GSQ-Tuning: Group-Shared Exponents Integer in Fully Quantized Training for LLMs On-Device Fine-tuning

Sifan Zhou[1,2†‡], Shuo Wang[1†], Zhihang Yuan[1†], Mingjia Shi[1], Yuzhang Shang[3✉], Dawei Yang[1✉]

[1]Houmo AI [2]Southeast University [3]Illinois Institute of Technology

Abstract

Large Language Models (LLMs) fine-tuning technologies have achieved remarkable results. However, traditional LLM fine-tuning approaches face significant challenges: they require large Floating Point (FP) computation, raising privacy concerns when handling sensitive data, and are impractical for resource-constrained edge devices. While Parameter-Efficient Fine-Tuning (PEFT) techniques reduce trainable parameters, their reliance on floating-point arithmetic creates fundamental incompatibilities with edge hardware. In this work, we introduce a novel framework for on-device LLM fine-tuning that eliminates the need for floating-point operations in both inference and training, named **GSQ-Tuning**. At its core is the Group-Shared Exponents Integer format, which efficiently represents model parameters in integer format using shared exponents among parameter groups. When combined with LoRA-like adapters, this enables fully integer-based fine-tuning that is both memory and compute efficient. We demonstrate that our approach achieves accuracy comparable to FP16-based fine-tuning while significantly reducing memory usage ($\sim$ 50%). Moreover, compared to FP8, our method can reduce $\sim 5\times$ power consumption and $\sim 11\times$ chip area with same performance, making large-scale model adaptation feasible on edge devices.

1 Introduction

Recent advances in Large Language Models (LLMs) have delivered impressive results in a variety of natural language tasks (Touvron et al., 2023a,b; Liu et al., 2023). LLMs are typically trained in several stages, including large-scale pretraining followed by one or more fine-tuning phases (Dubey et al., 2024; Liu et al., 2024a). LLM fine-tuning approaches like supervised fine-tuning (SFT) (Zhang et al., 2023), usually employ curated, high-quality corpora for refining the model with a standard language modeling (Chiang et al., 2023).

Despite their effectiveness, most LLM fine-tuning approaches require powerful cloud servers or GPUs equipped with large memory capacities. This poses two significant challenges in real-world settings: (1) uploading sensitive data to remote servers poses a fundamental privacy risk, and (2) in many practical scenarios, models must be deployed on resource-constrained edge devices—such as mobile processors or embedded AI accelerators—where memory and power budgets are tightly limited. *Such constraints become critical in LLM's personalized applications, where data cannot be shared with the cloud and model updates must remain local to ensure privacy. Meeting these challenges thus necessitates on-device adaptation methods capable of preserving data privacy and functioning within the limited memory and compute budgets of edge hardware.*

Parameter-Efficient Fine-Tuning (PEFT) (Han et al., 2024) techniques such as LoRA (Hu et al., 2021) and QLoRA (Dettmers et al., 2023) alleviate part of this burden by reducing trainable parameters to around 1% of the original model. Unfortunately, they remain reliant on floating-point operations for both forward and backward passes, which clashes with edge-device constraints in three ways. First, during fine-tuning, weights, activation and gradients must be stored and updated in high-precision floating-point. It introduces additional overhead or even makes the LLM fine-tuning impractical on edge devices. Second, floating-point representations incur high memory overhead (e.g., FP16 doubles the memory cost compared to INT8; for a 7B-parameter model, this can surpass 20GB memory during fine-tuning process, presenting substantial challenges for mobile processors). Last but not least, commercial edge AI accelerators (e.g., Qualcomm Hexagon (QUALCOMM, 2024)) typically

[†]Equal contribution

[‡]Work done as an intern at Houmo AI

Findings of the Association for Computational Linguistics: ACL 2025, pages 22971–22988

get peak throughput only on integers, leaving up to 84% of compute units idle under FP16 training.

Therefore, eliminating floating-point arithmetic for fine-tuning would have a substantial impact on software, hardware, and application design for efficient on-device LLM adaptation (ARM, 2020; Kim et al., 2021). While previous studies on integer quantization (Jacob et al., 2018a; Kim et al., 2021; Xiao et al., 2022; Yuan et al., 2023) verify the feasibility of inference, they do not extend to gradient quantization, which is required for effective fine-tuning of LLMs at the edge.

In this paper, we propose a new framework for resource-efficient on-device LLM fine-tuning, termed **GSQ-Tuning**. Central to our method is the *Group-Shared Exponents Integer* format, a novel quantization strategy that replaces floating-point with a specialized integer-based representation. We integrate this with parameter-efficient LoRA-like modules to enable fully on-device fine-tuning without incurring large memory and computation costs. We further examine this design through a Pareto frontier analysis, which demonstrates how various bits-rank settings impact the trade-off between fine-tuning memory costs and accuracy. Extensive experiments across models of varying scales, different fine-tuning datasets, and diverse tasks have demonstrated the effectiveness and generalizability. We highlight our main contributions as follows:

- **Group-Shared Exponents Integer Quantization:** We introduce a quantization strategy that shares exponents among groups, thereby reducing the storage and computation overhead while still representing model parameters in integer format. Combined with LoRA-like adapters, our method supports fine-tuning under tight memory constraints.
- **Integer Forward and Backward Computations:** By extending integer quantization pipelines beyond inference to include gradients, both forward and backward passes remain hardware-friendly and efficiently utilize integer-focused edge accelerators.
- **Pareto Frontier for Quantization Bits and Low-rank:** We demonstrates how various bits-rank settings impact the trade-off between fine-tuning memory costs and accuracy through a Pareto frontier analysis. We empirically show that our approach achieves accuracy on par with FP16-based fine-tuning while dramatically lowering both $\sim 50\%$ memory usage. Furthermore, compared with FP8, at comparable performance levels, our method (GSE-INT5) reduces the power consumption of MAC unit by $\sim 5\times$ and decreases chip area by $\sim 11\times$ comparing to the origin.

2 Method

In this section, we present GSQ-Tuning, a fully quantized training method for on-device LLM fine-tuning. We begin by reviewing the fundamentals of LLM PEFT, highlighting the bottlenecks of implementing existing PEFT methods on device, and then review relevant neural network quantization literature (Sec.2.1). Building on these insights, we propose a new LLM fine-tuning framework—*Group-Shared Exponents Integer in Fully Quantized Training*—for on-device scenarios. To enable this framework, we design two key components: (1) A *Group-Shared Exponents Integer data format* to replace floating-point representations (Sec.2.2). (2) A *Fully Quantized Fine-tuning Framework* that leverages our new data format (Sec.2.3). Finally, we explore the performance–efficiency trade-off in GSQ-Tuning via Pareto frontier analysis (Sec.2.4) , providing practical guidance for its use.

