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

Junyu CHEN
Southwestern University of Finance and Economics

with Junzhuo LI, Zhen PENG, Wenjie WANG, Yuxiang REN, Long SHI, Xuming HU

Code: github.com/Kingdalfgoodman/LoTA-QAF
Email: Kingdalfgoodman@foxmail.com

Main Contributions

LoTA-QAF provides a powerful and efficient solution for fine-tuning quantized models.

1. Lossless Merge, In-Grid Adaptation

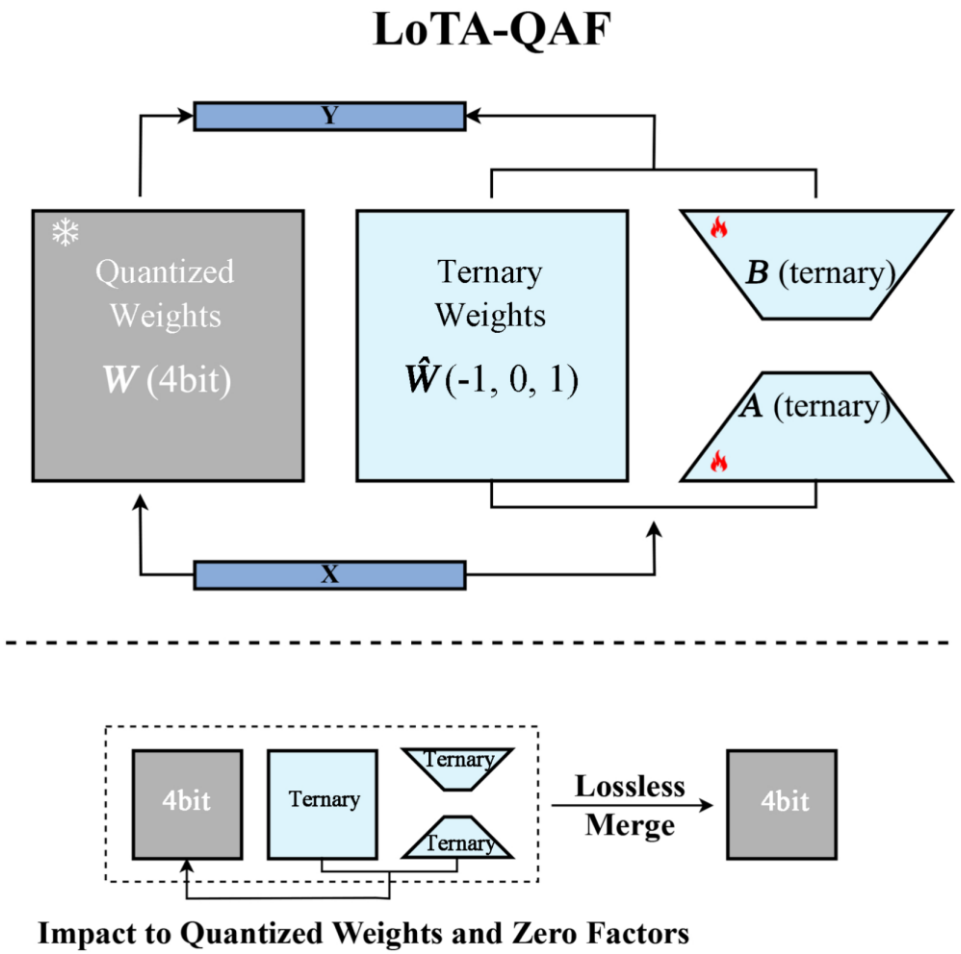
- Propose a ternary adaptation (TA) method that allows **direct adjustment of all quantized weights**.
- Achieve lossless merging of adapters, fully **preserving the effect of fine-tuning** by avoiding merge-induced accuracy loss.

