	58.71	76.54	73.25	66.96	68.69	74.10	51.15
LoTA-QAF	4	62.95	58.55	76.70	72.93	**67.18**	70.05	74.60	60.30
GPTQ	3	32.48	36.66	39.42	49.08	38.61	–	–	–
GPTQ+LoRA	3+16	50.24	46.91	63.24	60.80	54.68	65.66	75.80	82.20
QA-LoRA	3	45.55	46.94	61.33	59.35	52.37	62.17	72.20	43.86
LoTA-QAF	3	49.71	49.29	65.29	62.63	**55.89**	63.53	73.10	55.96
GPTQ	2	25.89	27.53	26.71	26.30	26.53	–	–	–
GPTQ+LoRA	2+16	27.21	28.64	26.91	26.68	27.35	34.12	72.50	80.70
QA-LoRA	2	25.70	26.67	27.72	26.62	26.56	17.36	56.20	37.18
LoTA-QAF	2	28.25	26.90	28.44	30.51	**28.49**	19.48	67.00	41.98

utilize the Alpaca (Taori et al., 2023) and subsequently evaluate 5-shot performance on the Massively Multitask Language Understanding (MMLU) benchmark (Hendrycks et al., 2020). For task-specific fine-tuning, we select three datasets: GSM8K (Cobbe et al., 2021), with 7.47k training and 1.32k test samples; SQL generation (Yu et al., 2018; Zhong et al., 2017), with 30k training and 1k test samples; ViGGO (Juraska et al., 2019), with 5.1k training and 1.08k test samples. We use the lm-eval framework (Gao et al., 2024) to test the MMLU benchmark and follow the custom evaluation framework from HALO (Ashkboos et al., 2025) for the task-specific evaluations. These datasets are chosen because they represent diverse specialized tasks that present unique challenges and require distinct model capabilities. Their inclusion allows us to clearly illustrate the practical significance of the QAF approaches in developing quantized models for specific applications.

Baselines and Hyper-parameters. We benchmark LoTA-QAF against two state-of-the-art methods: LoRA (Hu et al., 2022) and QA-LoRA (Xu et al., 2023). For fine-tuning quantized models to recover performance and align them with specific tasks, LoRA remains a widely adopted method (applied to GPTQ-quantized models, it can be considered a QLoRA (Dettmers et al., 2023) variant). Although LoRA introduces 16-bit adapters, resulting in reduced inference efficiency, the 16-bit precision of its adapters also enables more precise adjustment of the quantized weights. Moreover, as the closest work to ours, QA-LoRA maintains low-bit inference efficiency by lossless merging the adapters into zero factors.

Following QA-LoRA, we use a paged AdamW optimizer, a maximum gradient norm of 0.3, a batch size of 64, a source length of 1024, and a target length of 256. For performance-recovery fine-tuning, we set the learning rate to 1×10^{-5} for the 8B and 14B models, and 5×10^{-6} for the 32B and 70B models. The number of fine-tuning steps is 300 for Alpaca. For task-specific fine-tuning, the learning rate is set to 5×10^{-4} for the 8B and 14B models, and 1×10^{-4} for the 32B and 70B models. Single-epoch experiments are performed on the training sets of GSM8k, SQL generation, and ViGGO. Regarding adapter settings, the rank r is 64 for the 8B and 14B models, and 32 for the 32B and 70B models. Additionally, the coefficient α is twice the rank. Regarding the hyper-parameters of LoTA-QAF, we set the threshold ω to $0.75r$ for Alpaca, GSM8K and SQL generation, and ω to $0.875r$ for ViGGO. The dynamic percentile-based threshold, σ_t, is initialized to the top 5% and linearly decays to 0.1% during the first 80% of the training phase. For the final 20% of training (i.e., from 80% to 100% completion), it is fixed at 0.01%. Section 4.3 provides a detailed analysis of ω and σ_t. All experiments are conducted on one NVIDIA A800 GPU.

4.2 Main Results

Performance-Recovery Fine-Tuning. For performance-recovery fine-tuning, our objective is to restore the performance of quantized models to the level of the 16-bit base models. As Table 1 shows, across four model sizes and three quantization bits, the three quantization-aware fine-tuning (QAF) methods all achieve improvements compared to the quantized baseline. For instance, with 3-bit quantization, the fine-tuned performance surpasses that of the quantized model by up to 17.28% on Llama 3.1 8B. Similarly, for 2-bit quantization, the fine-tuned performance exceeds that of the quantized model by up to 16.97% on Qwen 2.5 32B. This indicates that in the low-bit regimes, there remains a performance potential that existing Post-Training Quantization (PTQ) methods may not fully exploit, but which can be harnessed through fine-tuning. This highlights the significant development potential of QAF methods.