2.1 Preliminaries

Low-rank Adaptation. LoRA (Hu et al., 2021) is a milestone method that injects trainable low-rank adapters into linear layers, allowing efficient fine-tuning while keeping the original parameters unchanged. Specifically, a LoRA linear layer is parameterized by a non-trainable weight matrix $\mathbf{W} \in \mathbb{R}^{oc\times ic}$, along with trainable components $\mathbf{A} \in \mathbb{R}^{r\times ic}$ and $\mathbf{B} \in \mathbb{R}^{oc\times r}$, where r is a small integer. The input $\mathbf{X} \in \mathbb{R}^{b\times ic}$ and output $\mathbf{Y} \in \mathbb{R}^{b\times oc}$ correspond to a linear layer with $oc \times ic$ processing a batch of size b. Building on LoRA, QLoRA integrates it with 4-bit NormalFloat (NF4) quantization and Double Quantization (DQ)techniques, enabling the fine-tuning of a 65B parameter model on a single 48GB GPU with minimal performance loss. In this paper, due to the memory constraint of on-device PEFT, we adopt QLoRA to quantize the weights of LLMs. The formulation is:

$$\mathbf{Y} = \mathbf{X}DQ(\mathbf{W}^{\mathbf{NF4}})^{\mathbf{T}} + \mathbf{X}\mathbf{A}^{\mathbf{T}}\mathbf{B}^{\mathbf{T}}$$

where we omit the transpose for similarity, $NF4$ means the 4 bit NormalFloat (NF) data type and

and DQ is *Double Quantization* operation to map weights from NF4 to BF16 in QLoRA. The low-rank components $\mathbf{A}$ and $\mathbf{B}$ and input $\mathbf{X}$ remain in BF16 during fine-tuning process. However, QLoRA still does not suit on-device PEFT scenarios because the low-rank term XA^TB^T remains in BF16, whereas most on-device hardware only supports integer operations. This limitation motivates our development of a new LoRA method that relies exclusively on integer operations.

Quantization. Quantization (Jacob et al., 2018b) is a crucial technique that maps a floating-point number to a discrete interval using integer values. Given a floating-point (FP) tensor $\mathbf{x}$ (such as weights, activations or gradients), the b-bits formulation is:

$$\mathbf{x_{int}} = Q(\mathbf{x}) = \text{clamp}\left(\left\lfloor \frac{\mathbf{x}}{s} \right\rceil + z, 0, 2^b - 1\right)$$

where $s = \frac{\max(|\mathbf{x}|)}{2^{b-1}-1}$ is the scaling factor, z denotes zero points, $\lfloor \cdot \rceil$ refers to the round-to-nearest, and the function $\text{clamp}(\cdot)$ clips values outside the integer range $[q_{\min}, q_{\max}]$ respectively. $[q_{\min}, q_{\max}]$ is the quantization range determined by the bit width b, where $q_{\min} = -sz$ and $q_{\max} = -s(2^b - 1 - z)$.

Fully Quantized Training (FQT). FQT involves quantizing all tensors—weights, activations, and gradients—needed for computation-intensive operations (like matrix multiplication) during both forward and backward propagation (Wang et al., 2018; Yang et al., 2020; Zhu et al., 2020). When the network's weights, activations, and gradients are each quantized to 8 bits, this is referred to as W8A8G8 quantization. Notably, FQT is different from Quantization-Aware Training (QAT), we also discuss the difference in Sec. A.1.

2.2 Group-Shared Exponents Integer

Low-bitwidth Floating Point (FP). Floating-point numbers are a commonly used data representation in deep learning. For instance, FP16 represents each number using 16 bits. Recently, lower-bit floating-point representations, such as FP8, have been introduced into the training processes of deep learning models (Micikevicius et al., 2022; Baalen et al., 2023). FP8 operates in two modes: the E4M3 and E5M2 formats. In these formats, E represents the number of exponent bits and M denotes that of mantissa bits.

Similar to quantization methods, low-bitwidth FP formats can effectively reduce memory storage requirements and decrease the hardware area and energy consumption of computational units.

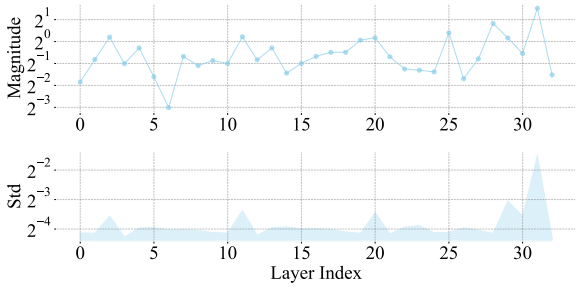

Figure 1: **In each layer, the weights' magnitudes are similar.** The standard deviations of weights across layers are less than 2^{-2} by 3-σ (about probability 99.7%). The weights are from Vicuna-7B-v1.5.

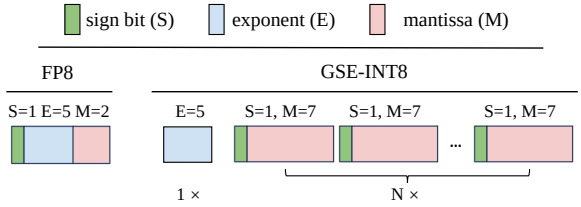

Figure 2: **The GSE format is memory efficient through group-shared exponent bits.** Comparison between FP8 and GSE-Int8.

However, we observe that low-bitwidth FP may not be the optimal solution for LoRA fine-tuning in large-scale models, primarily due to the following 3 reasons: (1) Neural network tensors exhibit spatial locality, meaning that adjacent elements within a tensor tend to have similar magnitudes, leading to redundancy in the exponent bits of FP representations. As illustrated in Fig. 1, the standard deviation of the values in the weight tensor is considerably lower than the magnitude of the values, indicating a small local variation; (2) The limited number of mantissa bits in low-bitwidth FP formats constrains precision, potentially impairing model performance. For instance, the E5M2 format, which has only two mantissa bits, is incapable of representing certain integers below ten, such as 5, 7, and 9; (3) FP computation demonstrates less efficiency in memory, chip area, and power consumption compared to GSE-INT computation, making it less suitable for resource-constrained environments. As shown in Table 7, FP formats incur considerably higher costs in power and chip area compared to integer-based computation, making them less suitable for edge environments.

Due to the inherent characteristics of FP representations and the requirement for relatively high precision in training, it is crucial to explore other data format to reduce both hardware area and energy consumption in resource-constrained cases.

Group-Shared Exponents Integer (GSE-INT). Inspired with block FP (Zhang et al., 2022), we propose the Group-Shared Exponents Integer (GSE) format as an alternative to FP formats for matrix multiplication in both forward-propagation and back-propagation. This format is also used for storing activations required by back-propagation to reduce memory consumption. As illustrated in Fig. 2, GSE introduces the following key modifications compared to traditional floating-point formats: (1) To leverage the locality of tensor values, we share the exponent across a group of N numbers. That is, all N numbers within the group use the same exponent. (2) The number of bits used for the shared exponent is fixed at 5. (3) The implicit leading 1 in floating-point representations is removed and replaced with a standard integer representation. The numerical representation in GSE is:

$$x = (-1)^s \cdot 2^e \cdot m$$

where s is sign, e is the exponent value (For simplicity, we omit the exponent bias), m is the mantissa value. The GSE format is memory efficient through sharing exponent bits. Memory for FP is $N(E + M + 1)$ and memory for GSE is $N(M + 1) + E$. As the group size N increases, the memory savings grow proportionally, while the overhead of the shared exponent is negligible.

Matrix Multiplication using GSE. Consider two vectors, $\mathbf{A}$ and $\mathbf{B}$, both represented using the GSE format and having a length of N. The dot product of the two vectors can be computed as:

$$y = 2^{e_A + e_B} \underbrace{\sum_{i=1}^{N} (-1)^{s_A \oplus s_B} m_{A,i} m_{B,i}}_{\text{standard integer multiply-accumulate}},$$

where $m_{A,i}$ and $m_{B,i}$ are the integer mantissas of the i-th elements of the vectors. The computation involves a standard integer multiply-accumulate (MAC) operation, followed by scaling with the combined exponent $2^{e_A+e_B}$.

