Quaff: Quantized Parameter-Efficient Fine-Tuning under Outlier Spatial Stability Hypothesis

Hong Huang
Department of Computer Science
City University of Hong Kong
Hong Kong, China
honghuang2000@outlook.com

Dapeng Wu
Department of Computer Science
City University of Hong Kong
Hong Kong, China
dpwu@ieee.org

Abstract

Large language models (LLMs) have made exciting achievements across various domains, yet their deployment on resource-constrained personal devices remains hindered by the prohibitive computational and memory demands of task-specific fine-tuning. While quantization offers a pathway to efficiency, existing methods struggle to balance performance and overhead, either incurring high computational/memory costs or failing to address activation outliers, a critical bottleneck in quantized fine-tuning. To address these challenges, we propose the Outlier Spatial Stability Hypothesis (**OSSH**): *During fine-tuning, certain activation outlier channels retain stable spatial positions across training iterations.* Building on OSSH, we propose **Quaff**, a Quantized parameter-efficient fine-tuning framework for LLMs, optimizing low-precision activation representations through targeted momentum scaling. Quaff dynamically suppresses outliers exclusively in invariant channels using lightweight operations, eliminating full-precision weight storage and global rescaling while reducing quantization errors. Extensive experiments across ten benchmarks validate OSSH and demonstrate Quaff's efficacy. Specifically, on the GPQA reasoning benchmark, Quaff achieves a 1.73× latency reduction and 30% memory savings over full-precision fine-tuning while improving accuracy by 0.6% on the Phi-3 model, reconciling the triple trade-off between efficiency, performance, and deployability. By enabling consumer-grade GPU fine-tuning (*e.g.,* RTX 2080 Super) without sacrificing model utility, Quaff democratizes personalized LLM deployment. The code is available at https://github.com/Little0o0/Quaff.git.

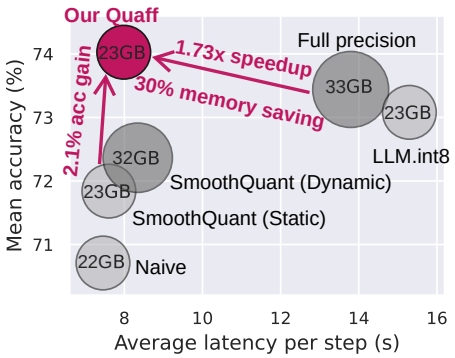
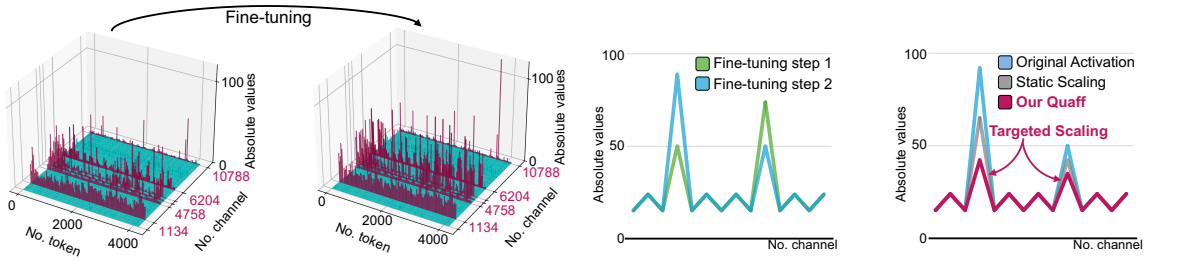


Figure 1: Comparison of accuracy, average latency per step of WAQ baselines with Phi-3 on the GPQA benchmark using LoRA fine-tuning. The size of the marker represents the GPU memory footprints.

1 Introduction

Large language models (LLMs) (Wu et al., 2023a; Floridi and Chiriatti, 2020; Zhang et al., 2022) exhibit remarkable achievement in various domains, from creative writing to conversational chatbots. Growing demands for privacy-preserving, personalized LLMs (*e.g.,* a virtual companion chatbot) deployed on local devices clash with the prohibitive computational and memory costs of fine-tuning, building a critical barrier for individuals and small enterprises. While parameter-efficient fine-tuning (PEFT) reduces trainable parameters, it still imposes unsustainable overhead when scaling to billion-parameter models.

Quantization (Dettmers et al., 2021, 2022; Lin et al., 2023; Frantar et al., 2022) offers a pathway to efficiency, but most of existing work focuses narrowly on weight-only quantization (WOQ) in fine-tuning (Kwon et al., 2022; Dettmers et al., 2024; Xu et al., 2023; Li et al., 2023; Liu et al., 2023; Guo et al., 2023; Kim et al., 2024; He et al., 2023; Lee et al., 2024a; Frantar et al., 2022; Lin et al., 2023), which compresses frozen weights to low precision (*e.g.,* 4-bit). However, WOQ introduces computational bottlenecks via hardware-unfriendly mixed-precision operations between quantized weights and full-precision activations. Weight-Activation Quantization (WAQ) (Zhou et al., 2016; Dettmers et al., 2021; Hubara et al., 2018) addresses this



(a) Activation distribution of LLaMA2-7B during fine-tuning **The outlier channels keep in stable indices**

(b) Maximum activation per channels **Outlier magnitudes change significantly**

(c) Scaled activation per channels **Quaff successfully suppress outliers**

Figure 2: (a) Spatial stability of outlier channels: activation magnitude distribution during fine-tuning, demonstrating stable spatial indices for outlier channels across iterations. (b) Activation distribution shift: the magnitudes of outliers change significantly during fine-tuning. (c) Quaff Efficacy: Failure of static scaling due to outlier fluctuations, contrasted with Quaff's targeted momentum scaling on stable outlier channels, successfully suppressing outliers.

by quantizing both weights and activations into hardware-friendly formats (*e.g.,* INT8), but faces a fundamental challenge in LLMs, **emergent channel-wise outliers** (Wei et al., 2022; Wu et al., 2023b; Yao et al., 2022; Dettmers et al., 2022; Xiao et al., 2023; Huang et al., 2024b), where large activations amplify quantization errors and degrade model performance.

Prior methods suppress outliers in *inference* via channel-wise scaling between activations and weights (Dettmers et al., 2022; Wei et al., 2022; Xiao et al., 2023; Wei et al., 2023) But these methods fail to adapt to *fine-tuning* due to activation distribution shifts (Fig. 2 (b)): 1. Static scaling (Xiao et al., 2023) predefines factors on calibration data, but mismatched scaling amplifies quantization errors as distributions evolve. While some methods (Lin et al., 2025; Huang et al., 2024b; Ashkboos et al., 2024) attempt to mitigate this by replacing scaling with rotation, they introduce architectural rigidity and computational Inefficiency. 2. Dynamic scaling (Dettmers et al., 2022; Xiao et al., 2023) adapts factors in real time but requires storing and rescaling of full-precision weights, incurring prohibitive memory/compute costs.

We find the key problem in previous approaches is **coupling bottleneck**: scaled weight quantization depends on real-time activation statistics, preventing hardware-friendly deployment. To address this, we first introduce the Outlier Spatial Stability Hypothesis (**OSSH**): *During fine-tuning, certain activation outlier channels retain stable spatial position across training iterations.* Building on OSSH, we propose **Quaff**, a **Qua**ntized parameter-e**ff**icient **f**ine-tuning framework for LLMs that decouples weight and activation quantization via targeted momentum scaling. Quaff dynamically computing scaling factors exclusively for invariant outlier channels, eliminating full-precision weight storage and global rescaling, enabling low quantization error and high efficiency.

We evaluate Quaff across ten benchmarks, including reasoning (MMLU-Pro (Wang et al., 2024b), GPQA (Rein et al., 2023), instruction-tuning(Alpaca-Finance (Bharti, 2023), Self-instruct (Wang et al., 2022)) and long text task (LongForm (Köksal et al., 2023), LAMBADA (Paperno et al., 2016)), using LLaMA-2 (Touvron et al., 2023), Phi-3 (Abdin et al., 2024), and OPT (Zhang et al., 2022) models with four general PEFT methods (LoRA(Hu et al., 2021), Prompt tuning(Lester et al., 2021), P-tuning(Liu et al., 2021) and IA3(Liu et al., 2022)). Our experiment also evaluates the deployability of Quaff on a consumer-grade laptop with RTX 2080 super GPU (8GB). Extensive experimental results suggest that Quaff achieves the best trade-offs: it achieves 1.73× faster fine-tuning and 30% memory reduction versus full-precision baselines while improving accuracy by 0.6% on GPQA. Against state-of-the-art (SOTA) WAQ methods, Quaff delivers a 2.1% accuracy gain under identical constraints (Fig. 1), validating its ability to harmonize efficiency (latency/memory), accuracy, and deployability. *To our knowledge, Quaff is the first work to systematically resolve the triple trade-off in WAQ LLM fine-tuning, enabling local-device fine-tuning as effortlessly as a toast.*

2 Background and Challenge

This section introduces foundational concepts of neural network quantization, analyzes the unique challenges of weight-activation quantiza-

tion (WAQ) in LLMs, and identifies fundamental limitations in existing approaches for fine-tuning.