Furthermore, LoTA-QAF outperforms LoRA (GPTQ+LoRA in Table 1) and QA-LoRA. On Qwen 2.5 14B 2-bit, LoTA-QAF improves performance by 5.14% compared to LoRA, the second-best performing method in this comparison. This superiority is attributed to LoTA-QAF directly adjusting the quantized weights and aligning with the quantization grid, which effectively recovers performance lost by quantization. In contrast, while LoRA utilizes a 16-bit adapter, it lacks a direct quantization-aware mechanism. As noted earlier, QA-LoRA only adjusts the zero factors in group-wise quantization, which limits its fine-tuning capability.

Task-Specific Fine-Tuning. For task-specific fine-tuning, the objective is to enable quantized models to acquire the knowledge and patterns specific to particular tasks. This necessitates that the model learn fine-grained capabilities specific to the task, rather than merely recovering general abilities. As shown in Table 1, LoTA-QAF outperforms QA-LoRA but is surpassed by LoRA. LoRA's 16-bit adapter possesses a higher representational capacity, enabling it to more effectively capture

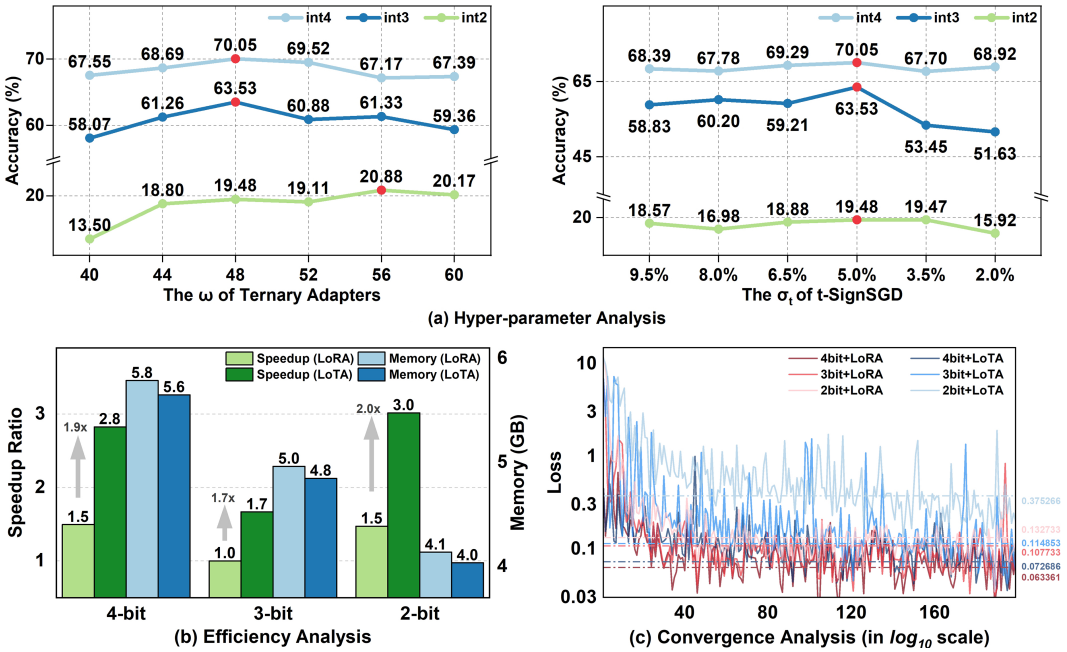

Figure 4: Further analysis of LoTA-QAF

these complex task-specific details. This higher precision is particularly crucial for heavily quantized models (e.g., 2-bit models). However, it is important to note the difference in inference efficiency: LoTA-QAF and QA-LoRA perform low-bit inference once their adapters are merged, whereas inference with LoRA requires computation involving the 16-bit adapter (as detailed in Section 4.3).