2. Introduce a Novel t-SignSGD Optimizer

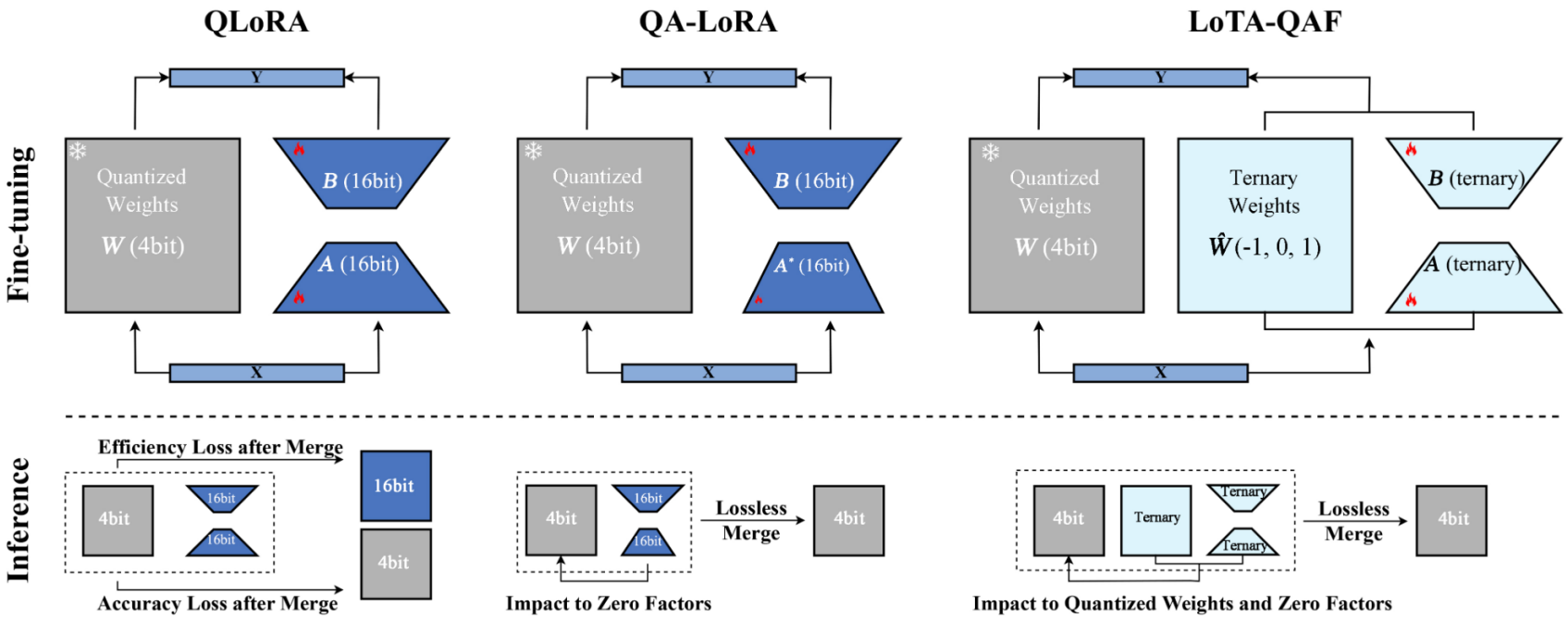
- Develop ternary signed gradient descent (t-SignSGD), **a learning-rate-free** optimizer for highly constrained ternary adapters.
- Leverage **dynamic percentile-based thresholding** to selectively update the most impactful weights.

3. Demonstrate Strong Performance & Efficiency

- 🏆 Performance Boost: Outperforms 16-bit LoRA by up to 5.14% on the MMLU benchmark.
- 🚀 Inference Speedup: Delivers 1.7x-2.0x faster inference than LoRA after merging ternary adapters.



Background



Merging a high-precision adapter (e.g., **16-bit**) with low-precision weights (e.g., **4-bit**) creates a difficult trade-off.

Loss Merge VS **Lossless Merge**

This approach avoids the trade-off between efficiency and accuracy.

- **Efficiency Loss**: The model's weights are de-quantized to **preserve the adapter's precision**, which sacrifices the speed benefits of low-bit inference.
- **Accuracy Loss**: The **adapter's precision is sacrificed** to match the low-bit weights, which can degrade the performance gains from fine-tuning.

- The adapter is designed to be utilized **identically during training and after being merged for inference**.
- LoTA-QAF directly **fine-tuning the quantized weights within their grid**, offering a more powerful fine-tuning capability.

Method || Lossless Ternary Adaptation

Ternary Matrix - $\hat{\mathbf{W}} \in \{-1, 0, 1\}^{D_{\text{in}} \times D_{\text{out}}}$

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

$\mathbf{A}_{\text{T}}$ and $\mathbf{B}_{\text{T}}$: Ternary Adapters (TA)

$\mathbb{I}_{|\Delta \mathbf{W}_{ij}| > \omega}$: Indicator Function

$\Delta \mathbf{W} = \mathbf{A}_{\text{T}} \mathbf{B}_{\text{T}}$: Auxiliary Matrix $\in$ [-r, r]