2.1 Quantization Fundamentals

Quantization reduces numerical precision in neural networks by converting weights and activations from high-bit formats (*e.g.,* FP32) to low-bit representations (*e.g.,* INT8), optimizing memory usage and computational efficiency. The standard symmetric round-to-nearest quantization (Jacob et al., 2018) maps a floating-point matrix $\mathbf{X}$ into an N-bit integer matrix $\mathbf{X}_{int}$:

$$\mathbf{X}_{int} = Q(\mathbf{X}) = \left\lfloor \frac{\mathbf{X}}{\Delta_{\mathbf{X}}} \right\rceil, \quad \Delta_{\mathbf{X}} = \frac{\max(|\mathbf{X}|)}{2^{N-1}-1}, \tag{1}$$

where $\lfloor \cdot \rceil$ denotes rounding function, and $\Delta_{\mathbf{X}}$ is quantization step size. The granularity of $\Delta_{\mathbf{X}}$ (scalar, vector, or matrix) determines how finely quantization is applied (see Appendix F).

2.2 Weight-Activation Quantization in LLMs

Weight-activation quantization (WAQ) compresses both weights $\mathbf{W} \in \mathbb{R}^{c_{in} \times c_{out}}$ and activations $\mathbf{X} \in \mathbb{R}^{t \times c_{in}}$, where t is the number of tokens, c_{in} is the number of input channels, and c_{out} is the number of output channels, to accelerate matrix multiplication $\mathbf{Y} = \mathbf{X}\mathbf{W}$:

$$\mathbf{Y} \approx \Delta_{\mathbf{X}} \cdot (\mathbf{X}_{int}\mathbf{W}_{int}) \cdot \Delta_{\mathbf{W}}, \tag{2}$$

where INT8 integer operations reduce compute costs by $4\times$ by the integer kernel. However, LLMs exhibit **emergent channel-wise outliers** (Fig. 2) with magnitudes $100\times$ larger than typical activations (Xiao et al., 2023), inflating $\Delta_{\mathbf{X}}$ and causing catastrophic quantization errors.

Prior work addresses outliers via channel-wise scaling (Dettmers et al., 2022; Wei et al., 2022; Xiao et al., 2023; Shao et al., 2023) by using:

$$\begin{aligned} \mathbf{Y} &= (\mathbf{X}\mathbf{s}^{-1})(\mathbf{s}\mathbf{W}) = \hat{\mathbf{X}}\hat{\mathbf{W}} \\ &\approx \Delta_{\hat{\mathbf{X}}} \cdot (\hat{\mathbf{X}}_{int}\hat{\mathbf{W}}_{int}) \cdot \Delta_{\hat{\mathbf{W}}}, \end{aligned} \tag{3}$$

where $\hat{\mathbf{X}} = \mathbf{X}\mathbf{s}^{-1}$ denotes scaled activations, $\hat{\mathbf{W}} = \mathbf{s}\mathbf{W}$ denotes scaled weights[1]. The channel-wise factors $\mathbf{s} \in \mathbb{R}^{c_{in}}$ determined by both $\mathbf{W}$ and $\mathbf{X}$ suppresses outliers in activation $\mathbf{X}$ by redistributing them to $\mathbf{W}$. While effective for *inference*, this approach falls short during *fine-tuning*.

[1]In this paper, we denote the multiplication between vector $\mathbf{s}$ and matrix $\mathbf{X}$ as element-wise, *i.e.*, $[\mathbf{s}\mathbf{W}]_{i,j} = \mathbf{s}_i\mathbf{W}_{i,j}$.

2.3 Challenges

Channel-wise scaling creates a **coupling bottleneck** between weight and activation quantization: the scaled weights $\hat{\mathbf{W}} = \mathbf{s}\mathbf{W}$ become dependent on real-time activations through scaling factors $\mathbf{s}$.

This coupling manifests in two failure modes: 1. **Static scaling** predefines $\mathbf{s}$ on calibration data and ignores activation distribution shifts during fine-tuning, leading to mismatched scaling factors (Fig. 10) that amplify quantization errors in $\hat{\mathbf{X}}$ and $\hat{\mathbf{W}}$. Some methods(Lin et al., 2025; Huang et al., 2024b; Ashkboos et al., 2024) attempt to address this by replacing scaling with rotation, setting $\mathbf{s}$ as a rotation matrix; however, they introduce computational inefficiency in computing $\hat{\mathbf{X}}$. 2. **Dynamic scaling** adapts $\mathbf{s}$ to activation distributions during fine-tuning, forcing repeated updating and requantization of $\hat{\mathbf{W}}$, requiring full-precision weight storage and dynamic recomputation, incurring unacceptable computational/memory costs.

Existing methods (Dettmers et al., 2022; Wei et al., 2022; Xiao et al., 2023; Lin et al., 2025) thus face a trilemma: preserving performance via dynamic scaling sacrifices efficiency; prioritizing efficiency and deployability via static scaling sacrifices adaptability. This fundamental limitation underscores the need for a decoupled quantization framework that preserves performance while enabling hardware-efficient fine-tuning.

3 Methodology

This section introduces the theoretical foundation of our approach, formalizes the Outlier Spatial Stability Hypothesis (OSSH), and presents the Quaff framework for efficient quantized fine-tuning.

3.1 Motivation: Decoupling WAQ

The core limitation of prior methods lies in the interdependence between scaled weights ($\hat{\mathbf{W}} = \mathbf{s}\mathbf{W}$) and activation quantization. To decouple this dependency, we reformulate channel-wise scaling in Eq. 3 as:

$$\begin{aligned} \mathbf{Y} &= \hat{\mathbf{X}}(\mathbf{s}\mathbf{W}) = \hat{\mathbf{X}}\left(\mathbf{W} + (\mathbf{s}-\mathbf{1})\mathbf{W}\right) \\ &= \hat{\mathbf{X}}\underbrace{\mathbf{W}}_{\text{Static}} + \hat{\mathbf{X}}\underbrace{(\mathbf{s}-\mathbf{1})\mathbf{W}}_{\text{Dynamic}}, \end{aligned} \tag{4}$$

isolating frozen, pre-quantizable weights $\mathbf{W}$ from the dynamic term $(\mathbf{s}-\mathbf{1})\mathbf{W}$. Critically, for non-outlier channels i, activations do not need to be scaled, *i.e.,* $\mathbf{s}_i = 1$, rendering $(\mathbf{s}-\mathbf{1})$ highly sparse.

Letting O denote the set of outlier channel indices, we simplify Eq. 4 to:

$$\begin{aligned}\mathbf{Y} &= \hat{\mathbf{X}}\mathbf{W} + \hat{\mathbf{X}}_{:,O}(\mathbf{s}_O - \mathbf{1})\mathbf{W}_O \\ &= \hat{\mathbf{X}}\mathbf{W} + \hat{\mathbf{x}}\hat{\mathbf{w}},\end{aligned} \tag{5}$$

where $\hat{\mathbf{x}} = \hat{\mathbf{X}}_{:,O}$ and $\hat{\mathbf{w}} = (\mathbf{s}_O - \mathbf{1})\mathbf{W}_O$ represent the submatrix of scaled activations and weights in outlier channels.[2] Notably, if outlier channels O remain invariant during fine-tuning, storing only the small static submatrix $\mathbf{W}_O$ in full precision suffices to compute $\hat{\mathbf{w}}$ in real-time, avoiding costly dequantization of $Q(\mathbf{W})$, enabling hardware-efficient fine-tuning. Therefore, we propose OSSH to validate the invariant outlier channels.

3.2 Outlier Spatial Stability Hypothesis

Building on empirical observations of channel-wise outliers in LLM inference (Dettmers et al., 2022; Wei et al., 2022; Xiao et al., 2023), we enhance its spatial stability and extend it to the fine-tuning regime, formalizing this as the **Outlier Spatial Stability Hypothesis (OSSH)**:

During fine-tuning, certain activation outlier channels retain stable spatial positions across training iterations.