4.3 Further Analysis of LoTA-QAF

Hyper-parameter Analysis. We investigate the hyper-parameters ω of the ternary adapter (TA) and σ_t of ternary signed gradient descent (t-SignSGD) through parameter experiments conducted on GSM8K with 4, 3 and 2 bit-widths of the Llama 3.1 8B model, with results shown in Fig. 4. First, TA employs a threshold ω to determine the extent to which it influences the quantized weights. This threshold ω is analogous to the LoRA coefficient α in Equation 1. A smaller ω allows values in the auxiliary matrix $\Delta\mathbf{W}$ to affect the quantized weights more readily, as detailed in Equations 3 and 5. Specifically, the threshold ω is a hyper-parameter related to the rank r. We set the rank r to 64 for the 8B model and experiment with ω values of $\{40, 44, 48, 52, 56, 60\}$. We observe that setting ω to 48 yields favorable performance across all three bit-widths. Notably, for 2-bit quantization, a relatively larger ω achieves better experimental results. This is because, in 2-bit quantization, each quantized weight possesses only four possible values. Consequently, updating the quantized weights with a larger ω, which exerts a more conservative influence, leads to improved outcomes.
Second, the dynamic percentile-based threshold σ_t (where t is the iteration step) acts as a learning rate in t-SignSGD, as explained in Equation 6. We conduct parameter experiments by selecting initial σ_t values corresponding to the top $\{9.5\%, 8.0\%, 6.5\%, 5.0\%, 3.5\%, 2.0\%\}$ of gradient magnitudes. For the 8B model, an initial σ_t of 5.0% (representing the top 5.0% of gradients) consistently yields favorable results. Conversely, selecting smaller initial σ_t values leads to a noticeable decline in performance for both 3-bit and 2-bit quantization. This observation can be attributed to the nature of task-specific fine-tuning, which necessitates more substantial adjustments to model parameters compared to recovery performance. This aligns with our general learning rate strategy for LoRA and QA-LoRA, where the learning rate for recovery performance is set lower than that for task-specific fine-tuning. Further hyper-parameter analysis is provided in the Appendix B.

Efficiency Analysis. We measure the inference speed and memory usage of LoRA (utilizing a forward pass with 4-bit quantized weights and 16-bit adapters) and LoTA (where the adapter is losslessly merged into the quantized weights). These measurements are conducted on the ViGGO dataset using the Llama 3.1 8B model with 4, 3 and 2 bit-widths quantization. For 4-bit and 2-bit quantized models, the TritonV2QuantLinear kernel is used. For the 3-bit quantized model, the

TorchQuantLinear kernel is employed. To evaluate inference speed, we measure throughput (tokens per second) with batch sizes ranging from 8 to 128 and a maximum inference length of 512 tokens. The speedup ratio, benchmarked against the slowest configuration (3-bit quantized weights with LoRA), is reported in Fig.4. LoTA achieves speedups of 1.9x, 1.7x, and 2.0x over LoRA across these respective bit-widths, revealing the efficiency gains from merging the adapter post fine-tuning. In terms of memory usage, LoTA also exhibits a slight advantage.

Convergence Analysis. Regarding convergence, we evaluate LoRA and LoTA on Llama 3.1 8B for SQL generation with 4, 3 and 2 bit-widths quantization. LoRA achieves best convergence across all three bit-widths. Notably, at 2-bit, LoRA reaches a convergence loss of 0.132, whereas LoTA achieves 0.375, showing LoRA's 16-bit adapter stability. However, LoTA, optimized with t-SignSGD, also converges. For 4-bit and 3-bit quantization, its convergence loss differs from LoRA < 0.01. Considering that LoTA can be losslessly merged into the quantized weights, whereas LoRA cannot, this makes LoTA a highly competitive method, especially when deployment efficiency is a priority.

5 Conclusion

Fine-tuning quantized LLMs for edge devices suffers from merging issues. LoTA-QAF enables in-grid adjustment of all quantized weights and lossless merging of ternary adapters using novel ternary adaptation and ternary signed gradient descent (t-SignSGD), and preserves performance by avoiding merging-induced accuracy loss. The Appendix details ternary adapters implementation, further parameters analysis, training efficiency analysis, limitations, and future work.

6 Acknowledgments

The work of Long Shi was supported by the National Natural Science Foundation of China under Grant 62201475 and Sichuan Science and Technology Program under Grant 2024NSFSC1436. The work of Xuming Hu was supported by the National Natural Science Foundation of China (Grant No.62506318); Guangdong Provincial Department of Education Project (Grant No.2024KQNCX028); CAAI-Ant Group Research Fund; Scientific Research Projects for the Higher-educational Institutions (Grant No.2024312096), Education Bureau of Guangzhou Municipality; Guangzhou-HKUST(GZ) Joint Funding Program (Grant No.2025A03J3957), Education Bureau of Guangzhou Municipality.

References

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Saleh Ashkboos, Mahdi Nikdan, Soroush Tabesh, Roberto L Castro, Torsten Hoefler, and Dan Alistarh. Halo: Hadamard-assisted lossless optimization for efficient low-precision llm training and fine-tuning. *arXiv preprint arXiv:2501.02625*, 2025.

