FALQON: Accelerating LoRA Fine-tuning with Low-Bit Floating-Point Arithmetic

Kanghyun Choi, Hyeyoon Lee, SunJong Park, Dain Kwon, Jinho Lee
Department of Electronic and Computer Engineering, Seoul National University

Accelerated Intelligent Systems Lab.

NEURAL INFORMATION PROCESSING SYSTEMS

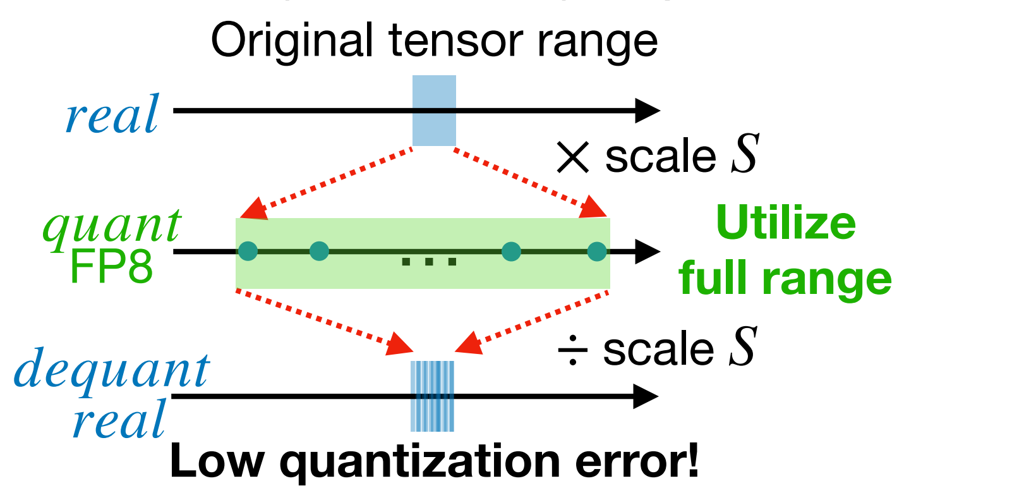
TL; DR: 3× faster quantized LoRA fine-tuning with FP8 by addressing the quantization overhead of LoRA adapter