Unlike prior methods (Lee et al., 2024b; Xiao et al., 2023) that assume activation stability based on empirical observations during inference (where the model is fixed), OSSH investigates outlier stability during fine-tuning, where activations undergo distributional shifts. This stability emerges from the interaction between (1) the preservation of foundational pre-trained features critical for cross-task generalization (Mosbach et al., 2020) and (2) semantic consistency for salient tokens (e.g., [CLS]) (Fu et al., 2023; Hewitt and Manning, 2019). Outlier channels serve as anchors for high-level semantic primitives, bridging the model's general knowledge and task-specific adaptations; their stability arises naturally because they consistently encode salient activations, ensuring robustness during fine-tuning.

OSSH enables the pre-identification of static outlier channels O prior to fine-tuning, eliminating resource-intensive runtime detection overhead. The validation in Fig. 3 further supports the OSSH, where with 5% predefined outlier channels, it can reach $> 90\%$ overall hit rate across fine-tuning iterations. More analysis of OSSH is in Sec. B.

[2]In this paper, we use the $\mathbf{X}_{:,O}$ to denote the submatrix containing O columns of matrix $\mathbf{X} \in \mathbb{R}^{t \times c_{in}}$, *i.e.,* $\mathbf{X}_{:,O} = (\mathbf{X}_{1,O}, \ldots, \mathbf{X}_{t,O})^T$, where $\mathbf{X}_{i,O} = (\mathbf{X}_{i,j} | j \in O)$.

Figure 3: The average hit rate of real-time versus predefined outlier channel indices across each layer of Phi3-3.8B during fine-tuning on OIG/Chip2. The shaded area represents the standard deviation.

3.3 Proposed Quaff

Building on OSSH, we introduce a brand new quantized parameter-efficient fine-tuning algorithm, Quaff, decoupling the dependency between weight and activation quantization by targeted momentum scaling, where scaling factors are targeted computed in spatially invariant outlier channels with a momentum mechanism, optimizing efficiency without sacrificing performance.

Quaff begins with weights preprocessing before task-specific fine-tuning. First, Quaff uses a calibration dataset to identify outlier channels O. The outlier detection strategy can vary. For example, the criterion ξ_o used to determine if a channel o is an outlier channel can be defined as

$$\xi_o = \sum_i \mathbf{1}_{\max(|\mathbf{X}^i_{:,o}|) > 100 \cdot \text{mean}(|\mathbf{X}^i|)}, \tag{6}$$

where $\mathbf{X}^i$ denotes the activations from the i-th sample in the calibration datasets. Therefore, the outlier channel O can be obtained by selecting outlier channels o based on ξ_o. After that, Frozen weights $\mathbf{W}$ are quantized to $\mathbf{W}_{int}$ and $\Delta_{\mathbf{W}}$ as in Eq. 1, while retaining full-precision of outlier channel weights $\mathbf{W}_O$. Based on empirical observations, we limit the overall overhead for $\mathbf{W}_O$ to less than 5% by controlling the size of O. It should be noted that this 5% budget is not uniformly distributed across all layers. As shown in previous work (Lin et al., 2025), certain layers, such as q_proj, contain few outlier channels, whereas others, like $down\_proj$, have a higher proportion of outlier channels. To accommodate this variance, we reallocate the budget from layers with fewer outliers, like q_proj, to

those with more, such as $down\_proj$, ensuring the total overhead remains below 5%.

After weights preprocessing, Quaff injects learnable task-specific parameters θ (*e.g.,* LoRA adapters) for fine-tuning. During the fine-tuning, a targeted momentum scaling mechanism is employed to stabilize outlier suppression. At the t-th step, scaling factors $\mathbf{s}_t$ blend historical values with current observations:

$$\mathbf{s}_t = \gamma \mathbf{s}_{t-1} + (1-\gamma)\beta, \tag{7}$$

where $\gamma \in [0,1]$ is the hyperparameters to control update inertia, and the $\beta \in \mathbb{R}^{c_{in}}$ is defined as:

$$\beta_i = \begin{cases} 1, & i \notin O \\ \max\left(1, \sqrt{\frac{\max(|\mathbf{X}_{:,i}|)}{\max(|\mathbf{W}_i|)}}\right), & i \in O \end{cases}. \tag{8}$$

This formulation prevents overreaction to transient activation shifts while maintaining compatibility with quantized weights. Then, the Quaff obtains the scaled activation $\hat{\mathbf{X}}$ and scaled outlier weights $\hat{\mathbf{w}}$ by Eq. 5. After that, using the uniform quantization as in Eq. 1,The forward pass in Eq. 5 approximates:

$$\begin{aligned} \mathbf{Y} &\approx \Delta_{\hat{\mathbf{X}}} \hat{\mathbf{X}}_{int} \mathbf{W}_{int} \Delta_{\mathbf{W}} + \Delta_{\hat{\mathbf{x}}} \hat{\mathbf{x}}_{int} \hat{\mathbf{w}}_{int} \Delta_{\hat{\mathbf{w}}} \\ &= \Delta_{\hat{\mathbf{X}}} (\hat{\mathbf{X}}_{int} \mathbf{W}_{int} \Delta_{\mathbf{W}} + \hat{\mathbf{x}}_{int} \hat{\mathbf{w}}_{int} \Delta_{\hat{\mathbf{w}}}), \end{aligned} \tag{9}$$

where $\Delta_{\hat{\mathbf{X}}} = \Delta_{\hat{\mathbf{x}}}$ and $\hat{\mathbf{x}}_{int} = [\hat{\mathbf{X}}_{int}]_{:,O}$ inherit activation quantization without overhead.

Compared to prior work, Quaff successfully addresses the trilemma of efficiency, performance, and deployability. **First**, Quaff reduces recomputation and memory overheads in scaling by 99% compared to dynamic scaling methods (Xiao et al., 2023; Dettmers et al., 2022) by targeted operations on the outlier channels O. Compared to naive WAQ in Eq. 2, Quaff incurs less than 5% overhead (storing $\mathbf{W}_O$ and computing $\hat{\mathbf{x}}_{int}\hat{\mathbf{w}}_{int}\boldsymbol{\Delta}_{\hat{\mathbf{w}}}$), while significantly reducing quantization error through outlier suppression. **Second**, the momentum-based scaling mechanism further stabilizes fine-tuning by smoothing out transient fluctuations in activation, prioritizing persistent distributional patterns for robust scaling. What's more, the $(\mathbf{s}-1)$ scaling terms reduce weight sensitivity relative to direct $\mathbf{s}$ scaling, enhancing quantization stability. **Lastly**, and most importantly, the efficiency of Quaff enables deployment of fine-tuning on edge devices (*e.g.,* RTX 2080 Super) via a server-client paradigm: public servers preprocess and distribute quantized model weights $\mathbf{W}_{int}$ and outlier weights $\mathbf{W}_O$, while clients perform personalized quantized fine-tuning without storing full-precision weights.

4 Evaluation

In this section, we comprehensively evaluate Quaff by comparing it against SOTA quantization approaches across ten benchmarks. All experiments are repeated five times, with means and standard deviations reported. In the table, the best and second-best results are highlighted in purple and blue, respectively.

4.1 Experimental Setup

Datasets settings. To ensure broad applicability, we evaluate the effectiveness of the proposed Quaff on diverse datasets spanning various downstream domains. Specifically, we use five instruction-tuning datasets (Alpaca-Finance (Bharti, 2023), HH-RLHF (Bai et al., 2022), Self-instruct (Wang et al., 2022), OIG/Chip2 (LAION, 2023), and Oasst1 (Köpf et al., 2024)), three reasoning datasets (GPQA (Rein et al., 2023), MathQA (Amini et al., 2019), and MMLU-Pro (Wang et al., 2024b)), and two long text datasets (Longform (Köksal et al., 2023) and LAMBADA (Paperno et al., 2016)).

The selection of these datasets is motivated by the need for a comprehensive evaluation, particularly given that existing LLMs (Touvron et al., 2023; Abdin et al., 2024) and quantization methods (Touvron et al., 2023; Xiao et al., 2023; Dettmers et al., 2024, 2022) have primarily focused on these benchmarks. For datasets without predefined training and testing splits, we randomly sample 80% of the data for training and allocate the remaining 20% for testing. Details for benchmarks are in Sec. E.