The dot product operation can be extended to large-scale matrix multiplication. For two matrices $\mathbf{X}$ and $\mathbf{Y}$, we partition the data into groups of size N. Specifically, rows of $\mathbf{X}$ are grouped along their elements, with each group sharing a single exponent, and columns of $\mathbf{Y}$ are grouped similarly. This grouping strategy simplifies hardware implementation and makes the GSE format a practical and efficient choice for large-scale matrix operations.

Transform from FP to GSE. The transformation from FP representation to GSE format is efficient due to the design of GSE. First, within a group of N FP numbers, identify the largest exponent $e_{\max}$ among them. Then, using $e_{\max}$ as the shared exponent for the group. For each FP value in the group, its mantissa is adjusted by adding the implicit leading bit (if applicable) and then right-shifting the value based on the difference between its original exponent and $e_{\max}$. This process ensures that all values are aligned to the shared exponent, leading the storage and computation efficiency while preserving precision.

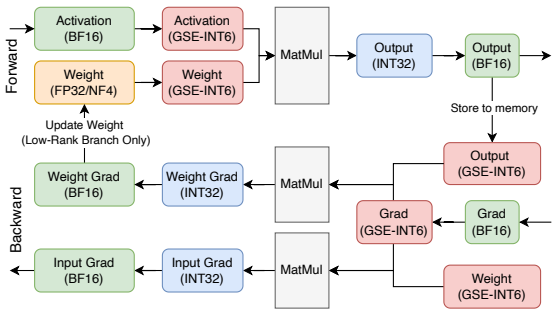

Figure 3: Dataflow of GSQ-Tuning. The weight is NF4 in full-rank branch and is FP32 in low-rank branch.

2.3 Fully Quantized Fine-tuning

As illustrated in Fig. 3, our GSQ-Tuning framework introduces a hardware-efficient quantization pipeline. Compared to QLoRA, we fully quantize weights, activations, and gradients to low-bit integers. While QLoRA primarily focuses on 4-bit quantization of frozen base model weights (NF4) while keeping adapters in high precision (BF16), our approach achieves superior computational and memory efficiency. Building on the *quantize-compute-dequantize* (QCD) paradigm for low-precision matrix multiplication (MM) (Xi et al., 2024), the QCD approach operates in three stages: (1) *Quantization*: Convert high-precision inputs matrices (e.g., BF16) to low-precision (e.g., GSE-INT6) using a quantizer $Q(\cdot)$; (2) *Computation in low-precision MM*: Perform low-precision MM to produce an intermediate output (e.g., GSE-INT6); and (3) *Dequantization*: Convert output back to high-precision using a dequantizer $Q^{-1}(\cdot)$.

Forward Propagation. The forward propagation for a linear layer is calculated as follows:

$$\mathbf{Y}^{\mathbf{BF16}} = \underbrace{Q^{-1}\left(Q(\mathbf{X}^{\mathbf{BF16}})Q\left(DQ(\mathbf{W}^{\mathbf{NF4}})\right)^T\right)}_{\text{frozen base model}} + \underbrace{Q^{-1}\left(Q(\mathbf{X}^{\mathbf{BF16}})Q(\mathbf{A}^{\mathbf{BF16}})^T Q(\mathbf{B}^{\mathbf{BF16}})^T\right)}_{\text{trainable adapter}}$$

Backward Propagation. Gradients are computed directly on quantized tensors using back propagation and chain rule:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = Q^{-1}(Q(\mathbf{B})^T Q\left(\frac{\partial \mathcal{L}}{\partial \mathbf{Y}}\right)^T Q(\mathbf{X}))$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{B}} = Q^{-1}(Q\left(\frac{\partial \mathcal{L}}{\partial \mathbf{Y}}\right)^T Q(\mathbf{X})Q(\mathbf{A})^T)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = Q^{-1}(Q\left(\frac{\partial \mathcal{L}}{\partial \mathbf{Y}}\right)(Q(\mathbf{W}) + Q(\mathbf{B})Q(\mathbf{A})))$$

2.4 Pareto Frontier for Quantization Bits and Low-rank.

Co-optimization Principle for Model Bits and Rank. The memory footprint and FLOPs during fine-tuning exhibit strong dependence on both quantization bit-width and LoRA rank $\mathcal{O}(b \cdot r)$ scaling. Excessive values in either dimension impose prohibitive computational burdens: (1) *Memory*:Mem $\propto b \cdot r$ (adapter parameter storage); (2) *Compute*: Flops $\propto r \cdot d^2$ (for hidden dimension d). This necessitates joint optimization of (b, r) to guide the accuracy-efficiency trade-off space effectively. Pure bit-width reduction sacrifices model capacity, while unrestrained rank scaling inflates computation costs disproportionately.

The effectiveness of GSQ-Tuning hinges on how quantization bit-width interacts with the dimensions of low-rank adapters. To inform real-world deployments, we systematically analyze this interplay by constructing a Pareto frontier that illustrates the balance between model memory consumption during fine-tuning and accuracy across various bits-rank settings. We hope our findings not only highlight optimal configurations, but also offer practical guidelines for practitioners to tailor solutions to specific hardware constraints.

Pareto Frontier Analysis. Based on our GSQ-Tuning, we construct a Pareto frontier by plotting model memory during fine-tuning against validation accuracy across different *bits-rank* configuration. As shown in Fig. 4, the frontier reveals three distinct optimization regimes (1) *High-Bit Low-Rank Regime (8-bit, r=64):* Reaches 65.60 Acc with suboptimal efficiency. 0.50 Acc gain from $r = 16$ to 64 indicates high-bit quantization inherently limits error magnitude, requiring less rank compensation. (2) *Mid-Bit Balanced Regime (6-bit, r=128):* Delivers 65.58 Acc with moderate resources. 0.71 Acc gain from $r = 16$ to 128 shows diminishing returns beyond this point (only

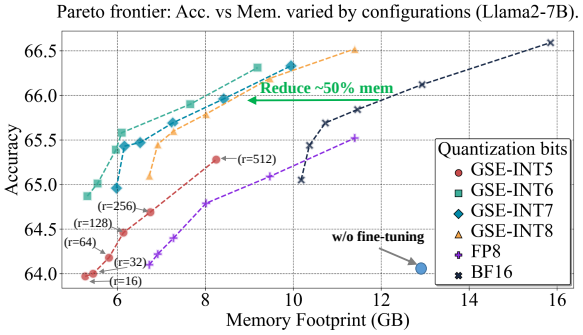

Figure 4: Pareto curve of accuracy-memory trade-offs. Compared to FP16, our GSQ-Tuning can reduce $\sim 50\%$ memory usage while having the comparable accuracy. Detailed results are in Tab.10

0.32 Acc gain from $r = 128$ to 512). (3) *Low-Bit High-Rank Regime (5-bit, r=512):* Achieves 64.88 Acc with minimal memory footprint. 0.91 Acc gain from $r = 16$ to 512 demonstrates that aggressive rank scaling can compensate for severe quantization errors. Besides, Compared to FP8, GSQ-Tuning offers higher finetuning performance at similar or lower memory (Table 2) and reduces chip area ($\sim$11×) and power ($\sim$5×, Table 7). We also provide the Pareto frontier and detailed results for other models in appendix. Extra extensive results yield similar guidance.