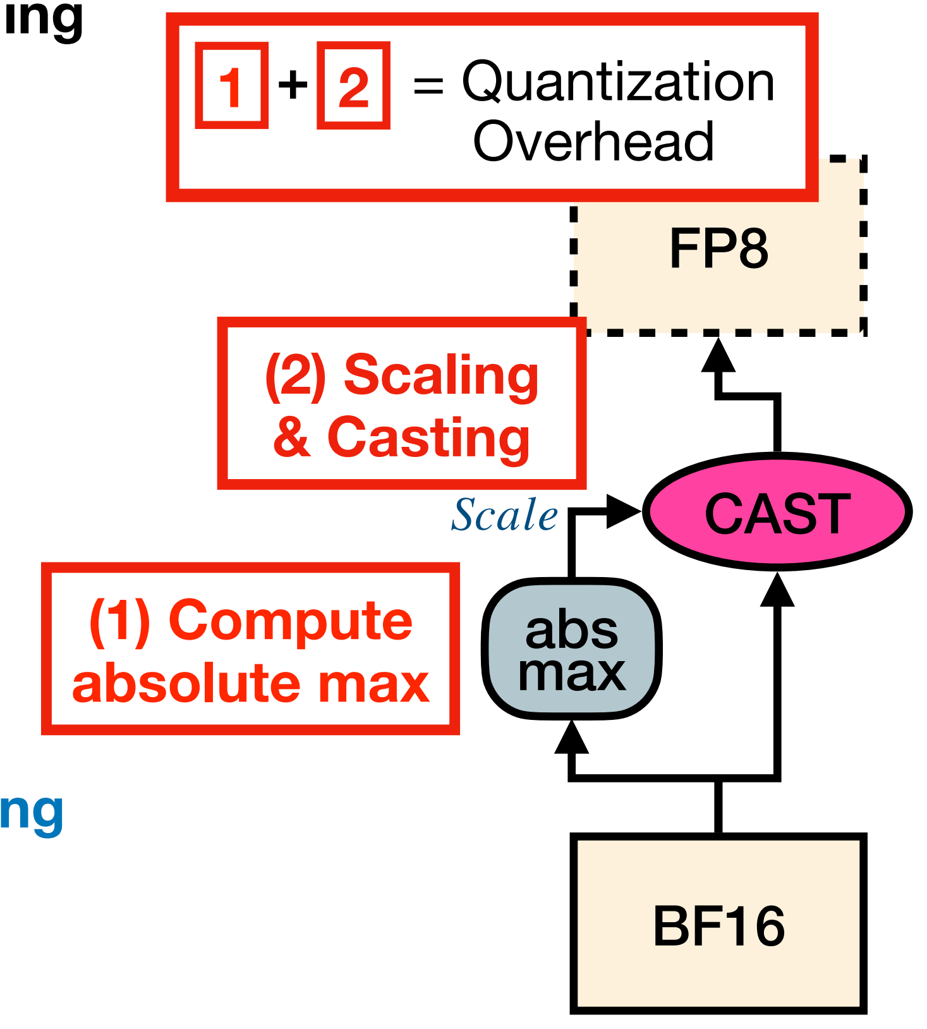
Key Contributions

- We analyze **FP8 quantization overhead** limits speedups when directly applied to LoRA's small-dimensional adapters.
- We propose **FALQON**, a novel framework that merges LoRA adapters into an FP8-quantized backbone during fine-tuning, significantly reducing overhead.
- We **reformulate forward and backward** for efficient gradient computation and **introduce a row-wise proxy update mechanism** that selectively integrates substantial updates.
- **FALQON achieves up to 3× faster fine-tuning** compared to existing methods while maintaining comparable accuracy.