Model and Training settings. We evaluate Quaff on LLaMA2 (Touvron et al., 2023), Phi3 (Abdin et al., 2024), and OPT (Zhang et al., 2022) using four popular PEFT methods: LoRA (Hu et al., 2021), Prompt (Lester et al., 2021), P-tuning (Liu et al., 2021) and IA3 (Liu et al., 2022). The default fine-tuning batch size is set to 16. We set a quantization precision of INT8 to ensure broad hardware compatibility. The models are maintained at 32-bit floating-point (FP32) precision by default. We opted for FP32 instead of FP16 because many LLMs (*e.g.,* LLaMA(Paperno et al., 2016)) only support Brain Floating Point 16-bit (BF16) training rather than FP16, and BF16 requires specialized,

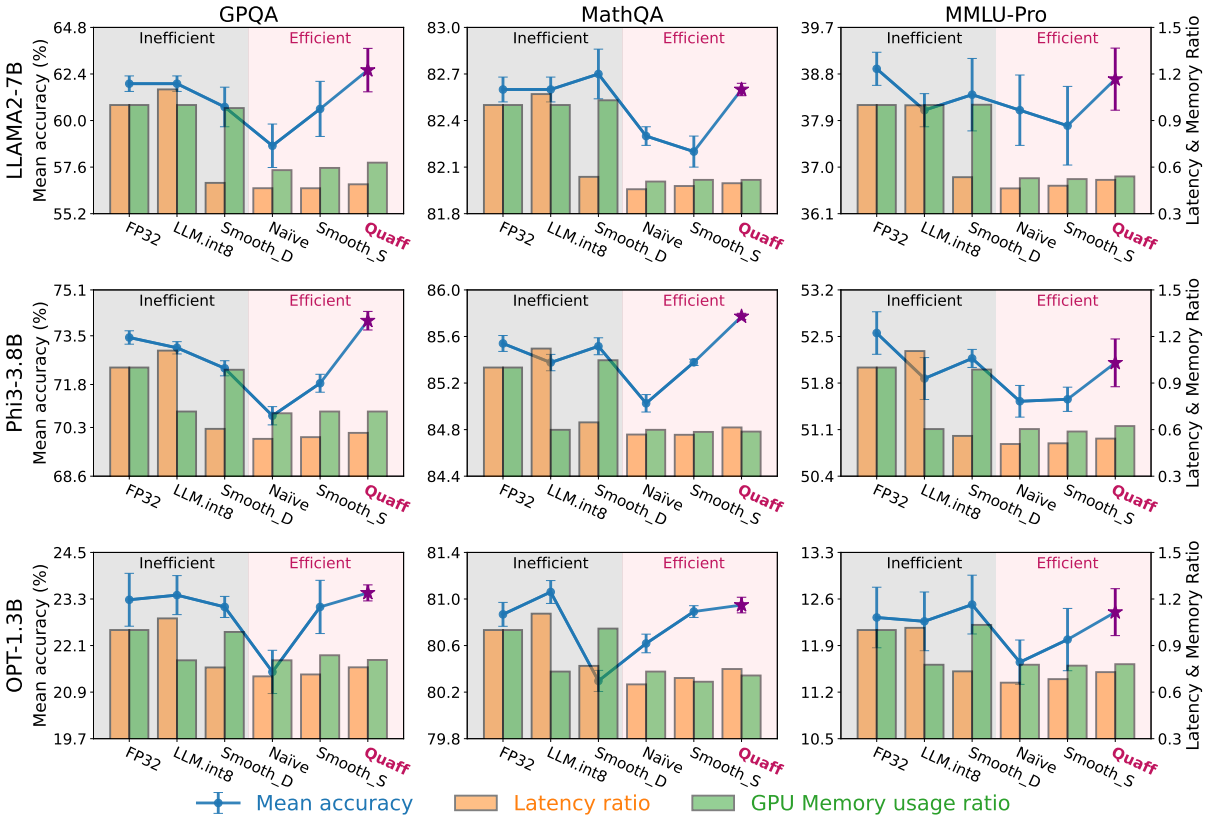


Figure 4: Comparison of accuracy, latency, and memory footprints between our proposed Quaff and various WAQ baselines on three reasoning datasets using LoRA fine-tuning. Latency and memory footprint values are reported as ratios relative to FP32 models.

			Oasst1			Self-instruct			Finance-Alpaca			HH-RLHF		
	Latency	Memory	ROUGE-L↑	PPL↓	Acc↑	ROUGE-L↑	PPL↓	Acc↑	ROUGE-L↑	PPL↓	Acc↑	ROUGE-L↑	PPL↓	Acc↑
FP32	7.86s	24.1GB	0.582	5.295	0.679	0.670	2.086	0.817	0.618	4.509	0.608	0.511	4.579	0.605
LLM.int8	8.92s	16.4GB	0.573	5.519	0.669	**0.670**	2.103	0.811	0.615	**4.599**	**0.607**	0.510	**4.565**	0.608
Smooth_D	4.48s	23.0GB	0.578	**5.323**	0.675	0.619	2.186	0.820	0.614	4.605	0.606	0.510	4.616	0.607
Naive	4.06s	14.6GB	0.578	5.382	**0.676**	0.650	2.123	0.820	0.615	4.633	0.605	0.510	4.614	0.608
Smooth_S	4.09s	14.7GB	**0.579**	5.342	0.676	0.636	**2.099**	**0.822**	**0.616**	4.645	0.605	**0.511**	4.595	**0.610**
Quaff	4.35s	14.9GB	**0.581**	**5.295**	**0.678**	**0.682**	**2.098**	**0.823**	**0.617**	**4.595**	**0.606**	**0.512**	**4.576**	**0.611**

Table 1: ROUGE-L, Perplexity (PPL) and accuracy (Acc) on four instruction-tuning datasets with Phi3-3.8B using LoRA fine-tuning. We report the average latency per step and the maximum GPU memory usage during fine-tuning.

advanced GPUs (at least with the Ampere architecture), limiting accessibility on many devices.

For outlier channel identification, we use 512 data samples from OIG/Chip2 (LAION, 2023) as the calibration dataset. For different layers, we set the size of O differently. Specifically, we allocate a maximum budget of $0.03\%c_{in}$ outlier channels for the q_proj, k_proj, v_proj, up_proj, $4\%c_{in}$ for o_proj , and $10\%c_{in}$ for $down\_proj$. The overall maximum overhead for outlier channels is maintained at less than 5%. Moreover, to simulate a typical user scenario, most experiments are conducted on a mid-range GPU, RTX 5880 Ada, which offers a similar computing speed to an RTX 4080 but with higher GPU memory. Additionally, some experiments are performed on a laptop with an RTX 2080 Super to demonstrate performance under custom-grade devices. Detailed hyperparameters for the experimental settings are provided in Sec. E.

Baseline settings. We compare our proposed Quaff framework with naive quantization (Naive), LLM.int8 (Dettmers et al., 2022), and SmoothQuant (Xiao et al., 2023), which has two versions: static (Smooth_S) and dynamic (Smooth_D). We also include the (FP32) in our experiments. We also include the FP32 baseline in our experiments. We followed the settings in the paper of baselines. Certain rotation-based WAQ methods, such as DuQuant (Lin et al., 2025) and RoLoRA (Huang et al., 2024b), are excluded

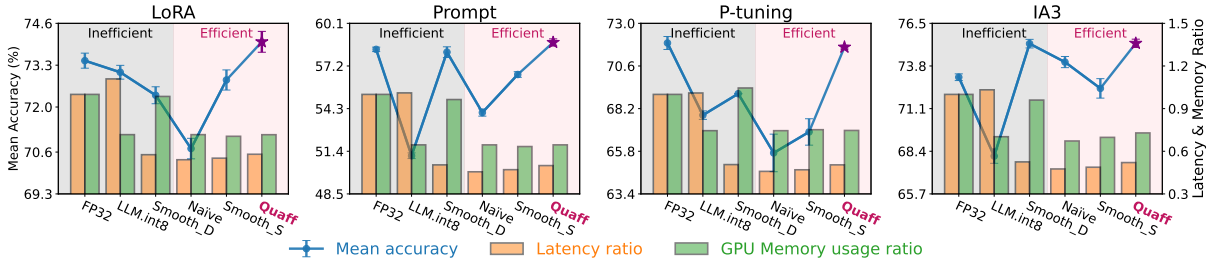


Figure 5: Accuracy and fine-tuning costs on the GPQA dataset using Phi3-3.8B with different fine-tuning strategies.

from our comparison due to their computational inefficiency and architectural rigidity in rotation during fine-tuning. Further discussion on the baseline methods is in Sec. A.

4.2 Performance Evaluation

In the result, methods are categorized into efficient and inefficient approaches based on latency and memory footprint levels, with pink and gray backgrounds, respectively.