3 Experiments

Foundation Models and Evaluation Metrics. We apply our method to the entire LLaMA family, including LLaMA-2 (7B/13B/70B)(Touvron et al., 2023b), and LLaMA-3 (3B-8B). We evaluate the fine-tuning models on up to 9 zero-shot commonsense question-answering (CSQA) tasks using the `lm-evaluation-harness` (version 0.4.7)(Gao et al., 2024), including BoolQ(Clark et al., 2019), HellaSwag (Zellers et al., 2019), LAMBADA (OpenAI)(Radford et al., 2019), OpenBookQA(Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), WinoGrande (Sakaguchi et al., 2019), ARC-Easy, and ARC-Challenge (Clark et al., 2018). The fine-tuning dataset follows Alpaca (Taori et al., 2023), with 52K instruction data from text-davinci-003. Besides, we also report the memory cost (Mem.(G)) during the model fine-tuning process.

Training Details. We employ the whole fine-tuning process based on LLaMA-Factory (Zheng et al., 2024). We implement GSQ-Tuning in `PyTorch` using models from `Hugging Face`. We freeze the parameters of the linear modules and

Table 1: 0-shot CSQA accuracy comparison with respect to different quantization bits in 64 rank setting. '4-8-8' means quantize weights, activation and gradients to 4-bit, 8-bit and 8-bit. GSQ-Tuning is built on Qlora, where all weights are quantized as NF4 firstly.Notably, the memory usage for LLaMA series model is for the model weights alone. Since these models are not fine-tuned, no fine-tuning phase (no gradient computation or updates) is involved. Denoted as 16-16-None for weights-activation-gradients.

Method	LLMs branch	low-rank branch	**Avg.**	**ARC-c**	**ARC-e**	**BoolQ**	**HellaS.**	**OBQA**	**PIQA**	**SCIQ.**	**WinoG.**	**Mem.(G)**
LLaMA2-7B	16-16-None	w/o	64.13	46.25	74.62	77.68	76.01	44.20	79.11	46.11	69.06	13.20
w/ QLoRA	4-16-16	16-16-16	65.69	47.14	74.75	79.50	76.46	45.50	79.63	50.26	71.32	10.73
	4-8-8	8-8-8	**65.60**	48.12	74.24	79.72	76.00	45.80	79.60	49.69	71.67	7.28
w/ GSQ-Tuning	4-6-6	6-6-6	**65.39**	47.70	74.58	79.24	76.05	44.60	79.60	50.41	70.96	5.97
	4-5-5	5-5-5	**64.18**	45.14	72.69	75.20	75.27	46.40	79.65	48.62	70.48	5.81
LLaMA2-13B	16-16-None	w/o	66.65	48.81	76.47	82.45	79.67	44.80	80.36	48.31	72.38	25.7
w/ QLoRA	4-16-16	16-16-16	67.61	49.66	77.23	83.30	78.95	45.40	80.74	51.59	73.24	17.42
	4-8-8	8-8-8	**67.48**	49.57	77.40	82.87	78.88	46.20	80.90	50.72	73.32	11.99
w/ GSQ-Tuning	4-6-6	6-6-6	**67.35**	49.66	77.27	82.75	78.66	79.05	80.90	50.97	73.16	10.89
	4-5-5	5-5-5	**66.97**	49.91	76.60	81.87	78.15	46.20	80.41	49.54	73.09	10.33
LLaMA2-70B	16-16-None	w/o	70.68	56.91	80.05	85.78	83.59	48.60	82.48	48.67	79.40	137.42
w/ QLoRA	4-16-16	16-16-16	72.22	59.81	82.20	86.51	83.89	50.40	83.13	51.48	80.35	66.82
	4-8-8	8-8-8	**72.20**	59.90	82.32	86.51	83.90	50.20	83.08	51.59	80.11	52.17
w/ GSQ-Tuning	4-6-6	6-6-6	**72.10**	59.39	82.15	86.51	83.94	50.00	83.30	50.92	80.58	48.71
	4-5-5	5-5-5	**71.70**	58.87	81.48	85.90	83.91	49.60	82.81	50.67	80.43	46.98
LLaMA3-3B	16-16-None	w/o	62.64	45.90	71.68	73.00	73.64	43.20	77.42	47.08	69.22	6.42
w/ QLoRA	4-16-16	16-16-16	64.11	48.63	74.07	77.22	73.12	41.60	78.51	49.33	70.40	6.78
	4-8-8	8-8-8	**64.02**	47.30	74.44	77.26	72.51	42.40	78.60	49.80	69.86	3.93
w/ GSQ-Tuning	4-6-6	6-6-6	**63.71**	47.07	73.59	76.64	72.23	41.80	78.20	48.76	71.36	3.47
	4-5-5	5-5-5	**62.74**	48.04	73.44	73.36	71.96	40.40	78.13	47.65	68.98	3.24
LLaMA3-8B	16-16-None	w/o	67.18	53.50	77.74	81.13	79.20	45.00	80.63	47.03	73.24	15.01
w/ QLoRA	4-16-16	16-16-16	68.45	55.63	80.13	83.67	78.78	44.80	81.28	50.41	72.93	11.64
	4-8-8	8-8-8	**68.61**	55.97	80.22	83.61	78.68	45.20	81.50	50.41	73.32	7.63
w/ GSQ-Tuning	4-6-6	6-6-6	**68.22**	55.55	79.29	83.67	78.47	44.80	80.90	50.05	73.09	6.86
	4-5-5	5-5-5	**66.69**	54.10	77.99	81.65	77.12	43.80	79.54	47.90	71.43	6.47

update a smaller (low-rank) set of parameters during the fine-tuning. The group size of our GSQ-Tuning is 32. We fine-tune models using the 8-bits AdamW optimizer (Dettmers et al.) in bfloat16 precision. We choose the constant learning rate schedule and set the learning rate to be 1×10^{-5} for all models. In all cases, we tune the hyper-parameters on the base BF16 tasks, and re-use the same values for low-precision training. We always perform single-epoch experiments using a linear learning rate warm-up of 100 steps. The batch size and sequence length are fixed at 16 and 2048. The number of fine-tuning steps is 3.24K for Alpaca.

Hardware Synthesis. We implemented the hardware in Verilog RTL and synthesized it using Synopsys Design Compiler with a 7nm technology library to estimate the process engine's area, latency, and power consumption (Clark et al., 2016). The hardware operates at 1 GHz and has a capability of 50 TOPS. The memory subsystem is not considered in our settings and analyses.

3.1 Overall Results

GSQ-Tuning Results on LLaMA Family. Here, we compare the fine-tuning performance across LLaMA family (3B 70B) against QLoRA. As shown in Tab. 1, GSQ-Tuning achieves comparable or better zero-shot accuracy across different LLaMA model scales (7B-70B) under a fully low-bit quantization fine-tuing setting. With 8-bit quantization precision (W8A8G8), GSQ-Tuning matches or exceeds QLoRA's performance on 83% of tasks, despite using 50% fewer bits for activations and gradients. Even at aggressive 5-bit quantization (W5A5G5), the GSQ-Tuning maintains 98.6% of QLoRA's average accuracy while reducing memory footprint by 2.67×. These results confirm GSQ-Tuning's effectiveness for resource-constrained edge deployment.

Comparison with FP8. Here, we compare the designed GSE data format with FP8 in fully quantized fine-tuning framework. As shown in Tab. 2, the results demonstrate that the designed GSE implemented in our GSQ-Tuning method achieves superior fine-tuning performance compared to FP8 while significantly reducing computation efficiency. Even under 5-bit settings, GSQ-Tuning maintains fine-tuning performance on par with FP8, further validating its effectiveness. Additionally, to mitigate the impact of rank variations, we report the fine-tuning results using 64-rank setting in Tab. 15

Table 2: 0-shot CSQA accuracy comparison with FP8 in different quantization bits in 32 rank setting.