Lukas Balles, Fabian Pedregosa, and Nicolas Le Roux. The geometry of sign gradient descent. *arXiv preprint arXiv:2002.08056*, 2020.

Yelysei Bondarenko, Riccardo Del Chiaro, and Markus Nagel. Low-rank quantization-aware training for llms. *arXiv preprint arXiv:2406.06385*, 2024.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. Efficientqat: Efficient quantization-aware training for large language models. *arXiv preprint arXiv:2407.11062*, 2024.

Wenhua Cheng, Weiwei Zhang, Haihao Shen, Yiyang Cai, Xin He, Kaokao Lv, and Yi Liu. Optimize weight rounding via signed gradient descent for the quantization of llms. *arXiv preprint arXiv:2309.05516*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

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.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

Hang Guo, Yawei Li, Tao Dai, Shu-Tao Xia, and Luca Benini. Intlora: Integral low-rank adaptation of quantized diffusion models. *arXiv preprint arXiv:2410.21759*, 2024.

Muhammad Usman Hadi, Rizwan Qureshi, Abbas Shah, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Jia Wu, Seyedali Mirjalili, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. *Authorea Preprints*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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.

Xijie Huang, Zechun Liu, Shih-Yang Liu, and Kwang-Ting Cheng. Rolora: Fine-tuning rotated outlier-free llms for effective weight-activation quantization. *arXiv preprint arXiv:2407.08044*, 2024.

Hyesung Jeon, Yulhwa Kim, and Jae-joon Kim. L4q: Parameter efficient quantization-aware fine-tuning on large language models. *arXiv preprint arXiv:2402.04902*, 2024.

Juyong Jiang, Fan Wang, Jiasi Shen, Sungju Kim, and Sunghun Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

Juraj Juraska, Kevin K Bowden, and Marilyn Walker. Viggo: A video game corpus for data-to-text generation in open-domain conversation. *arXiv preprint arXiv:1910.12129*, 2019.

Tanishq Kumar, Zachary Ankner, Benjamin F Spector, Blake Bordelon, Niklas Muennighoff, Mansheej Paul, Cengiz Pehlevan, Christopher Ré, and Aditi Raghunathan. Scaling laws for precision. *arXiv preprint arXiv:2411.04330*, 2024.

Bingrui Li, Wei Huang, Andi Han, Zhanpeng Zhou, Taiji Suzuki, Jun Zhu, and Jianfei Chen. On the optimization and generalization of two-layer transformers with sign gradient descent. *arXiv preprint arXiv:2410.04870*, 2024.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Xiuxian Li, Kuo-Yi Lin, Li Li, Yiguang Hong, and Jie Chen. On faster convergence of scaled sign gradient descent. *IEEE Transactions on Industrial Informatics*, 20(2):1732–1741, 2023a.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. *ICLR'24 Oral*, 2023b.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6: 87–100, 2024.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

Michael McTear and Marina Ashurkina. *Transforming conversational AI: Exploring the power of large language models in interactive conversational agents*. Springer Nature, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and PeterJ. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv: Learning,arXiv: Learning*, 2019.

Mher Safaryan and Peter Richtárik. Stochastic sign descent methods: New algorithms and better theory. In *International Conference on Machine Learning*, pages 9224–9234. PMLR, 2021.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

Zhongwei Wan, Xin Wang, Che Liu, Samiul Alam, Yu Zheng, Jiachen Liu, Zhongnan Qu, Shen Yan, Yi Zhu, Quanlu Zhang, et al. Efficient large language models: A survey. *arXiv preprint arXiv:2312.03863*, 2023.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. Qa-lora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint arXiv:2309.14717*, 2023.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*, 2018.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly define this paper's main contributions and scope. These contributions, summarized at the end of the Introduction (Section 1), are further detailed and supported in the Experiments (Section 4.2). All claims accurately reflect the presented results, and the paper's scope is clearly delineated.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, we discuss the limitations of our work. For instance, the convergence properties of our proposed method are analyzed in the Experiments (Section 4.3). Furthermore, a broader discussion of the work's limitations is provided in the Appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper does not present new mathematical theorems, or formal proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Our method LoTA-QAF process is detailed in the Method (Section 3) and illustrated in Fig. 3. Full experimental settings are disclosed in the Experiments (Section 4).