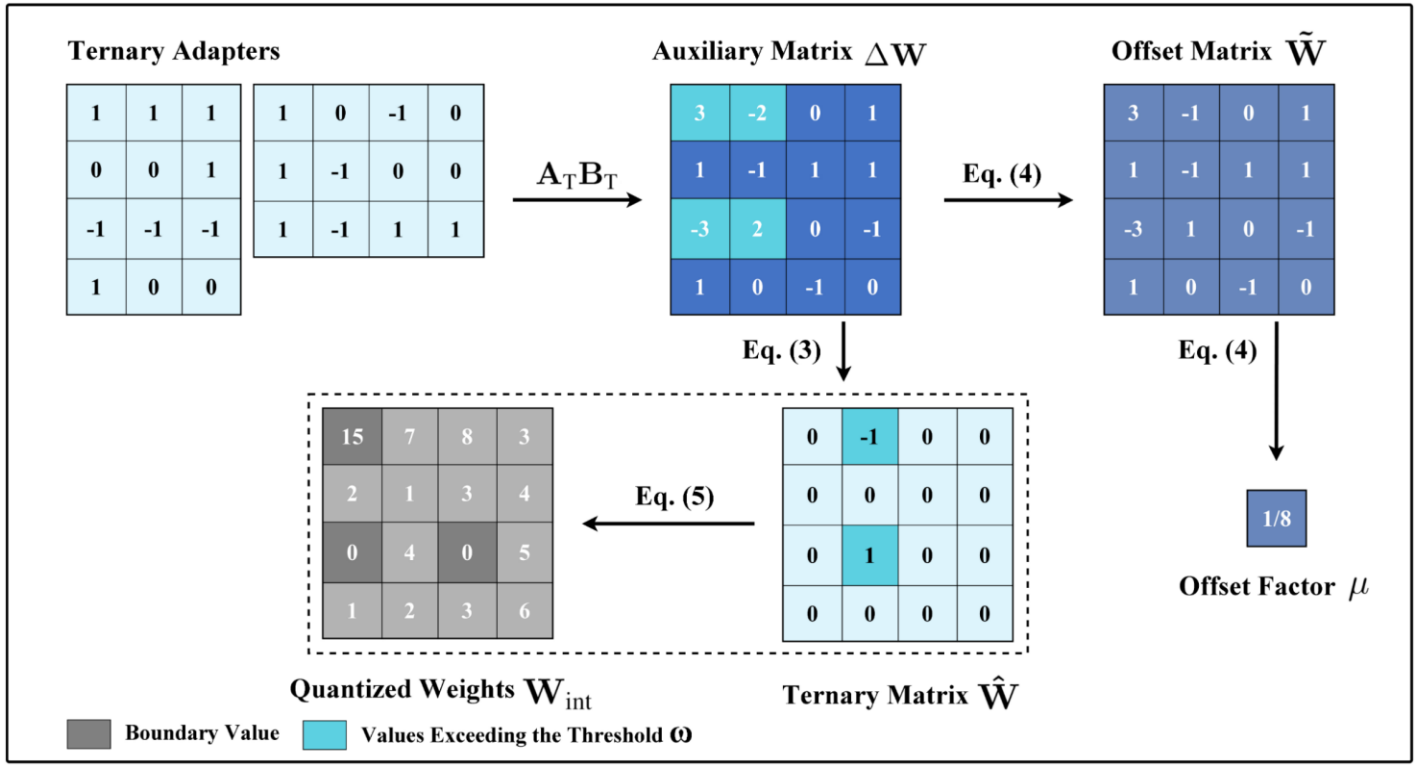
sign(·) : Sign Function

$\omega \in (0, r)$: Controls the impact of the TA on the quantized weights.

Offset Factor - μ

$$\tilde{\mathbf{W}}_{ij} = \Delta \mathbf{W}_{ij} - \omega \hat{\mathbf{W}}_{ij}$$
$$\mu = \frac{\sum_{i=1}^{D_{\text{in}}} \sum_{j=1}^{D_{\text{out}}} \tilde{\mathbf{W}}_{ij}}{D_{\text{in}} D_{\text{out}}} \quad (4)$$

$\tilde{\mathbf{W}}$: Offset Matrix represents the values that remain from $\Delta \mathbf{W}$, after the thresholding operation has been applied.



Lossless Merge

$$\mathbf{W}'_{\text{int}} = \mathbf{W}_{\text{int}} + \hat{\mathbf{W}}$$
$$z' = z + s\mu \quad (5)$$

$\mathbf{W}'_{\text{int}}$ belong to the set {0, 1, . . . , 2^N − 1}. z is Zero Factor, s is Scaling Factor.

TA generates $\hat{\mathbf{W}}$ and μ to enable the lossless merging of adaptation weights and the adjustment of all quantized weights.

Forward Pass

$$\mathbf{y} = (s \cdot \mathbf{W}'_{\text{int}} + z')^T \mathbf{x}$$

During the fine-tuning phase, the Ternary Adaptation (TA) forward pass is equivalent to the merge result. This identical application of the adapter's effect in both phases is what defines it as "lossless."

Low-Rank Adaptation

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

Asymmetric Affine Quantization

$$\mathbf{W}_{\text{q}} = s\mathbf{W}_{\text{int}} + z$$

Method || Ternary Signed Gradient Descent

Update Mechanism

The t-SignSGD update is ***learning-rate-free*** and ***selectively*** modifies weights based on ***salient gradient***.