Method	LLMs branch	low-rank branch	Avg.	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	SCIQ.	WinoG.	Mem.(G)
LLaMA2-7B	16-16-None	w/o	64.13	46.25	74.62	77.68	76.01	44.20	79.11	46.11	69.06	13.2
w/ QLoRA	4-16-16	16-16-16	65.24	47.27	75.04	78.87	76.11	44.60	79.76	49.95	70.32	9.37
w/ FP8	4-8-8	8-8-8	64.09	45.90	73.27	77.86	75.83	45.40	78.45	46.98	69.06	6.52
w/ GSQ-Tuning	4-8-8	8-8-8	**65.45**	48.12	74.71	78.38	76.14	46.00	79.71	49.64	70.96	6.52
	4-5-5	5-5-5	**64.00**	44.97	73.32	75.29	74.95	44.60	79.27	48.93	70.24	5.45
LLaMA3-8B	16-16-None	w/o	67.18	53.50	77.74	81.13	79.20	45.00	80.63	47.03	73.24	15.01
w/ QLoRA	4-16-16	16-16-16	68.31	55.55	80.39	83.36	78.65	44.60	81.28	50.05	72.61	11.02
w/ FP8	4-8-8	8-8-8	66.62	50.77	76.43	81.59	78.17	43.80	80.20	47.44	74.59	7.23
w/ GSQ-Tuning	4-8-8	8-8-8	**68.45**	55.72	80.22	83.43	78.60	45.00	81.18	50.20	73.32	7.23
	4-5-5	5-5-5	**66.48**	51.71	77.69	82.11	76.91	44.20	79.43	48.16	71.67	6.07

Table 3: Cross-modal task evaluation on LLaVA-v1.5-7B: the default setting is QLoRA on 4-bits/64-rank without finetuning. Shared exponents shows robustness to LLM's finetuning, referring to the comparison with BF16.

Settings	LLMs branch	low-rank branch	POPE-random			POPE-adversarial			TextVQA	MMBench
			accuracy	precision	F1-score	accuracy	precision	F1-score		
LLaVA-v1.5-7B	16-16-None	w/o	84.19	84.08	84.80	74.40	69.96	76.96	6.51	55.45
w/ QLoRA	4-16-16	16-16-16	87.47	92.51	86.68	83.87	85.52	83.48	47.68	67.08
w/ GSQ-Tuning	4-8-8	4-8-8	87.84	96.21	87.08	84.03	87.40	83.28	45.19	69.75
	4-6-6	4-6-6	88.08	96.08	87.39	83.43	85.80	82.87	49.13	70.19

Table 4: 0-shot CSQA accuracy on CS170K dataset in 64 rank setting.

Method	LLMs branch	low-rank branch	Avg.	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	SCIQ.	WinoG.
LLaMA2-7B	16-16-None	w/o	64.13	46.25	74.62	77.68	76.01	44.20	79.11	46.11	69.06
w/ QLoRA	4-16-16	16-16-16	67.78	51.79	78.86	81.28	75.77	46.00	79.98	53.89	75.37
w/ GSQ-Tuning	4-8-8	8-8-8	**67.73**	51.79	78.37	82.32	75.74	46.20	79.27	53.12	75.06
	4-6-6	6-6-6	**67.56**	50.77	77.74	82.23	75.07	47.40	79.60	53.58	74.11

of the appendix. Extensive experiments consistently support the advantages of our approach.

Table 5: Fine-tuning time comparison with full-finetuning and low-rank adapter on alpaca-52K dataset in LLaMA2-7B model.

Method	LLMs branch	low-rank branch	Avg.	Ft-Time.(h)	Mem.(G)
LLaMA2-7B	16-16-None	w/o	64.13	0.0	13.20
w/ Full-finetuing	16-16-16	16-16-16	65.88	2.4	58.56
w/ QLoRA	4-16-16	16-16-16	65.69	3.2	10.73
w/ GSQ-Tuning	4-6-6	6-6-6	**65.39**	3.4	5.97

Fine-tuning Time Comparison. Here, we compare the fine-tuning time with QLoRA, full-finetune, and our proposed GSQ-Tuning on the LLaMA2-7B model. As shown in Tab. 5, the results demonstrate that full fine-tuning achieves 65.88 accuracy in 2.4 hours but requires up to a typically infeasible 58.56GB memory 10× our GSQ-Tuning's 5.97GB and 5× QLoRA's 10.73GB. GSQ-Tuning (GSE-INT6) and QLoRA (BF16+NF4) both converge in 3.2 3.4 hours with matching accuracy (65.39% vs 65.69%), yet GSQ-Tuning uses nearly half QLoRA's memory. It verifies that GSQ-Tuning matches QLoRA's training time while offering superior memory efficiency, in resource-constrained edge scenarios where our primary focus. Although PEFT methods like ours inherently require longer convergence times, our primary focus lies in striking a balance between memory efficiency and computational feasibility.

Table 6: Fine-tuning training memory (GB) cost with different fine-tuning settings in the LLaMA family.

Setting	LLaMA2-7B	LLaMA2-13B	LLaMA2-70B	LLaMA3-3B	LLaMA3-8B
Full-finetuing	58.56	112.80	597.74	28.12	65.86
w/ QLoRA	10.73	17.42	66.82	6.78	11.64
w/ GSQ-Tuning	**5.97**	**10.89**	**48.71**	**3.47**	**6.86**

Fine-tuning Memory Comparison. Here, as shown in Table 6, we reported the training memory cost for these baselines when training/full fine-tuning (batch size = 1, sequence length=2048), no activation checkpointing and memory offloading as shown in table R1. However, their huge training memory costs (>10x more than GSQ-Tuning) outside our edge-focused fine-tuning scope, where full finetuing/training is typically infeasible due to resource constraints.

Hardware Efficiency Analysis. Table 7 compares the chip area and power consumption of different formats through hardware synthesis. GSE-INT format demonstrates significant advantages over FP: (1) Area Efficiency: GSE-INT6 process engine requires only 0.47mm^2, 10.7× smaller than

Table 7: Comparison of hardware overhead between FP process engine and GSE-INT process engine (7nm).

Format	Area (mm^2)↓	Power (W) ↓
FP8 (E5M2)	4.36	2.53
FP8 (E4M3)	5.06	3.23
FP7 (E3M3)	5.05	2.75
FP6 (E3M2)	3.40	2.09
GSE-INT8	0.85	1.24
GSE-INT7	0.61	1.00
GSE-INT6	0.47	0.76
GSE-INT5	**0.39**	**0.53**

FP8(E4M3). This stems from simplified integer arithmetic logic and group-wise exponent sharing that eliminates complex alignment. (2) Power Superiority: At comparable bit-widths, GSE-INT6 consumes 0.76W (the 23.52% of FP8's 3.23W).

3.2 Generalization Experiments

Generalization of GSQ-Tuning for Vision-Language Model (LLaVA). Model used is LLaVA-v1.5-7B (Liu et al., 2024b) with Vicuna-7B-v1.5 (Zheng et al., 2023) as language model and CLIP ViT-L-336px (Radford et al., 2021) as vision tower, connected by a 2-layer MLP. Instruction dataset and other settings for finetuning follow the LLaVA official repository, LLaVA-Instruction (Liu et al., 2024c) and the improved one (Liu et al., 2024b). Tab. 3 shows performance drop of the vanilla quantization of 4-bits/64-rank QLoRA, especially, referring to the TextVQA evaluation. Fine-tuning with GSE shows comparable performance compared to that with BF16. BF16 is of E8M7 while GSE is of E5M7, demonstrating the redundancy of the dynamic range *w.r.t.* exponents is at least 3-bits much. Moreover, the memory cost of GSE is about a half of BF16.