Backgrounds: FP8 Quantization and Overhead

- FP8 quantization (conversion) requires **scaling**



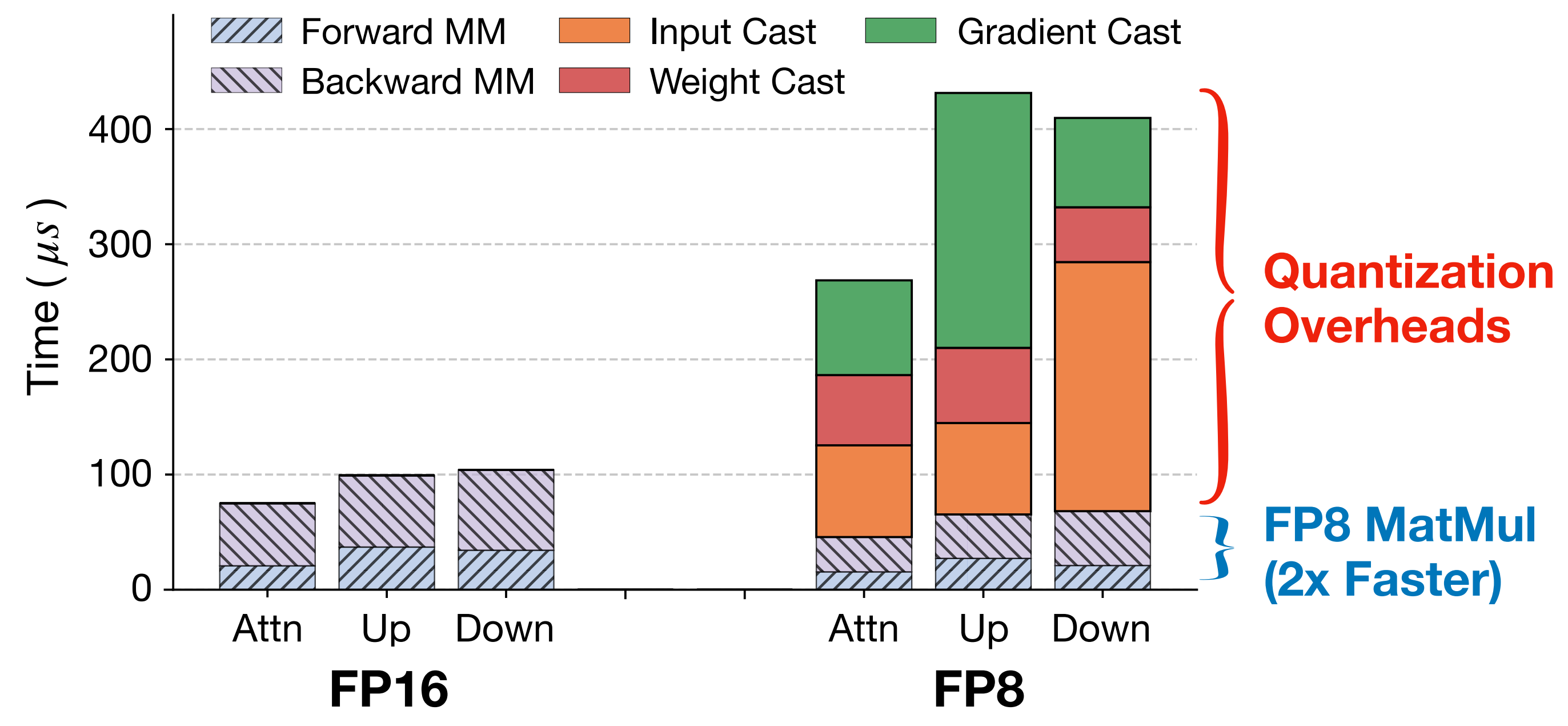


- Scale: absolute max (amax)
- For quantization,
  we need a **reduction** for amax and **scaling**
- For small-dimensional MatMul,
  the overhead exceeds the speed up

Motivational Study

- FP8 quantization overhead of LoRA layers (LLaMA-7B linear dimensions)