Fine-tuning on Reasoning Tasks. To demonstrate the efficiency and effectiveness of Quaff, we compare it with other WAQ baselines on three reasoning benchmarks (GPQA, MMLU-Pro, and MathQA), using OPT-1.3B, Phi3-3.8B, and LLaMa2-7B models with LoRA fine-tuning. A comprehensive comparison in terms of performance, end-to-end latency, and maximum GPU memory usage during fine-tuning is presented in Fig. 4. Remarkably, Quaff achieves the best trade-off among accuracy, computational cost, and memory footprint across all cases. For instance, on the GPQA dataset using the LLaMA2-7B model, Quaff delivers a 2.0% accuracy gain over comparable baselines with similar efficiency. Additionally, Quaff reduces latency by 51.1% and GPU memory usage by 37.1% versus FP32, even slightly outperforming full-precision training. These results validate Quaff's ability to mitigate activation outlier impacts without sacrificing performance. Moreover, Quaff achieves better or at least similar performance compared to Smooth_D, which dynamically scales all channels, demonstrating outlier channel invariance as assumed in the OSSH.

Notably, the Phi3-3.8B model surpasses the LLaMA2-7B model in accuracy across most datasets despite having only half the parameters. Given this empirical trend, we select the Phi3-3.8B as our default model for subsequent experiments.

	Latency	Memory	ROUGE-L↑	PPL↓	Acc↑
FP32	115.76s	8+7.1G	0.598	4.042	0.665
Smooth_D	131.67s	8+7.1G	0.589	4.116	0.663
LLM.int8	20.43s	6.6G	0.633	3.185	0.697
Naive	10.90s	6.3G	**0.639**	2.995	0.705
Smooth_S	11.90s	6.3G	0.638	**2.970**	**0.706**
Quaff	12.46s	6.3G	**0.643**	**2.962**	**0.707**

Table 2: Results after 24 hours of LoRA fine-tuning on the OIG/CHIP2 dataset using Phi3-3.8B on a laptop (RTX 2080 Super 8GB) with 16GB shared memory.

Fine-Tuning on Instruction-tuning Tasks. To assess personalized chatbot adaptation, we evaluate Quaff on four instruction-tuning datasets (Oasst1, Self-Instruct, Finance-Alpaca, and HH-RLHF) using the Phi3-3.8B model. We report metrics including ROUGE-L, perplexity, average accuracy, average latency per fine-tuning step, and maximum GPU memory usage. As shown in Table 1, Quaff achieves the best/second-best perplexity, accuracy, and ROUGE-L scores across all tasks while maintaining low latency and memory usage. This demonstrates its viability for real-world conversational LLM deployment on local devices.

Consumer Hardware Compatibility. To emphasize accessibility for ordinary users, we evaluated the efficiency of Quaff and other baselines on an MSI laptop equipped with an NVIDIA RTX 2080 Super (8 GB) and an Intel Core i7 processor. We conducted experiments on the OIG/CHIP2 dataset using the Phi3-3.8B model for 24 hours of training, with a batch size of 1 and a gradient accumulation of 16. The results, illustrated in Tab. 2, show that Quaff achieves the best performance among all metrics. Due to out of CUDA memory, the FP32 and Smooth_D suffer from high latency, so Quaff achieves 8.29× speedup versus FP32.

Fine-Tuning with Different Strategies. Fine-tuning strategies may vary depending on the downstream task. To demonstrate the versatility of our proposed Quaff, we compare it with other WQA

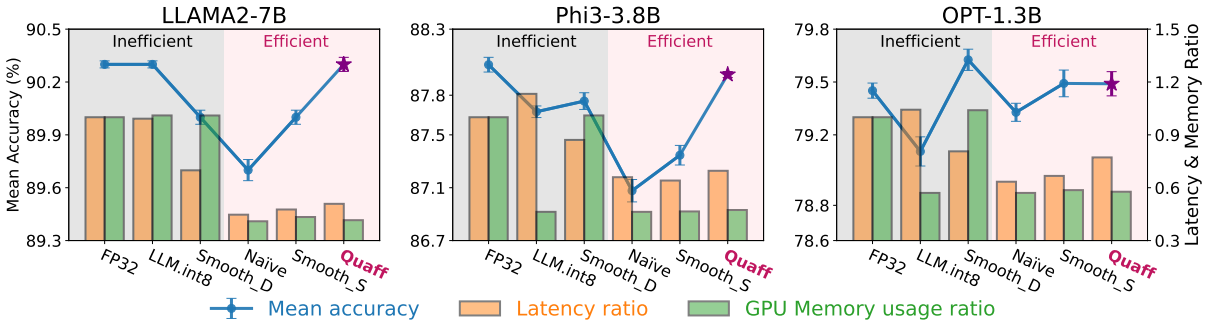


Figure 6: Results of LoRA fine-tuning on LAMBADA dataset with input/output size of 4K on different models.

	OIG/CHIP2	LAMBADA	GPQA
OIG/CHIP2	0.665	0.880	0.740
LAMBADA	0.659	0.880	0.733
GPQA	0.660	0.873	0.742

Table 3: The Impact of cross-dataset calibration on Rouge-L (OIG/CHIP2) and accuracy (others). Columns: fine-tuning datasets; rows: calibration datasets.

	LoRA	Prompt	P-Tuning	IA3
Best baseline	0.731	0.581	0.690	**0.751**
Quaff w/o Mo	**0.732**	**0.583**	**0.710**	0.745
Quaff	**0.740**	**0.588**	**0.717**	**0.752**

Table 4: Mean accuracy on the GPQA dataset using the Phi3-3.8B model for Quaff, Quaff without Momentum, and the best baseline across different fine-tuning strategies, where the best baseline refers to the highest results achieved among prior WAQ methods.

	Latency	Memory	ROUGE-L↑	PPL↓	Acc↑
FP32	14.14s	27.0GB	0.516	7.865	0.594
LLM.int8	16.36s	18.0GB	0.511	**8.068**	0.589
Smooth_D	15.79s	27.6GB	**0.512**	8.199	**0.590**
Naive	10.28s	17.6GB	0.510	8.262	0.588
Smooth_S	10.53s	17.7GB	0.510	8.262	0.589
Quaff	11.60s	17.7GB	**0.513**	**8.189**	**0.590**

Table 5: Results of LoRA fine-tuning on the LongForm dataset with output size of 4K on Phi3-3.8B model.

baselines on the GPQA dataset using four popular PEFT strategies (LoRA, Prompt, P-tuning, and IA3) with Phi3-3.8B model. Figure 5 shows Quaff outperforms all baselines, including FP32, across all strategies. This highlights the robustness of Quaff in diverse fine-tuning paradigms.

The Impact of Cross-Datasets Calibration. Different calibration datasets may produce distinct outlier channels, which may not be suitable for downstream tasks if the calibration dataset is mismatched. To investigate the selection of the calibration dataset, we evaluate Quaff on three datasets: OIG/CHIP2 (instruction-tuning), LAMBADA (long-context task), and GPQA (reasoning), using each as the calibration dataset for the others. We track ROUGE-L for OIG/CHIP2, accuracy for LAMBADA and GPQA. The results shown in Tab. 3 indicate that OIG/CHIP2 performs best as the calibration dataset, highlighting the advantage of instruction-tuning datasets for calibration.

Fine-Tuning on Long Text Tasks. In long-text tasks, activations in large language models (LLMs) become increasingly complex and unpredictable. To demonstrate that our Quaff approach maintains both effectiveness and efficiency in such scenarios, we conducted experiments on the LAMBADA (Paperno et al., 2016) (last word prediction) and LongForm (Köksal et al., 2023) datasets (instruction following), designed for long-context understanding and long-text generation, respectively. Both input and output sequences were set to a maximum length of 4K tokens, with a batch size of 1 and gradient accumulation of 16. Quaff achieved performance comparable to FP32 models in long-context language understanding (Fig. 6) and the best/second-best performance among WAQ methods in the long-text generation task(in Tab. 5).

The Impact of Momentum Mechanism. To evaluate the effectiveness of the momentum mechanism in Quaff, we conducted an ablation experiment isolating the impact of Quaff's momentum mechanism on the GPQA dataset using the Phi3-3.8B model. The results, presented in Table 4, demonstrate that momentum-based scaling factors enhance Quaff's performance, and improve accuracy by $0.5\% - 0.8\%$ by prioritizing persistent Notably, even without the momentum mechanism, Quaff still outperforms the best baseline, highlighting the effectiveness of reduced weight sensitivity by $(s - 1)$ scaling.