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Yes, we provide open access to all data and code necessary to faithfully reproduce the main experimental results presented in this paper. The supplementary material contains our source code, all relevant datasets links, and a comprehensive README file. The README provides clear, step-by-step instructions for environment setup, data preparation, and script execution.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Full experimental settings are disclosed in the Experiments (Section 4.1).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our experimental results are based on multiple independent runs in Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, sufficient details of the compute resources required to reproduce our experiments are provided at the end of Sections 4.1 and 4.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have thoroughly reviewed the NeurIPS Code of Ethics and confirm that all research presented in this paper strictly adheres to its guidelines, encompassing data privacy, responsible conduct, and transparent reporting of results and limitations.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses positive societal impacts. For instance, in the Introduction (Section 1), we highlight that our high-efficiency method reduces computational costs, contributing to environmental sustainability.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This work does not release high-risk data or models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, all existing assets used are publicly available, appropriately credited, and their licenses and terms of use are respected. Citations are provided in the paper.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: This paper does not introduce or release new public datasets or software packages.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This research does not involve crowdsourcing experiments or new studies with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This research does not involve new studies with human subjects that would require Institutional Review Board (IRB) approval.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: LLMs were not used as an important, original, or non-standard component of the core research methods presented in this paper.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Implementation of Ternary Adapters

The PyTorch framework does not natively support `ternary` or `torch.int2` data formats. Furthermore, the PyTorch deep learning training ecosystem primarily relies on floating-point computations. Therefore, to seamlessly integrate into this established system, we simulate `ternary` data types by representing values of $\{-1, 0, 1\}$ using `bfloat16`.

For the process of forming the ternary matrix $\hat{\mathbf{W}}$ via the auxiliary matrix $\Delta\mathbf{W} = \mathbf{A}_{\mathrm{T}}\mathbf{B}_{\mathrm{T}}$ (as described in Eq. (3) of the main text), we implement custom operations using `Triton` to enhance the execution efficiency on GPUs. Moreover, we update the quantized weights $\mathbf{W}_{\text{int}}$ which requires boundary checks [1] [2]. Specifically, through kernel fusion, `Triton` combines the formation of $\hat{\mathbf{W}}$ and the application of boundary checks (boolean masks) into a single, optimized GPU kernel. This fusion minimizes overhead and contributes to maintaining acceptable training costs for the LoTA-QAF.

Notably, the implementation of ternary adapters faces challenges primarily due to framework limitations. Methodologically, LoTA-QAF employs ternary adapters, which offer significant advantages in storage and computational efficiency over 16-bit adapters. Consequently, further engineering for production-level deployment is valuable and represents a promising avenue for future work (see Section F). Currently, the training costs are acceptable with an implementation based on PyTorch and accelerated by `Triton`. A detailed discussion on training efficiency is available in Section C.

B Hyper-parameters of LoTA-QAF

To further analyze the hyperparameters of LoTA-QAF, we focus on two aspects: i) the Llama 3.1 8B model across various datasets, specifically (a) MMLU, (b) SQL, and (c) ViGGO (with GSM8K detailed in the main text); and ii) the Qwen 2.5 14B and 32B models on the GSM8K dataset, as shown in (d) and (e). For all analyses of ω (defined in Eq. (3)), we set σ_t to 5%. For the analysis of σ_t, we set ω to $0.75r$ (except $0.875r$ for ViGGO), where r is the rank. Other settings are detailed in Section 4.1.

Overall, more detailed hyperparameter tuning can reveal accuracies (e.g., as shown in Fig. 5 (b)) that exceed those reported in main text Table 1. This is because different quantization bit-widths, datasets, and model sizes often have their own more suitable parameter settings. The objective of our further hyperparameter analysis is to identify the characteristics of these settings and understand the LoTA-QAF properties they reflect.

For the Llama 3.1 8B model on various datasets: i) the ω parameter of the ternary adapters (TA, defined in Eq. (3)) primarily determines how intensity $\hat{\mathbf{W}}$ adjusts the quantized weights. For 4-bit quantized models, a smaller ω can potentially yield gains (e.g., Fig. 5 (b), int4), attributed to the robustness and adjustability of 4-bit models. However, for 3-bit and 2-bit quantization, a smaller ω generally leads to poorer performance (e.g., Fig. 5 (b) and (c), int3 and int2). Indeed, for 2-bit quantization, setting a higher ω to limit the adjustment intensity of $\hat{\mathbf{W}}$ on quantized weights conversely achieves higher accuracy (e.g., Fig. 5 (c), int2). ii) the σ_t parameter of t-SignSGD (defined in Eq. (6)), which functions as the learning rate in t-SignSGD, selects for update ternary adapter weights with gradient magnitudes in the top $x\%$. For both SQL and ViGGO datasets (Fig. 5 (b) and (c)), employing a larger σ_t results in higher accuracy. This aligns with our LoRA training configurations. Specifically,

[1] Two main approaches exist for performing these boundary checks: one is to decode the boundaries from the quantized weights during each forward propagation, which adversely affects training efficiency. Consequently, we adopt an alternative approach: we first identify the boundaries and then store them using a boolean packing technique (`def pack_bool_tensor()`) for use in the forward propagation. While this method introduces some memory overhead, the cost is acceptable as each boolean flag (1 bit) is packed.