Generalization of GSQ-Tuning on Other Fine-tune Dataset. Here, we also select Commonsense170K (CS170K) (Hu et al., 2023) to evaluate the generalization ability of GSQ-Tuning across different fine-tuing dataset. CS170K is a dataset constructed from the training sets of BoolQ, PIQA, SIQA, HellaSwag, WinoGrande, ARC-e, ARC-c, and OBQA with pre-defined templates, comprising 170K commonsense reasoning samples. As shown in Tab. 4, on larger fine-tuning datasets, our GSQ-Tuning also demonstrates comparable or even superior accuracy compared to QLoRA, while being more computationally efficient.

3.3 Ablation Study

Group Size Analysis. As shown in Tab. 8, our GSQ-Tuning with 32 groups achieves optimal accuracy-efficiency balance in 6-bit configurations (W6A6G6). The 32-group setting yields significantly higher average accuracy (65.39) compared to 64-group (64.72) and 128-group (64.27) variants, while maintaining comparable memory efficiency (70.96 vs 69.22/69.53). This sweet spot emerges from the tension between quantization bit width and hardware deployment - smaller groups better capture value distributions but increase computation overhead, while larger groups sacrifice adaptation granularity. We therefore adopt group=32 as the default configuration.

Table 8: The effect of the number of shared group on fine-tuning performance in 64 rank setting.

Method	LLMs branch	low-rank branch	**Group**	**Avg.**	**Mem. (G)**
LLaMA2-7B	16-16-None	w/o	-	64.13	13.2
w/ GSQ-Tuning	5-6-6	6-6-6	32	**65.39**	6.17
			64	64.72	6.32
			128	64.27	6.56

Table 9: The ablation of the exponent bit width (E) and the number of shared group (N) on fine-tuning performance in 64 rank setting.

Method	LLMs branch	low-rank branch	**Group**	**Avg.**	**Mem. (G)**
LLaMA2-7B	16-16-None	w/o	-	64.13	13.2
w/ GSQ-Tuning	4-6-6	6-6-6	32	62.85	6.29
			64	62.66	6.16
	5-6-6	6-6-6	32	**65.39**	6.32
			64	64.72	6.17
	6-6-6	6-6-6	32	65.41	6.35
			64	64.99	6.18

Exponenet Bits Analysis. As shown in Tab. 9, experiments demonstrate that fixing exponent bits E=5 achieves the optimal trade-off between accuracy and memory overhead. Adopting a smaller exponent (for example, E = 4) leads to significant performance degradation, while increasing E=6 yields marginal performance gains. This ablation study validates the effectiveness of our fixed shared exponent design E = 5.

4 Related Work

Parameter-Efficient Fine-Tuning (PEFT). PEFT reduces memory and computational costs by introducing a small set of trainable parameters while keeping the pretrained model frozen. Approaches including soft prompt tuning (Wang et al., 2023), partial fine-tuning (Fu et al., 2023), and low-rank adaptation (Hu et al., 2021). Among these, LoRA stands out as a seminal work, injecting trainable low-rank matrices into linear layers to enable efficient fine-tuning without modifying the base model weights. QLoRA

extends this with 4-bit NF4 quantization and Double Quantization, supporting 65B model fine-tuning on a single 48GB GPU with minimal performance degradation. Recent work further improves quantization-aware fine-tuning (Xu et al., 2023; Li et al.) and extends LoRA for better efficiency, stability, and performance (Hu et al., 2023; Liu et al., 2024d; Zhao et al., 2024; Hayou et al., 2024; Meng et al., 2024).

Quantization. Much works (Yuan et al., 2023; Yuan et al.; Hu et al., 2024, 2025b,a) make efforts to accelerate the LLMs. For instance, GPTQ (Frantar et al., 2022) quantizes weights to 3-4 bit with slight accuracy drop based on approximate second-order information. AWQ (Lin et al., 2024) and SmoothQuant (Xiao et al., 2022) explore the scheme of smoothing by detecting the importance of different activation channels. Recent works (e.g., Quarot (Ashkboos et al., 2024), SpinQuant (Liu et al., 2024e)) further suppress outliers by utilizing computation-invariant rotation transformation. However, above methods focus on the inference optimization. Leveraging the key benefits of FQT, several studies (Banner et al., 2018; Wu et al., 2018; Langroudi et al., 2019; Yang et al., 2020; Zhu et al., 2020; Xi et al., 2023) have explored its implementation and optimization. However, challenges remain, especially during the backward pass due to the wide dynamic range of gradients. For example, LM-FP8 (Peng et al., 2023) trains LLMs from scratch using FP8, achieving performance comparable to BF16. Some studies have also explored full integer quantization, utilizing efficient hardware implementations. SwitchBack (Wortsman et al., 2023) quantizes partial matrix multiplications with INT8, but it is limited to vision models with up to 1B parameters. Jetfire (Xi et al., 2024) proposes a 2D block-wise quantization approach that maintains accuracy and achieves significant memory savings (1.4-1.5x) when training in INT8. However, these methods have not yet been explored to fine-tuning tasks, and using FQT to lower-bit ($<$ INT8) poses considerable challenges.

5 Conclusion

In this paper, we propose GSQ-Tuning, a resource-efficient framework that addresses the critical challenges of floating-point dependency, privacy risks, and hardware incompatibility in on-device LLM fine-tuning. By integrating Group-Shared Exponents Integer (GSE) quantization with parameter-efficient adaptation, our method achieves three key advancements:(1) Full Integer Pipeline: Eliminates floating-point operations across both forward and backward passes, reducing memory usage by 50% compared to FP16 while maintaining comparable accuracy. (2) Hardware-Optimized Design: The GSE format reduces metadata overhead via group-wise exponent sharing, enabling 5-8bit integer representations. Combined with LoRA-like adapters, this achieves 5 $\times$ lower power consumption and 11$\times$ smaller chip area compared to FP8 at equivalent accuracy levels. (3) Practical Deployment Guidance: A Pareto frontier analysis guides optimal bit-rank configurations for diverse edge constraints. These innovations establish GSQ-Tuning as a foundational step toward democratizing LLM adaptation for resource-constrained environments. This breakthrough makes private, on-device LLM adaptation practical for sensitive applications. Future work will explore sub-4bit quantization to further push the boundaries of edge AI.

6 Limitations

While our GSQ-Tuning significantly advances on-device LLM adaptation through integer-focused optimization and parameter-efficient quantization, two key limitations warrant discussion:

Non-linear Operator Precision. Our current implementation maintains non-linear operations (e.g., LayerNorm, Softmax) in 16-bit to preserve numerical stability. This introduces partial precision conversion overhead during computation. However, non-linear operations do not contain additional learnable parameters and thus do not consume memory. Moreover, these non-linear operations are generally computation-light, making their computational burden negligible. Future work could explore fully integer implementations for non-linear layers.

Bit-Width Range Constraints. The current framework operates effectively in 5-8bit configurations but didn't present the performance at extreme low bit ($\leq$ 4bit) precision. This stems from gradient direction distortion under extreme quantization—a challenge requiring new error compensation mechanisms. We plan to investigate two directions: (1) 4bit stochastic rounding with gradient-aware scaling, and (2) mixed-precision adapters allocating higher bits to critical gradient dimensions.

Furthermore, future work could explore (1) full integer fine-tuning, (2) low-bit quantized fine-tuning and (3) co-design with emerging integer-optimized AI accelerators.