5 Related Work

Parameter-Efficient Fine-Tuning. Parameter-Efficient Fine-Tuning (PEFT) adapts tasks by training only a small subset of a pretrained model's parameters. Techniques such as adapter tuning (Houlsby et al., 2019) (inserting lightweight modules), prefix/prompt tuning (Li and Liang, 2021; Lester et al., 2021) (prepending learnable tokens), LoRA (Hu et al., 2021) (low-rank weight updates), and IA3 (Liu et al., 2022) (scaling activations) reduce computational and memory costs versus full fine-tuning. Yet, for billion-parameter LLMs, these methods can still be too resource-intensive for edge-device deployment.

Quantization Fine-Tuning for LLMs. Quantization (Jacob et al., 2018) addresses the limitations of PEFT by compressing model weights and activations. Weight-Only Quantization (WOQ) (Kwon et al., 2022; Dettmers et al., 2024; Xu et al., 2023; Li et al., 2023; Liu et al., 2023; Guo et al., 2023; Kim et al., 2024; He et al., 2023; Lee et al., 2024a) compresses pretrained weights to low precision (Frantar et al., 2022; Lin et al., 2023) while retaining a small set of trainable parameters in full precision for updating. Although WOQ reduces memory usage, it introduces mixed-precision computational overhead, often resulting in slower training compared to full-precision baselines.

Weight-Activation Quantization (WAQ) quantizes both weights and activations for hardware-friendly computation (Zhou et al., 2016). However, LLMs exhibit emergent channel-wise outliers inflating quantization errors (Wu et al., 2023b). To mitigate this, previous works (Dettmers et al., 2022; Wei et al., 2022; Xiao et al., 2023; Wei et al., 2023; Wang et al., 2024a) redistribute outliers to weights via channel-wise scaling. Some variants (Ashkboos et al., 2024; Lin et al., 2025; Huang et al., 2024b; Liu et al., 2024; Kampeas et al., 2023) refine this by replacing scaling with rotation, but they suffer from computational inefficiency and architectural rigidity for activation rotation during fine-tuning, as detailed in Sec. A. Therefore, existing WAQ methods suffer from **coupling** between weight and activation quantization, and suffer from either high memory/compute overhead for handling full-precision weights (dynamic scaling) or accuracy loss from distribution shifts (static scaling). Quaff solves this by targeted scaling based on OSSH to decouple quantization, enabling efficient fine-tuning with near-FP32 accuracy and minimal overhead.

6 Conclusion

This paper proposes an Outlier Spatial Stability Hypothesis (OSSH): During fine-tuning, certain activation outlier channels retain stable spatial positions across training iterations. Based on OSSH, we propose Quaff, a quantized parameter-efficient fine-tuning framework for LLMs. Quaff decouples the quantization between weights and activations by targeted momentum scaling on stable outlier channels, achieving lower quantization error with little overhead. We conduct extensive experiments on ten benchmarks and show that Quaff outperforms existing approaches in terms of performance, computational cost, and memory footprints.

Acknowledgment

This paper is partially supported by Hong Kong Research Grants Council (RGC) grant #11203523.

Limitations

Our work prioritizes democratizing quantized fine-tuning for non-expert users. Therefore, our work lacks in-depth exploration and design in the following areas: 1. Larger Model. We focus exclusively on models up to 7B parameters (e.g., LLaMA-2-7B, Phi-3-3.8B), and do not consider larger, state-of-the-art models. 2. Hardware-Specific Optimizations. We did not use hardware-specific optimizations (such as Ampere Tensor Cores or H100 FP8 acceleration (Kim et al., 2025)) and top-tier GPUs (e.g., A100) in our experiments, limiting exploration of high efficiency on advanced hardware for enterprise users. 3. Precision Constraints. We adopt only INT8 quantization and do not explore INT4/INT2 precision, resulting in a lower compression rate. 4. Layer-Agnostic Implementation. Quaff applies uniform quantization to linear layers without custom fusion or exploiting sparsity, sacrificing potential latency gains from architecture-specific tuning. 5. Single-GPU Focus. We only consider single-GPU scenarios in our experiments and do not explore multi-GPU configurations.

These limitations reflect our commitment to accessibility and compatibility for non-expert users while acknowledging areas for future enhancement.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Gaurang Bharti. 2023. gbharti/finance-alpaca. *https://huggingface.co/datasets/gbharti/finance-alpaca*.

Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. 2020. An overview on edge computing research. *IEEE access*, 8:85714–85728.

Ning Chen, Tie Qiu, Xiaobo Zhou, Songwei Zhang, Weisheng Si, and Dapeng Oliver Wu. 2024. A distributed co-evolutionary optimization method with motif for large-scale iot robustness. *IEEE/ACM Transactions on Networking*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 2021. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.

George H. Forman and John Zahorjan. 1994. The challenges of mobile computing. *Computer*, 27(4):38–47.

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

Zihao Fu, Anthony Man-Cho So, and Nigel Collier. 2023. A stability analysis of fine-tuning a pre-trained model. *arXiv preprint arXiv:2301.09820*.

Han Guo, Philip Greengard, Eric P Xing, and Yoon Kim. 2023. Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning. *arXiv preprint arXiv:2311.12023*.

Yefei He, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. 2023. Efficientdm: Efficient quantization-aware fine-tuning of low-bit diffusion models. *arXiv preprint arXiv:2310.03270*.

Jung Hwan Heo, Jeonghoon Kim, Beomseok Kwon, Byeongwook Kim, Se Jung Kwon, and Dongsoo Lee. 2023. Rethinking channel dimensions to isolate outliers for low-bit weight quantization of large language models. *arXiv preprint arXiv:2309.15531*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hong Huang, Hai Yang, Yuan Chen, Jiaxun Ye, and Dapeng Wu. 2025. Fedrts: Federated robust pruning via combinatorial thompson sampling. *arXiv preprint arXiv:2501.19122*.

Hong Huang, Lan Zhang, Chaoyue Sun, Ruogu Fang, Xiaoyong Yuan, and Dapeng Wu. 2023. Distributed pruning towards tiny neural networks in federated learning. In *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, pages 190–201. IEEE.

Hong Huang, Weiming Zhuang, Chen Chen, and Lingjuan Lyu. 2024a. Fedmef: Towards memory-efficient federated dynamic pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27548–27557.

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

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2018. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30.

Tomasz Imielinski and Henry F Korth. 1996. *Mobile computing*, volume 353. Springer Science & Business Media.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

Aojie Jiang, Li Du, and Yuan Du. 2024. Groupq: Group-wise quantization with multi-objective optimization for cnn accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Joseph Kampeas, Yury Nahshan, Hanoch Kremer, Gil Lederman, Shira Zaloshinski, Zheng Li, and Emir Haleva. 2023. Rotation invariant quantization for model compression. *arXiv preprint arXiv:2303.03106*.

Jeonghoon Kim, Jung Hyun Lee, Sungdong Kim, Joonsuk Park, Kang Min Yoo, Se Jung Kwon, and Dongsoo Lee. 2024. Memory-efficient fine-tuning of compressed large language models via sub-4-bit integer quantization. *Advances in Neural Information Processing Systems*, 36.

Jiwoo Kim, Joonhyung Lee, Gunho Park, Byeongwook Kim, Se Jung Kwon, Dongsoo Lee, and Youngjoo Lee. 2025. An investigation of fp8 across accelerators for llm inference. *arXiv preprint arXiv:2502.01070*.

Abdullatif Köksal, Timo Schick, Anna Korhonen, and Hinrich Schütze. 2023. Longform: Optimizing instruction tuning for long text generation with corpus extraction. *arXiv preprint arXiv:2304.08460*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Se Jung Kwon, Jeonghoon Kim, Jeongin Bae, Kang Min Yoo, Jin-Hwa Kim, Baeseong Park, Byeongwook Kim, Jung-Woo Ha, Nako Sung, and Dongsoo Lee. 2022. Alphatuning: Quantization-aware parameter-efficient adaptation of large-scale pre-trained language models. *arXiv preprint arXiv:2210.03858*.

LAION. 2023. Open-instruction-generalist dataset. *https://github.com/LAION-AI/Open-Instruction-Generalist*.

Changhun Lee, Jun-gyu Jin, Younghyun Cho, and Eunhyeok Park. 2024a. Qeft: Quantization for efficient fine-tuning of llms. *arXiv preprint arXiv:2410.08661*.