[2] Boundary checks are not strictly necessary during the training phase for 4-bit or even 3-bit quantization, as the probability of encountering boundary conditions is relatively low in schemes that utilize 16 or 8 distinct values, respectively, especially considering that the number of values at the boundaries is typically smaller than that of values near the center. However, for 2-bit quantization, which involves only 4 distinct values, boundary checks become crucial. Specifically, neglecting boundary checks during training and only applying them when loading or merging the adapter can lead to inconsistencies between training behavior and the operational results of the adapter for 2-bit quantized models. In our experiments, we enable boundary checks universally (including in training efficiency tests) to ensure consistency, even though they could be optionally disabled for 4-bit and 3-bit scenarios during training without a significant adverse impact.

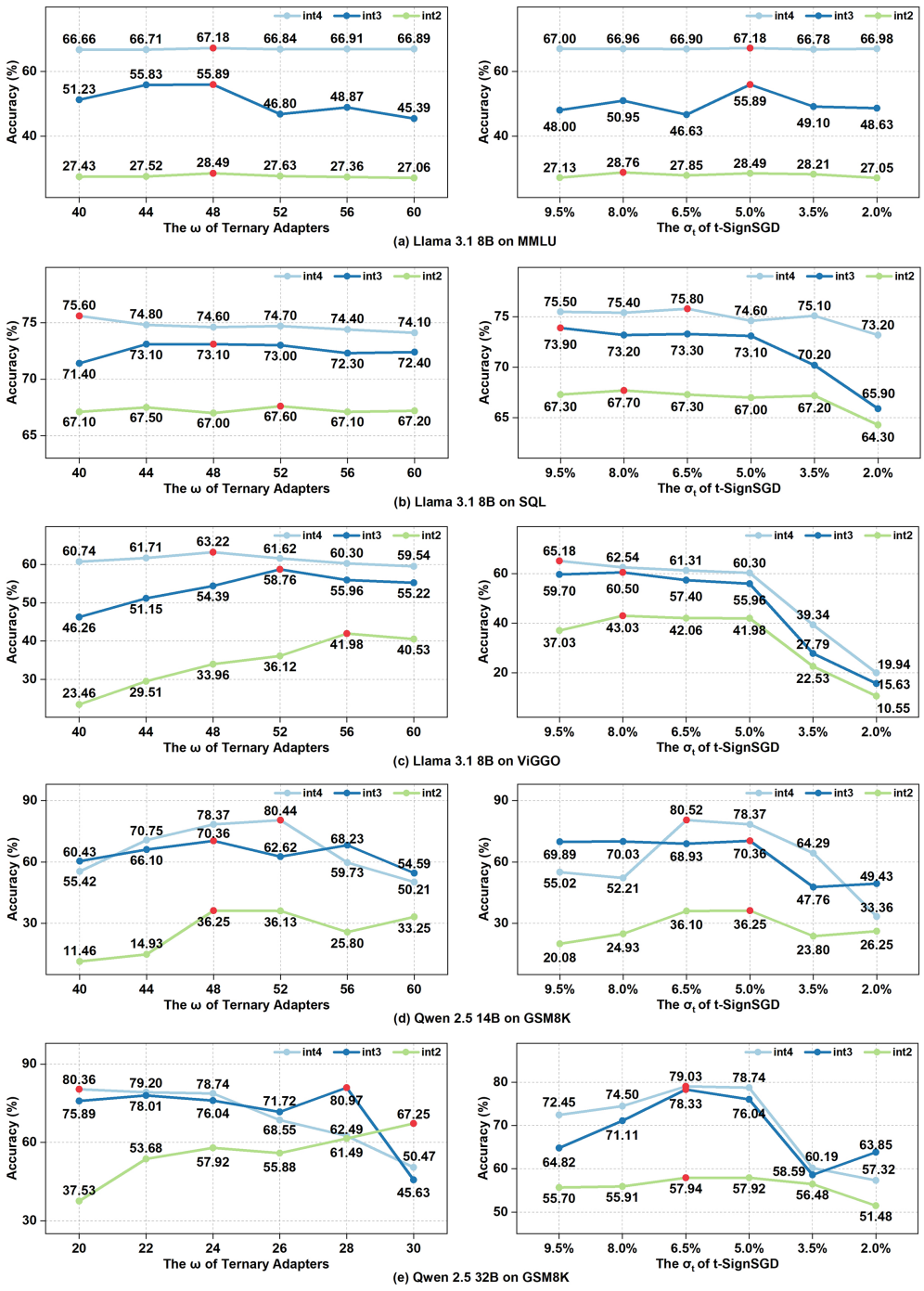

Figure 5: Parameter analysis for Llama 3.1 8B on various datasets in (a) MMLU, (b) SQL and (c) ViGGO. Parameter analysis for Qwen 2.5 14B and 32B on the GSM8K dataset in (d) and (e).

task-specific fine-tuning requires a higher learning rate than performance-recovery fine-tuning, as learning specific tasks demands greater model weight adjustments than performance recovery.

For the Qwen 2.5 14B and 32B models on the GSM8K dataset: i) setting ω to $0.9375r$ leads to a significant decrease in accuracy for 4-bit and 3-bit quantization (e.g., Fig. 5 (d) and (e)). Conversely, for 2-bit quantized weights, this ω $(0.9375r)$ value achieves higher accuracy. This is consistent with our earlier findings. ii) setting σ_t to either 6.5% or 5.0% is a robust choice. An overly small σ_t (e.g.,