References

ARM. 2020. Cortex-m. In *https://developer.arm.com/processors/cortex-m*.

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

Mart van Baalen, Andrey Kuzmin, Suparna S Nair, Yuwei Ren, Eric Mahurin, Chirag Patel, Sundar Subramanian, Sanghyuk Lee, Markus Nagel, Joseph Soriaga, et al. 2023. Fp8 versus int8 for efficient deep learning inference. *arXiv preprint arXiv:2303.17951*.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2020. Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*.

Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. 2018. Scalable methods for 8-bit training of neural networks. In *Advances in Neural Information Processing Systems*, pages 5145–5153.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Lawrence T Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. 2016. Asap7: A 7-nm finfet predictive process design kit. *Microelectronics Journal*, 53:105–115.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115.

Zhen Dong, Zhewei Yao, Yaohui Cai, Daiyaan Arfeen, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2019a. Hawq-v2: Hessian aware trace-weighted quantization of neural networks. *arXiv preprint arXiv:1911.03852*.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. 2019b. Hawq: Hessian aware quantization of neural networks with mixed-precision. pages 293–302.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. 2019. Learned step size quantization. In *International Conference on Learning Representations*.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

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, Haoran Yang, Anthony Man-Cho So, Wai Lam, Lidong Bing, and Nigel Collier. 2023. On the effectiveness of parameter-efficient fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12799–12807.

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. 2024. A framework for few-shot language model evaluation.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. 2019. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4852–4861.

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. 2024. Lora+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*.

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*.

Xing Hu, Zhixuan Chen, Dawei Yang, Zukang Xu, Chen Xu, Zhihang Yuan, Sifan Zhou, and Jiangyong Yu. 2025a. Moequant: Enhancing quantization for mixture-of-experts large language models via expert-balanced sampling and affinity guidance. *arXiv preprint arXiv:2505.03804*.

Xing Hu, Yuan Cheng, Dawei Yang, Zukang Xu, Zhihang Yuan, Jiangyong Yu, Chen Xu, Zhe Jiang, and Sifan Zhou. 2025b. Ostquant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. *arXiv preprint arXiv:2501.13987*.

Xing Hu, Yuan Cheng, Dawei Yang, Zhihang Yuan, Jiangyong Yu, Chen Xu, and Sifan Zhou. 2024. I-llm: Efficient integer-only inference for fully-quantized low-bit large language models. *arXiv preprint arXiv:2405.17849*.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Ka-Wei Lee. 2023. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018a. 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.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018b. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2021. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR.

Hamed F Langroudi, Zachariah Carmichael, and Dhireesha Kudithipudi. 2019. Deep learning training on the edge with low-precision posits. *arXiv preprint arXiv:1907.13216*.

Yixiao Li, Yifan Yu, Chen Liang, Nikos Karampatziakis, Pengcheng He, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. In *The Twelfth International Conference on Learning Representations*.

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2024b. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024c. Visual instruction tuning. *Advances in neural information processing systems*, 36.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024d. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023. Gpt understands, too. *AI Open*.

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

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Paulius Micikevicius, Dusan Stosic, Neil Burgess, Marius Cornea, Pradeep Dubey, Richard Grisenthwaite, Sangwon Ha, Alexander Heinecke, Patrick Judd, John Kamalu, et al. 2022. Fp8 formats for deep learning. *arXiv preprint arXiv:2209.05433*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Houwen Peng, Kan Wu, Yixuan Wei, Guoshuai Zhao, Yuxiang Yang, Ze Liu, Yifan Xiong, Ziyue Yang, Bolin Ni, Jingcheng Hu, et al. 2023. Fp8-lm: Training fp8 large language models.

QUALCOMM. 2024. Qualcomm hexagon npu. In *https://www.qualcomm.com/processors/hexagon*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020a. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020b. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

Hanlin Tang, Xipeng Zhang, Kai Liu, Jianchen Zhu, and Zhanhui Kang. 2022. Mkq-bert: Quantized bert with 4-bits weights and activations. *arXiv preprint arXiv:2203.13483*.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. 2018. Training deep neural networks with 8-bit floating point numbers. In *Advances in Neural Information Processing Systems*, pages 7675–7684.

Zhen Wang, Rameswar Panda, Leonid Karlinsky, Rogerio Feris, Huan Sun, and Yoon Kim. 2023. Multitask prompt tuning enables parameter-efficient transfer learning. *arXiv preprint arXiv:2303.02861*.

Zheng Wang, Juncheng B Li, Shuhui Qu, Florian Metze, and Emma Strubell. 2022. Squat: Sharpness-and quantization-aware training for bert. *arXiv preprint arXiv:2210.07171*.

Mitchell Wortsman, Tim Dettmers, Luke Zettlemoyer, Ari Morcos, Ali Farhadi, and Ludwig Schmidt. 2023. Stable and low-precision training for large-scale vision-language models. *Advances in Neural Information Processing Systems*, 36:10271–10298.

Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. 2018. Training and inference with integers in deep neural networks. In *International Conference on Learning Representations*.

Haocheng Xi, Yuxiang Chen, Kang Zhao, Kaijun Zheng, Jianfei Chen, and Jun Zhu. 2024. Jetfire: Efficient and accurate transformer pretraining with int8 data flow and per-block quantization. *arXiv preprint arXiv:2403.12422*.

Haocheng Xi, Changhao Li, Jianfei Chen, and Jun Zhu. 2023. Training transformers with 4-bit integers. *Advances in Neural Information Processing Systems*, 36:49146–49168.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Julien Demouth, and Song Han. 2022. Smoothquant: Accurate and efficient post-training quantization for large language models. *arXiv preprint arXiv:2211.10438*.

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*.

Yukuan Yang, Lei Deng, Shuang Wu, Tianyi Yan, Yuan Xie, and Guoqi Li. 2020. Training high-performance and large-scale deep neural networks with full 8-bit integers. *Neural Networks*, 125:70–82.

Zhihang Yuan, Lin Niu, Jiawei Liu, Wenyu Liu, Xinggang Wang, Yuzhang Shang, Guangyu Sun, Qiang Wu, Jiaxiang Wu, and Bingzhe Wu. 2023. Rptq: Reorder-based post-training quantization for large language models. *arXiv preprint arXiv:2304.01089*.

Zhihang Yuan, Yuzhang Shang, and Zhen Dong. Pb-llm: Partially binarized large language models. In *The Twelfth International Conference on Learning Representations*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 36–39. IEEE.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. 2018. LQ-Nets: Learned quantization for highly accurate and compact deep neural networks. In *The European Conference on Computer Vision (ECCV)*.

Sai Qian Zhang, Bradley McDanel, and HT Kung. 2022. Fast: Dnn training under variable precision block floating point with stochastic rounding. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 846–860. IEEE.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. 2020. Ternarybert: Distillation-aware ultra-low bit bert. *arXiv preprint arXiv:2009.12812*.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. 2017. Incremental network quantization: Towards lossless cnns with low-precision weights. *International Conference on Learning Representations*.

Feng Zhu, Ruihao Gong, Fengwei Yu, Xianglong Liu, Yanfei Wang, Zhelong Li, Xiuqi Yang, and Junjie Yan. 2020. Towards unified int8 training for convolutional neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1969–1979.