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

$g_t = \nabla_{\mathbf{A}_\mathrm{T}} \mathcal{L}|_{\mathbf{A}_\mathrm{T}}$ — The gradient of the loss ***L*** with respect to the ternary adapter weights ***AT*** at the current iteration ***t***.

σ and τ — ***A dynamic percentile threshold σ*** (5% -> 0.01% linearly decay) ***A fixed minimum gradient threshold τ*** (e.g., 1 × 10−9)

$\mathbb{I}_{|g_t|>\max(\tau,\sigma_t)}$ — The indicator function acts as a gate. It allows an update to proceed only if the gradient's magnitude

Theoretical Foundation

Core Principle:

The foundation of t-SignSGD lies in SignSGD.

Adaptation for Ternary Space:

This principle is highly relevant for updating our ternary adapters, which are constrained to the discrete values of $\{-1,0,1\}$.

Heuristic Search:

The gradient acts as a high-quality heuristic, transforming the immense combinatorial search space into a manageable, iterative optimization process.

Convergence Properties

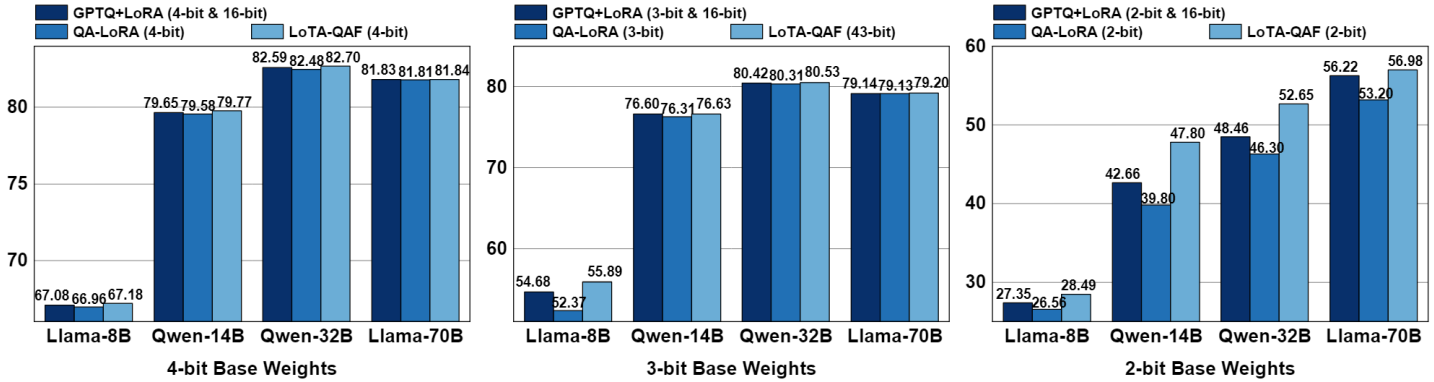
1. Stability via Noise Filtering

- By filtering out small, noisy gradients, the threshold mechanism **prevents unstable parameter "jitter"** (e.g., oscillation between 0 and 1) and promotes a smoother convergence path.

2. Annealing-like Dynamics

- The decaying threshold σ enables a **coarse-to-fine** search strategy.
- Early Training (High σ): Focuses on "broad-stroke" adjustments by updating the most impactful parameters (exploration).
- Later Training (Low σ): Allows for more fine-grained adjustments to refine the solution (exploitation).

Experiments || Main Results



Method	#Bit	MMLU (5-shot)					Task-Specific (0-shot)		
		Hums.	STEM	Social	Other	Avg.	GSM8K	SQL	ViGGO
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

Performance-Recovery Fine-Tuning

Objective: To recover the performance of a quantized model to the level of its 16-bit counterpart.

Result: LoTA-QAF consistently outperforms both LoRA and QA-LoRA in this scenario. On the Qwen 2.5 14B 2-bit model, LoTA-QAF improves performance by 5.14% compared to LoRA.

Insight: LoTA-QAF's superiority is attributed to its ability to directly adjust the **quantized weights within the quantization grid**, effectively recovering performance.

Task-Specific Fine-Tuning

Objective: To enable a quantized model to learn the fine-grained knowledge and patterns of a specific task.

Result: 16-bit LoRA achieves superior results, though LoTA-QAF still outperforms other methods capable of a lossless merge.

Insight: The 16-bit LoRA adapter has a higher representational capacity, which is more effective at capturing complex, task-specific details. While LoRA performs better, it requires computation involving its **16-bit adapter during inference**. LoTA-QAF merges its adapter to **maintain fast, low-bit inference efficiency**.

Code: github.com/Kingdalfgoodman/LoTA-QAF

E-mail: Kingdalfgoodman@foxmail.com

THANKS!