2.0%) results in poorer accuracy (e.g. Fig. 5 (d) and (e) int4 and int2), limiting its ability to learn effectively. Furthermore, due to the high cost of extensive hyperparameter testing for 70B models, we offer a baseline configuration: setting ω to $0.75r$ and σ_t to 5.0% generally yields good accuracy. For further improvement, we suggest trying $\sigma_t = 2.0\%$ for performance-recovery fine-tuning and $\sigma_t = 6.5\%$ for task-specific fine-tuning. Regarding the ω setting for 70B models, suggests that experimenting with lower ω values may improve accuracy, which is attributed to the 4-bit quantized 70B model's robustness.

C Training Efficiency Analysis

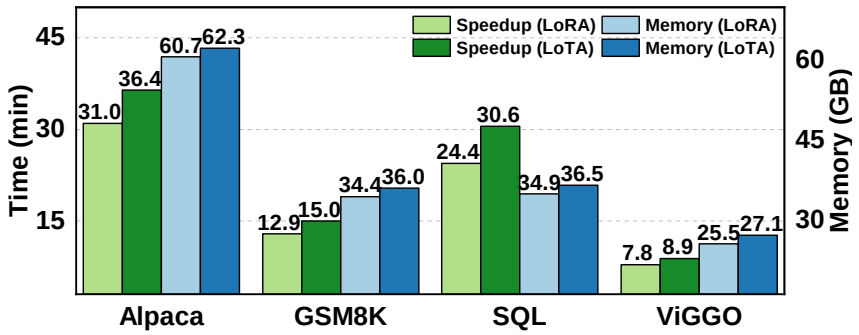

Figure 6: Comparative analysis of training efficiency for LoRA versus LoTA. The evaluation utilized the Llama 3.1 8B model with 4-bit precision on four datasets. Metrics reported are total training execution time and peak memory usage. For parameter details, including a batch size of 64, and other settings, please refer to the Section 4.1.

Our simulated '`ternary`' adapter utilizes '`bfloat16`' for implementation due to the lack of native support for the '`ternary`' dtype within the PyTorch framework. Consequently, LoTA-QAF does not exhibit a training efficiency advantage over LoRA. Specifically, LoTA requires 14.1%-25.4% more training time compared to LoRA. This increased training time is primarily attributed to the implementation of Eq. (4) and the process of aligning with the dequantization process of GPTQModel. Despite this higher training cost, LoTA's ternary adapters can be losslessly merged into quantized weights, a capability that LoRA does not offer. In terms of memory, LoTA incurs an additional 2.6%-6.3% overhead compared to LoRA. This overhead derives from the boundary checks detailed in Section A, though the overall cost remains modest. Section F discusses feasible directions for further enhancing LoTA's training efficiency.

D Ternary Adapter Weight Changes under t-SignSGD Training

Table 2: Analysis of ternary adapter weight changes.