Current FP8 framework suffer from quantization overhead on LoRA



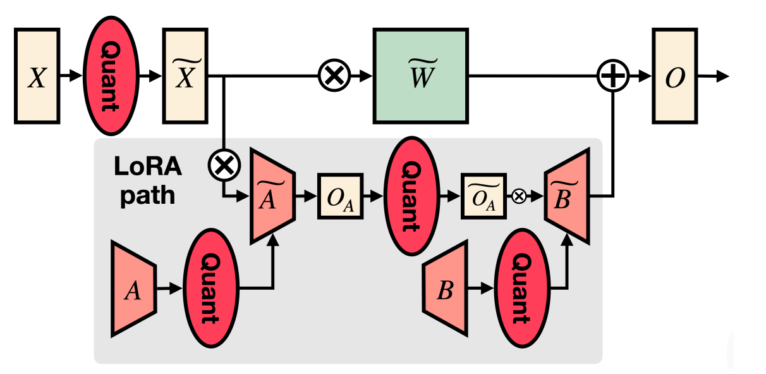
Proposed Method

Key Idea: Merge the LoRA branch into the backbone while training

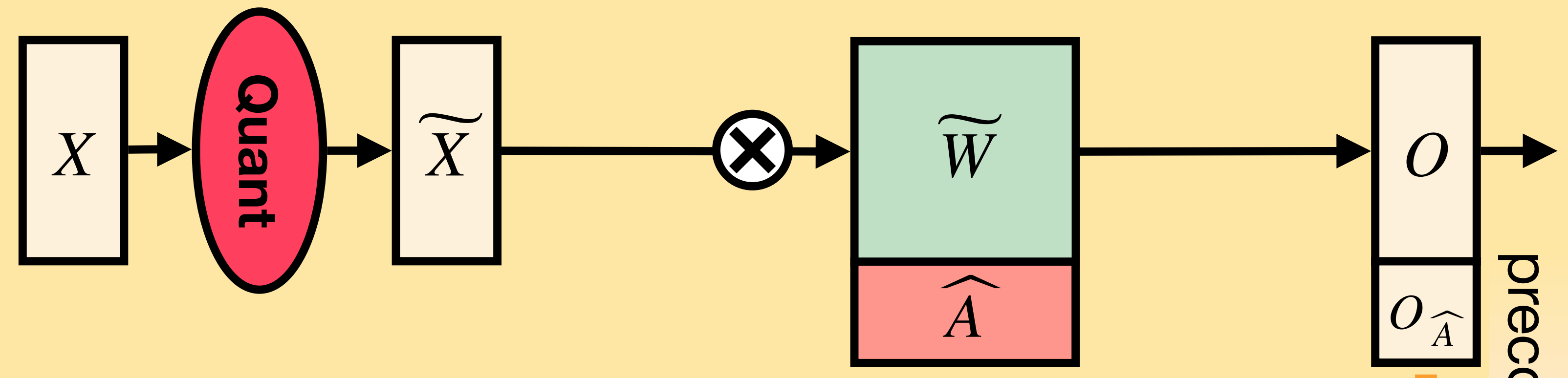
1. Forward

- LoRA Framework: $W_{FT} = W_{orig} + \overbrace{\Delta W}^{\text{weight update}} \approx W_{orig} + \overbrace{BA}^{\text{low-rank projection}}$
- Quantization Error: $\widetilde{W} = \text{Quantize}(W)$
  $\widetilde{W} = W_{orig} + \underbrace{\Delta W_Q}_{\text{quantization error}} \Rightarrow \approx W_{orig} + \underbrace{\widehat{B}\,\widehat{A}}_{\textbf{Quantization Error as LoRA}}$