Changhun Lee, Jungyu Jin, Taesu Kim, Hyungjun Kim, and Eunhyeok Park. 2024b. Owq: Outlier-aware weight quantization for efficient fine-tuning and inference of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 13355–13364.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. 2023. Loftq: Lora-fine-tuning-aware quantization for large language models. *arXiv preprint arXiv:2310.08659*.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. 2025. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems*, 37:87766–87800.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. 2023. Qllm: Accurate and efficient low-bitwidth quantization for large language models. *arXiv preprint arXiv:2310.08041*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024. Spinquant–llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. 2023. Omniquant: Omnidirectionally calibrated quantization for large language models. *arXiv preprint arXiv:2308.13137*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jinguang Wang, Yuexi Yin, Haifeng Sun, Qi Qi, Jingyu Wang, Zirui Zhuang, Tingting Yang, and Jianxin Liao. 2024a. Outliertune: Efficient channel-wise quantization for large language models. *arXiv preprint arXiv:2406.18832*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

A Waswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A Gomez, L Kaiser, and I Polosukhin. 2017. Attention is all you need. In *NIPS*.

Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. 2023. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*.

Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. 2022. Outlier suppression: Pushing the limit of low-bit transformer language models. *Advances in Neural Information Processing Systems*, 35:17402–17414.

Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. 2023a. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136.

Xiaoxia Wu, Cheng Li, Reza Yazdani Aminabadi, Zhewei Yao, and Yuxiong He. 2023b. Understanding int4 quantization for transformer models: Latency speedup, composability, and failure cases. *arXiv preprint arXiv:2301.12017*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

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

Jiaming Yang, Chenwei Tang, Caiyang Yu, and Jiancheng Lv. 2024. Gwq: Group-wise quantization framework for neural networks. In *Asian Conference on Machine Learning*, pages 1526–1541. PMLR.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*.

A More WAQ Method Analysis

We analyze key weight-activation quantization (WAQ) baselines beyond the classical scaling method (Xiao et al., 2023), highlighting their computational and practical limitations compared to Quaff.

LLM.int8. LLM.int8 (Dettmers et al., 2022) employs mixed-precision outlier handling by dynamically detecting high-magnitude activation channels via a fixed threshold σ. During fine-tuning, it splits computations into:

$$\mathbf{Y} = \mathbf{X}\mathbf{W} = \underbrace{\mathbf{X}_{:,\bar{O}}\mathbf{W}_{\bar{O}}}_{\text{Quantized}} + \underbrace{\mathbf{X}_{:,O}\mathbf{W}_{O}}_{\text{Full-Precision}}, \tag{10}$$

where O denotes outlier channels and $\bar{O}$ denotes normal channels. Though structurally similar to Quaff, LLM.int8's reliance on dynamic detection forces full-weight dequantization to retrieve W_O, incurring prohibitive latency. As activation distributions shift during fine-tuning, $\text{card}(O)$ often grows to match c_{in} (Fig. 4), rendering memory savings negligible versus FP32.

Conceptually, LLM.int8 can be reframed as a dynamic scaling variant:

$$\begin{aligned}\mathbf{Y} = \mathbf{X}\mathbf{W} &= \mathbf{X}\mathbf{s}^{-1}\mathbf{s}\mathbf{W} + \mathbf{X}\bar{\mathbf{s}}^{-1}\bar{\mathbf{s}}\mathbf{W} \\ &= \hat{\mathbf{X}}\hat{\mathbf{W}} + \mathbf{X}\bar{\mathbf{s}}^{-1}\bar{\mathbf{s}}\mathbf{W} \\ &\approx \Delta_{\hat{\mathbf{X}}} \cdot (\hat{\mathbf{X}}_{int}\hat{\mathbf{W}}_{int}) \cdot \Delta_{\hat{\mathbf{W}}} + \mathbf{X}\bar{\mathbf{s}}^{-1}\bar{\mathbf{s}}\mathbf{W},\end{aligned} \tag{11}$$

where $\mathbf{s} + \bar{\mathbf{s}} = \mathbf{1}$, and $\mathbf{s}_i = \mathbf{1}_{\max(|\mathbf{X}_{:,\mathbf{i}}| > \sigma)}$, where σ is a predefined threshold. This exposes its core inefficiency: real-time scaling factor computation and global requantization.

Rotation-Based Methods. Several rotation-based approaches (Lin et al., 2025; Huang et al., 2024b; Ashkboos et al., 2024) also recognize that static scaling cannot effectively suppress outliers due to fluctuation. To address this, these methods replace scaling with orthogonal transformations:

$$\begin{aligned}\mathbf{Y} = (\mathbf{X}\mathbf{R})(\mathbf{R}^{\mathbf{T}}\mathbf{W}) &= \hat{\mathbf{X}}\hat{\mathbf{W}} \\ &\approx \Delta_{\hat{\mathbf{X}}} \cdot (\hat{\mathbf{X}}_{int}\hat{\mathbf{W}}_{int})\Delta_{\hat{\mathbf{W}}},\end{aligned} \tag{12}$$

where $\mathbf{R}$ is an orthogonal matrix satisfying $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $|\mathbf{R}| = 1$. While effective for reducing post-training quantization error, rotation-based methods incur significant computational overhead. Although the Fast Walsh-Hadamard Transform (FWHT) reduces rotation complexity to $O(nlogn)$ complexity, its recursive computation pattern limits vectorization and parallelism on general hardware and increases memory access fragmentation, making it much less efficient than hardware-friendly scaling methods. To mitigate these inefficiencies, recent approaches (Huang et al., 2024b; Lin et al., 2025) avoid online rotation by merging rotational transformations (R_1) into consecutive linear layers. However, this introduces **architectural rigidity**, necessitating careful architectural adjustments (e.g., position encodings, residual, multi-head concat, layer norm). This design constraint hinders the generalization adaptability to emerging variants like multi-head latent attention blocks.

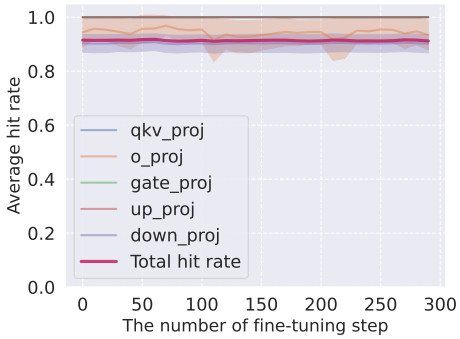


Figure 7: Average hit rate of real-time vs. predefined outlier channel indices across layers in **LLaMA2-7B** during fine-tuning on OIG/Chip2.

Bias-Centric Variants. Methods like Omniquant (Shao et al., 2023) augment scaling with learnable bias terms:

$$\begin{aligned}\mathbf{Y} = \mathbf{X}\mathbf{W} + \mathbf{B} &= [(\mathbf{X} - \delta)\mathbf{s}^{-1}][\mathbf{s}\mathbf{W}] + (\mathbf{B} + \delta\mathbf{W}) \\ &= \hat{\mathbf{X}}\hat{\mathbf{W}} + \hat{\mathbf{B}} \approx \Delta_{\hat{\mathbf{X}}} \cdot (\hat{\mathbf{X}}_{int}\hat{\mathbf{W}}_{int})\Delta_{\hat{\mathbf{W}}} + \hat{\mathbf{B}},\end{aligned} \tag{13}$$

Where δ is a shift factor. However, bias terms do not resolve the fundamental coupling between weight and activation quantization, $\hat{\mathbf{W}}$ still depend on real-time $\hat{\mathbf{X}}$. Moreover, many LLMs (e.g., LLaMA, Phi-3) omit bias terms entirely, limiting generality.

B Analysis of Outlier Spatial Stability Hypothesis

Empirical Validation. We validate the Outlier Spatial Stability Hypothesis (OSSH) by measuring the overlap between predefined outlier channels O in fine-tuning iterations. And dynamically detected channels during fine-tuning. We analyze six

Figure 8: Average hit rate of real-time vs. **uniformly** distributed predefined outlier channel indices across layers in Phi3-3.8B during fine-tuning on OIG/Chip2.