Metric	Value	Percentage	Description
Total Parameters	167,772,160	100%	Total adapter parameters across 32 layers. Adapters are applied to 7 key projections within the attention and MLP blocks of each layer.
Increased Parameters	28,952,566	17.26%	Count of weights that changed in the positive direction.
Decreased Parameters	28,960,070	17.26%	Count of weights that changed in the negative direction.
Unchanged Parameters	109,859,524	65.48%	Count of weights that are unchanged.

To better understand the dynamics of our optimizer, we analyzed the parameter changes of the ternary adapters for Llama 3.1 8B after fine-tuning on the Alpaca dataset. Table 2 shows that the

rate of change in the adapters is nearly identical in both positive and negative directions (17.26% vs. 17.27%). This suggests that the adaptation is not a simple unidirectional shift but a balanced, bidirectional fine-tuning process to fit the target task. Moreover, one-third of the parameters changed during adaptation, while two-thirds remained unchanged. This phenomenon is a result of the t-SignSGD design, specifically the adaptive threshold $\mathbb{I}_{|g_t|>\max(\tau,\sigma_t)}$. This mechanism ensures our updates are based on a large gradient magnitude, which implies a high probability that the sign is correct. Conversely, it filters out updates associated with small gradient signals, which can be noisy or ill-scaled.

E Limitations

Regarding training efficiency, LoTA-QAF keeps its training costs at a level comparable to LoRA through `Triton`, but LoTA-QAF has not fully realized the potential high efficiency inherent in ternary adapters. A common challenge in current low-bit research is bridging the gap between theoretical and realized efficiency. Although `Triton` and `CUTLASS` offer pathways to achieve efficient implementations, its implementation is challenging.

Regarding convergence, LoTA-QAF exhibits less favorable convergence compared to LoRA (discussed in Section 4.3). This can be attributed to two main aspects: firstly, ternary adapters adjust the quantized weights based on the quantization grid. For 4-bit and 3-bit quantization, which involve 16 and 8 possible values respectively, such adjustments demonstrate reasonable robustness. However, under 2-bit quantization with only four possible values, these adjustments exhibit higher volatility. Secondly, t-SignSGD is a novel update mechanism that operates without a learning rate, and we do not incorporate first or second-order momentum. Moreover, the primary constraint is a linear decay of the threshold σ_t (as defined in Eq. (6)). While the t-SignSGD approach is effective for updating ternary adapters, our exploration of its optimization remains limited.

F Future Work

We believe that the current training efficiency disadvantage of the ternary adapter (TA) can be addressed. Although TA involves more computational logic during forward propagation, the operational advantages of `ternary` arithmetic can be further exploited. The auxiliary matrix $\Delta\mathbf{W} = \mathbf{A}_{\mathrm{T}}\mathbf{B}_{\mathrm{T}}$ is calculated using `bfloat16`, but executing operations on ternary values $\{-1, 0, 1\}$ could further improve efficiency. The forward execution logic in Eq. (5) is merged with the dequantization process of GPTQModel. However, given that $\hat{\mathbf{W}}$ is a sparse ternary matrix, the potential for dedicated sparse computation logic warrants exploration. Such implementations, however, must consider not only computational efficiency but also ensure the correct functioning of the gradient mechanism for the `ternary` data type within the PyTorch framework.

Our exploration of t-SignSGD has been limited. Further research could focus on incorporating momentum mechanisms, refining constraints on t-SignSGD, and applying optimization techniques (e.g., cosine annealing, cyclical learning rates, or adaptive step-size strategies). In t-SignSGD, we replaced the learning rate with a dynamic percentile-based threshold, σ_t, to select the top-$x\%$ of gradient signs for directly updating the ternary adapter weights within the set $\{-1, 0, 1\}$. Under this mechanism, a more detailed analysis of gradient distribution characteristics could be pursued to guide the selection of ternary adapter weights for updates, potentially offering improved convergence properties.

Further enhancements to LoTA-QAF could also be explored by expanding the representational range of the auxiliary matrix $\hat{\mathbf{W}}$. Currently, this matrix is constrained to the set $\{-1, 0, 1\}$. However, the LoTA mechanism, as outlined in Eq. (3), could be extended to broaden the value range of $\hat{\mathbf{W}}$ to include, for instance, $\{-2, -1, 0, 1, 2\}$, or even a wider set of integers. Such an extension could further enhance the optimization capability for quantized weights. It is important to note, though, that this approach might be less suitable for `int2` or even `int3` quantization formats due to the potentially large adjustment step sizes. Nevertheless, for `int4` or `int8` quantization, exploring this direction is highly valuable.