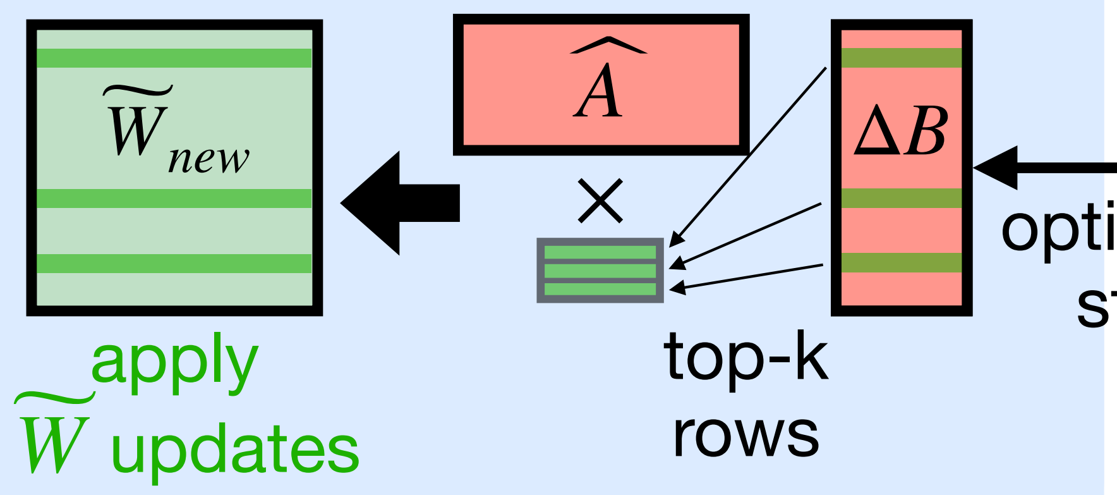
Baseline FP8 LoRA



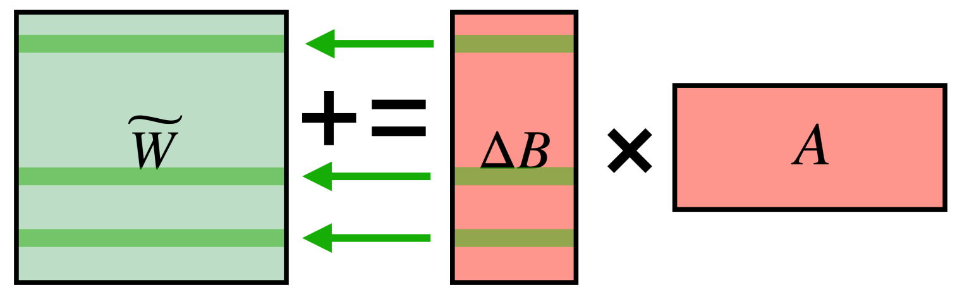
Melded LoRA



3. Update

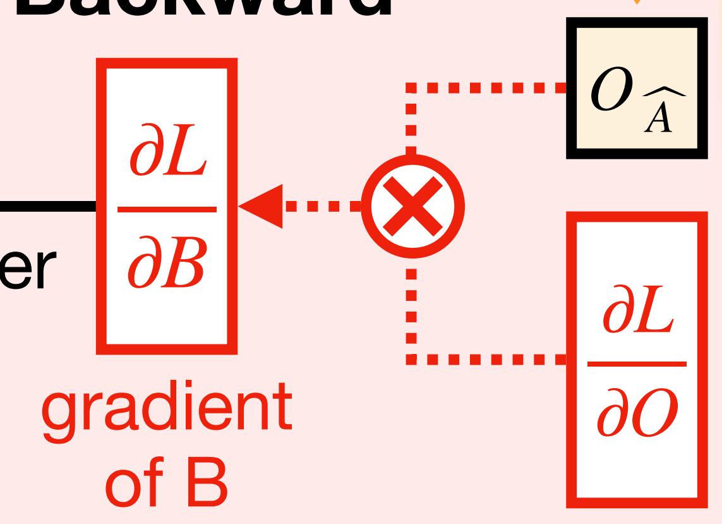


- ΔBuffer:
  - Initialized to all-zero matrix
  - Store updates of B
- **Top-K Row-wise Update**
  - Small updates cannot exceed the quantization grid
  - Find largely updated rows only
- More efficient updates



High-magnitude rows

2. Backward


$O_{\widehat{A}}$
$\frac{\partial L}{\partial B}$
gradient of B
$\frac{\partial L}{\partial O}$

(1) Freeze the matrix A

(2) Compute gradient of matrix B

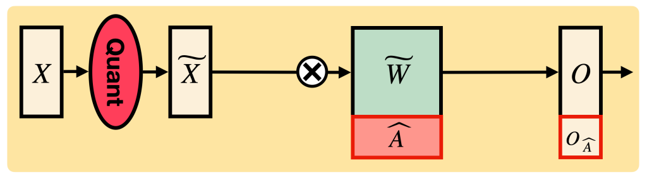
$$\frac{\partial \mathscr{L}}{\partial B} = \frac{\partial \mathscr{L}}{\partial O}(Ax)^{\top}$$

(2)-1 Merge matrix A to W:

$$\widetilde{W}' = \begin{bmatrix} \widetilde{W} \\ \widetilde{A} \end{bmatrix} \in \mathbb{R}^{(m+r)\times n}$$

(2)-2 Precompute Ax in forward:

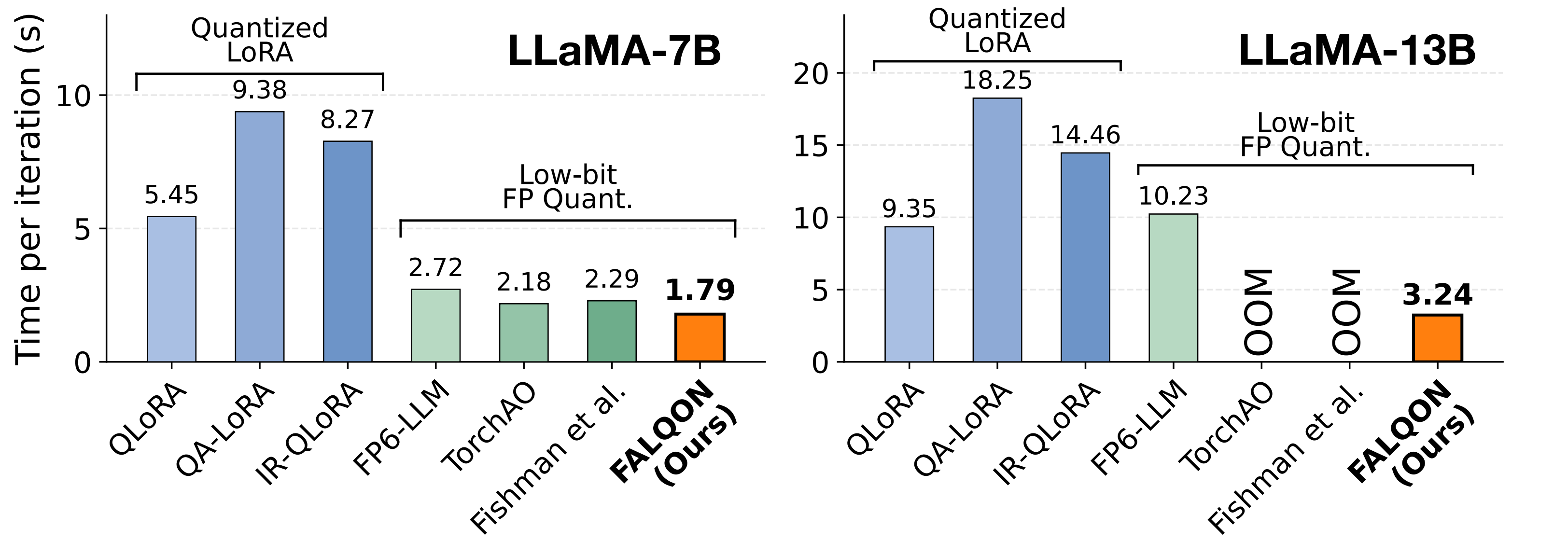
$$\widetilde{W}'\widetilde{x} = \begin{bmatrix} O \\ Ax \end{bmatrix} \in \mathbb{R}^{(m+r)\times d}$$



Precompute Ax

Experimental Results

Overall Computational Cost Comparison



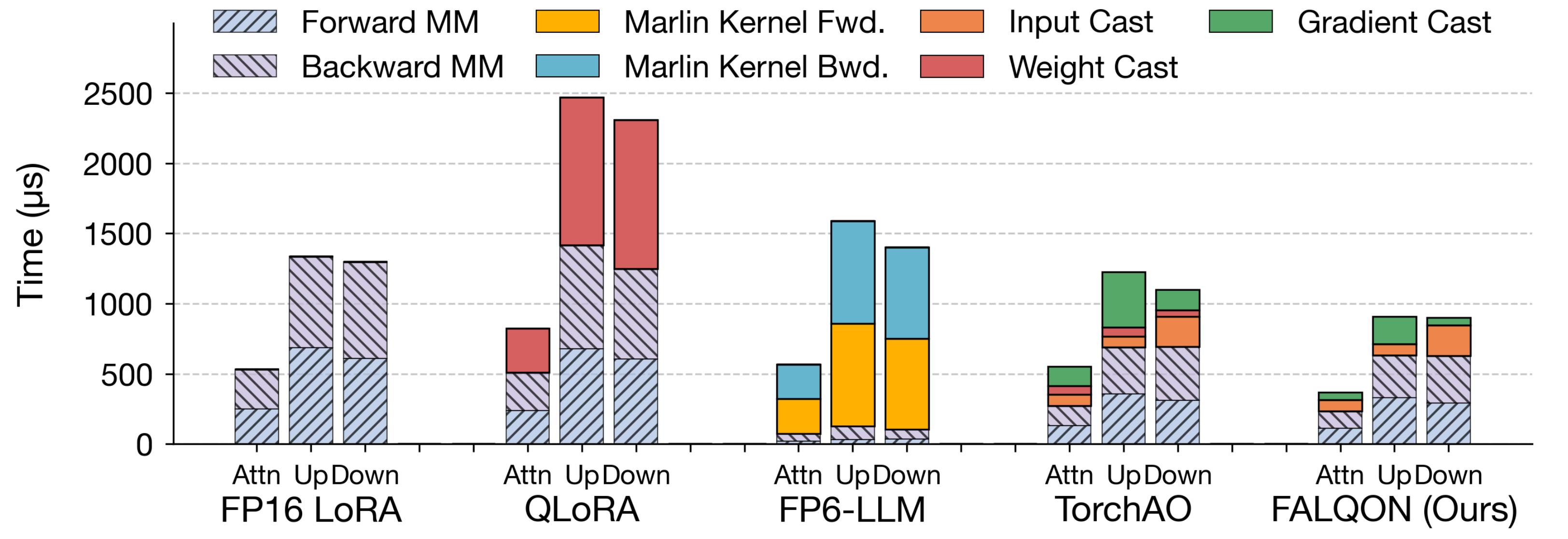
Fine-tuning Quality Comparison of Quantized LoRA (5-shot MMLU)