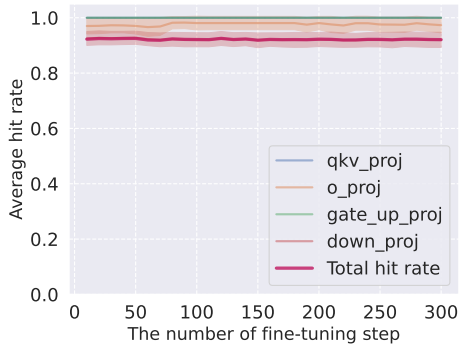


Figure 9: Average hit rate of real-time vs. predefined outlier channel indices across layers in Phi-3-3.8B **during fine-tuning on GPQA.**

core linear layers in transformer (Waswani et al., 2017; Devlin et al., 2018) of LLMs: 1. Attention projections: q_proj, k_proj, v_proj. 2. Output projection: o_proj. 3. FFN layers up_proj and $down\_proj$. Notably, as previous works (Lin et al., 2025) indicate, o_proj and $down\_proj$ exhibit higher outlier channel volatility due to input-sensitive activation patterns. To address this, we implement a non-uniform budget allocation: Stable layers q_proj, k_proj, v_proj, up_proj have $0.03\%c_{in}$. Volatile layers o_proj have $4\%c_{in}$ and highly dynamic layers $down\_proj$ have $10\%c_{in}$. It should be noted that while the activations in $down\_proj$ and o_proj exhibit volatility, they still align with our OSSH, as the overall outlier channels set remain stable. As shown in Figs. 7 and 8, this strategy achieves $> 90\%$ hit rates for LLaMA2-7B and Phi-3-3.8B on OIG/Chip2. In contrast, uniform budget allocation (Fig. 8) reduces hit rates to $< 50\%$ for volatile layers, confirming the necessity of layer-specific budget distribution.

Cross-Dataset Generalization. To test OSSH's robustness, we evaluate Phi-3-3.8B on reasoning dataset GPQA using outlier channels calibrated on instruction-tuning OIG/Chip2. As shown in Fig. 9, hit rates remain >90%, demonstrating hypothesis invariance across task domains.

C Outlier Distribution Shift Analysis

Static scaling methods fail to adapt to activation distribution shifts, as evidenced by the declining similarity between predefined and real-time scaling factors (Fig. 10). In layers $down\_proj$, similarity drops to -35% after 1,000 iterations, explaining the accuracy degradation observed in prior work.

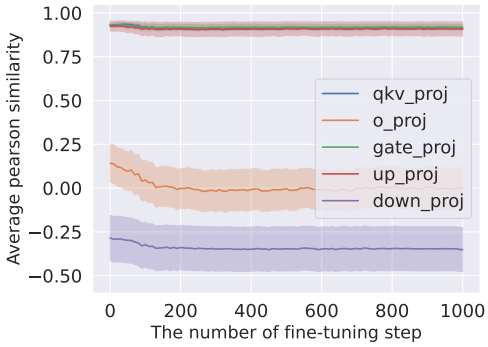


Figure 10: Pearson similarity between static and dynamic scaling factors (top 1%) across layers in LLaMA2-7B during fine-tuning on OIG/Chip2.

D More Experiment Analysis

D.1 The Impact of Long Context

To explore The impact of Long Context, we investigated outlier stability on Phi-3 with a 32K input/output context. Using a 5% budget, we pre-identified outlier channels via OIG/CHIP-2 and measured the average hit rate (*i.e.,* whether an outlier appears in pre-identified outlier channels) during fine-tuning on the LongForm dataset. The results in Table 6 demonstrates the effectiveness of OSSH in by 90% average hit rate in 32K context task.

Layer	Average Hit rate
QKV_proj	100%
gate_up_proj	100%
o_proj	98.3%
down_proj	91.1%

Table 6: Average Hit rate for each type of layer in 32K context task.

	GPQA		LAMBADA	
	llama2 7B	Phi 3 3.8B	llama2 7B	Phi 3 3.8B
5%	62.6	74.0	90.3	87.9
3%	62.4	74.0	90.4	87.9
1%	61.2	73.6	90.2	87.9
0.1%	59.0	72.4	89.6	87.2
0%	58.7	70.7	89.7	87.0

Table 7: Performance with different overall budgets.

D.2 The Impact of budget for outlier channels

To show the impact of different budgets, we conduct experiments with different budget ratios. Since the current budget is sufficient, we focused on experiments exploring the impact of a lower budget. We evaluate budgets of 5%, 3%, 1%, 0.1%, and 0% across different models and tasks. Specifically, we reduced the budget for $down\_proj$ and o_proj to obtain the overall budget of 3% and 1%, and we set a uniform layer-wise budget of 0.1% to achieve an overall budget of 0.1%. The results in the Table 7 show that sensitivity decreases for long-text tasks (*e.g.,* LAMBADA), smaller models (*e.g.,* Phi-3).

E More Experimental Details

Dataset settings. We set the number of fine-tuning epochs to 1 for instruction-tuning datasets (Alpaca-Finance, HH-RLHF, Self-Instruct, OIG/Chip2, and Oasst1) as well as long-text datasets (Longform and LAMBADA). For reasoning datasets (GPQA, MathQA, and MMLU-Pro), we set the number of fine-tuning epochs to 5. The prompts for instruction-tuning and long-text datasets follow their respective dataset settings, while for GPQA, MathQA, and MMLU-Pro, the prompt is:

"#Input Please select one of the following options: (A) #Option1. (B) #Option2. (C) #Option3. (D) #Option4."

and the reference text is formatted as :

#Explanation. The answer is #Correct.

And for MMLU-pro which does not provide sufficient explanation for training data, therefore, it left "#Explanation" as blank.

Model and experimental settings. We use Adam optimizer with learning rate as 2e-4 following previous work (Dettmers et al., 2024). The rank of LoRA is 16, the alpha of LoRA is 16, and the LoRA dropout is 0.1. The number of virtual tokens in Prompt and P-tuning is set as 20. We leverage bitsandbytes (Dettmers et al., 2024) for INT8 acceleration. We set $\gamma = 0.2$ in the Equation 7.

F Quantization Granularity

Quantization has different levels of granularity related to different sizes of the quantization step size $\Delta_{\mathbf{X}}$ and $\Delta_{\mathbf{W}}$. The ***per-tensor*** quantization uses one single quantization step for the entire matrix, i.e., $\Delta_{\mathbf{X}} \in \mathbb{R}$ and $\Delta_{\mathbf{W}} \in \mathbb{R}$, which can achieve fast quantization speed but high quantization loss. The ***per-token*** and per-output-channel (***per-OC***) quantization use different quantization step sizes for each token of activations and each output channel of weights, *i.e.,* $\Delta_{\mathbf{X}} \in \mathbb{R}^t$ and $\Delta_{\mathbf{W}} \in \mathbb{R}^{c_{out}}$, which introduce lower quantization loss and higher computational overhead compared to per-tensor quantization. There is also a coarse-grained version of per-channel quantization called group-wise quantization (*per-group*) (Yang et al., 2024; Jiang et al., 2024), which divides weight into different groups by classification and uses different quantization steps for different groups. However, per-group quantization requires specific hardware support for grouping operations. The per-input-channel (***per-IC***) quantization (Heo et al., 2023) using different quantization step sizes for each input channel, i.e., $\Delta_{\mathbf{X}} \in \mathbb{R}^{c_{in}}$ and $\Delta_{\mathbf{W}} \in \mathbb{R}^{c_{in}}$. However, per-IC quantization converts the MatMul into $\mathbf{Y} \approx \mathbf{X}_{int}\Delta_{\mathbf{X}}\Delta_{\mathbf{W}}\mathbf{W}_{int}$, which fails to achieve the integer MatMul for acceleration. *Therefore, only per-tensor, per-token, and per-OC quantization methods can achieve both memory and computational efficiency on general hardware.*

G Broader Impact

The deployment of large language models (LLMs) on personal and resource-constrained devices remains a key challenge due to the computational and memory demands of fine-tuning. This work introduces Quaff, a quantized parameter-efficient fine-tuning framework that enables efficient LLMs fine-tuning on consumer-grade hardware without full-precision weight storage. By leveraging the Outlier Spatial Stability Hypothesis (OSSH), Quaff facilitates hardware-friendly quantization, thereby bridging the gap between state-of-the-art language model capabilities and real-world accessibility.

This democratization of Quaff opens up significant societal and technological benefits in resource-constrained scenarios (Huang et al., 2023, 2024a), such as mobile computing (Forman and Zahorjan,

1994; Imielinski and Korth, 1996), edge computing (Cao et al., 2020; Chen et al., 2024) and cross-device federated learning (McMahan et al., 2017; Huang et al., 2025). It empowers individuals, educators, small businesses, and developers in regions with limited computational resources to personalize and adapt LLMs for their specific needs, ranging from localized chatbots to domain-specific assistants without relying on cloud infrastructure. Furthermore, enabling on-device fine-tuning supports data privacy, as sensitive user data can remain local, reducing the risk of data leakage through centralized training.

However, with broader accessibility comes the potential for misuse. Easy personalization of LLMs could be exploited to fine-tune harmful behaviors or generate misinformation. We encourage the development of safeguards, such as differential privacy and fine-tuning auditing tools, to mitigate these risks.

Overall, Quaff advances the vision of inclusive and privacy-conscious AI by making powerful language technologies more accessible, efficient, and sustainable across a wide range of real-world environments.