	Data: Alpaca	QLoRA	QA-LoRA	IR-QLoRA	FALQON (Ours)
7B	Time / Step (s)	5.45	9.44	8.27	**1.80 (3.02×)**
	#T. Params.	160M	89M	89M	80M
	MMLU Acc.	0.3272	0.3548	0.3388	0.3491
13B	Time / Step (s)	9.37	18.02	14.46	**3.26 (2.87×)**
	#T. Params.	250M	140M	140M	125M
	MMLU Acc.	0.4443	0.4729	0.4349	0.4644

	Data: OASST1	QLoRA	QA-LoRA	IR-QLoRA	FALQON (Ours)
7B	Time / Step (s)	5.45	9.38	8.34	**1.79 (3.04×)**
	#T. Params.	160M	89M	89M	80M
	MMLU Acc.	0.3564	0.3609	0.3605	0.3481
13B	Time / Step (s)	9.35	18.25	15.22	**3.24 (2.89×)**
	#T. Params.	250M	140M	140M	125M
	MMLU Acc.	0.4605	0.4769	0.4620	0.4645

Method	Type	Time / Step (s)	# Trainable Params	Alpaca (MMLU)					OASST1 (MMLU)				
				Hum.	STEM	Social	Other	Avg.	Hum.	STEM	Social	Other	Avg.
LoRA	FP16	2.87	160M	0.3295	0.3031	0.3717	0.3873	0.3456	0.3401	0.3258	0.4006	0.4102	0.3656
TorchAO	FP8	2.18	160M	0.3231	0.2969	0.3679	0.3785	0.3393	0.3273	0.3092	0.3672	0.3869	0.3452
FP6-LLM	E2M3	2.72	160M	0.2421	0.2125	0.2171	0.2398	0.2295	0.2448	0.2125	0.2177	0.2411	0.2308
FP6-LLM	E3M2	2.72	160M	0.2487	0.2693	0.2532	0.2333	0.2509	0.2423	0.2249	0.2190	0.2411	0.2330
Fishman *et al.*	FP8	2.29	160M	0.3337	0.3108	0.3893	0.3923	0.3537	0.3241	0.2969	0.3773	0.3714	0.3401
FALQON (Ours)	FP8	**1.79**	80M	0.3322	0.3086	0.3858	0.3795	0.3491	0.3373	0.3130	0.3776	0.3708	0.3481

Breakdown Analysis of LoRA Fine-tuning



Training Time and Monetary Cost on Cloud GPU Platforms

Device	Training Time (days, 8 GPUs)			Training Cost ($ USD)			Cost Reduction ($ USD)	
	QLoRA	QA-LoRA	FALQON	QLoRA	QA-LoRA	FALQON	vs QLoRA	vs QA-LoRA
RTX 4090	89.3	153.7	35.7	6,001	10,328	1,971	⇓4,030	⇓8,357
L40S	98.3	164.0	37.7	35,126	58,603	10,070	⇓25,057	⇓48,533
H100	31.1	25.1	13.3	41,122	33,114	13,419	⇓27,703	⇓19,695