A Appendix

A.1 Differences with Quantization-aware training (QAT):

Quantization-aware training (QAT) (Choi et al., 2018; Zhang et al., 2018; Zhou et al., 2017; Jacob et al., 2018a; Dong et al., 2019b,a; Shen et al., 2020a; Zafrir et al., 2019; Shen et al., 2020b; Tang et al., 2022; Zhang et al., 2020; Bai et al., 2020; Foret et al., 2020; Wang et al., 2022) is an *inference acceleration* technique which trains networks with quantizers inserted in the forward propagation graph, so the trained network can perform efficiently during inference. QAT can compress activation/weights to extremely low precision (e.g. 1-2 bits). It is tempting to think that directly applying a quantizer for QAT to FQT can lead to similar low activation/weights bit-width. However, even only quantizing the forward propagation for FQT is much more challenging than QAT because: ❶ QAT requires a converged full-precision model as initialization (Esser et al., 2019) and/or as a teacher model for knowledge distillation (Bai et al., 2020); ❷ QAT may approximate the discrete quantizer with continuous functions during training (Gong et al., 2019), which cannot be implemented with integer arithmetic. Due to these challenges, it is still an open problem to do FQT with low-bit activations/weights.

A.2 Detailed results on different rank setting:

Here, we also report the results of our GSQ-Tuning on different LlaMA model, including LlaMA2-7B (Tab.10), LlaMA2-13B (Tab.11), LlaMA2-70B(Tab.12), LlaMA3-3B(Tab.13), and LlaMA3-8B(Tab.14). The results consistently demonstrated the effectiveness and efficiency of GSQ-Tuning.

A.3 Comparison with FP8 with 64 rank

Here, we compare the designed GSE data format with FP8 in fully quantized fine-tuning framework with 32 rank setting. As shown in Tab. 15, the results still demonstrate that the designed GSE implemented in our GSQ-Tuning method achieves superior fine-tuning performance compared to FP8 while significantly reducing computation efficiency. Even under 5-bit settings, GSQ-Tuning maintains fine-tuning performance on par with FP8, validating its effectiveness.

Table 10: 0-shot commonsense QA accuracy (%) across different bits and rank on llama2-7B.

Method	**rank**	LLMs branch	low-rank branch	**Avg.**	**ARC-c**	**ARC-e**	**BoolQ**	**HellaS.**	**OBQA**	**PIQA**	**SCIQ.**	**WinoG.**	**Mem. (G)**
LLaMA2-7B		16-16-None	w/o	64.13	46.25	74.62	77.68	76.01	44.20	79.11	46.11	69.06	13.2
w/ QLoRA	16	4-16-16	16-16-16	65.05	47.53	75.17	78.59	76.09	44.00	79.54	49.44	70.09	10.18
w/ GSQ-Tuning	16	4-8-8	8-8-8	65.10	47.53	74.71	78.35	75.99	45.00	79.65	49.28	70.32	6.73
		4-7-7	7-7-7	64.96	47.18	75.21	78.10	75.98	44.80	79.27	49.95	69.38	5.98
		4-6-6	6-6-6	64.87	46.84	73.78	78.07	75.88	45.80	79.22	49.39	70.01	5.32
		4-5-5	5-5-5	63.97	46.76	72.64	75.78	74.95	45.20	79.05	48.62	68.75	5.27
w/ QLoRA	32	4-16-16	16-16-16	65.44	47.27	75.04	78.87	76.11	44.60	79.76	49.95	70.32	10.37
w/ GSQ-Tuning	32	4-8-8	8-8-8	65.45	48.12	74.71	78.38	76.14	46.00	79.71	49.64	70.96	6.92
		4-7-7	7-7-7	65.43	47.35	74.20	78.99	75.84	46.00	79.92	49.59	71.59	6.16
		4-6-6	6-6-5	65.01	47.44	74.62	78.65	76.03	44.00	79.60	50.05	69.69	5.55
		4-5-5	5-5-5	64.00	44.97	73.32	75.29	74.95	44.60	79.27	48.93	70.24	5.45
w/ QLoRA	64	4-16-16	16-16-16	65.69	47.14	74.75	79.50	76.46	45.50	79.63	50.26	71.32	10.73
w/ GSQ-Tuning	64	4-8-8	8-8-8	65.60	48.12	74.24	79.72	76.00	45.80	79.60	49.69	71.67	7.28
		4-7-7	7-7-7	65.47	47.78	74.71	79.51	76.09	45.80	79.60	49.80	70.48	6.52
		4-6-6	6-6-6	65.39	47.70	74.58	79.24	76.05	44.60	79.60	50.41	70.96	5.97
		4-5-5	5-5-5	64.18	45.14	72.69	75.20	75.27	46.40	79.65	48.62	70.48	5.81
w/ QLoRA	128	4-16-16	16-16-16	65.84	48.24	74.91	79.78	76.27	45.52	79.77	50.48	71.79	11.46
w/ GSQ-Tuning	128	4-8-8	8-8-8	65.79	48.12	74.83	80.28	75.96	45.80	79.54	50.61	71.19	8.02
		4-7-7	7-7-7	65.69	48.04	74.87	79.79	76.08	45.00	79.49	50.61	71.67	7.26
		4-6-6	6-6-6	65.58	47.87	74.54	80.09	76.05	45.40	79.38	50.10	71.27	6.10
		4-5-5	5-5-5	64.46	46.50	72.77	75.99	75.31	46.60	79.00	48.98	70.56	6.14
w/ QLoRA	256	4-16-16	16-16-16	66.12	48.33	75.00	80.94	76.37	45.61	79.97	51.13	71.64	12.93
w/ GSQ-Tuning	256	4-8-8	8-8-8	66.19	48.55	75.13	80.76	76.14	47.00	79.38	50.72	71.82	9.47
		4-7-7	7-7-7	65.96	48.46	75.08	80.43	76.04	45.60	79.76	50.72	71.59	8.42
		4-6-6	6-6-6	65.90	48.38	74.16	79.94	75.81	46.80	79.43	50.87	71.82	7.66
		4-5-5	5-5-5	64.59	46.33	72.60	76.51	75.57	46.40	79.60	49.39	70.32	6.75
w/ QLoRA	512	4-16-16	16-16-16	66.59	49.26	75.20	81.99	76.06	46.74	79.49	51.71	72.27	15.85
w/ GSQ-Tuning	512	4-8-8	8-8-8	66.52	49.49	74.92	81.28	75.89	47.60	79.49	51.59	71.90	11.40
		4-7-7	7-7-7	66.33	48.89	74.75	81.41	76.06	47.00	79.54	51.74	71.27	9.95
		4-6-6	6-6-6	66.31	48.55	75.51	80.80	76.42	46.00	79.60	51.64	71.98	9.19
		4-5-5	5-5-5	64.86	47.44	73.15	76.85	75.62	47.00	79.33	49.18	70.32	8.25

Table 11: 0-shot commonsense QA accuracy (%) across different bits and rank on llama2-13B.

Method	**rank**	LLMs branch	low-rank branch	**Avg.**	**ARC-c**	**ARC-e**	**BoolQ**	**HellaS.**	**OBQA**	**PIQA**	**SCIQ.**	**WinoG.**	**Mem. (G)**
LLaMA2-13B	-	16-16-None	w/o	66.65	48.81	76.47	82.45	79.67	44.80	80.36	48.31	72.38	25.70
w/ QLoRA	16	4-16-16	16-16-16	67.32	49.74	76.98	82.94	78.85	46.00	80.52	50.36	73.16	16.56
w/ GSQ-Tuning	16	4-8-8	8-8-8	67.35	49.83	77.06	83.09	78.89	46.00	80.47	50.31	73.16	11.13
		4-7-7	7-7-7	67.29	49.91	76.94	83.03	78.90	45.40	80.58	50.61	73.01	10.58
		4-6-6	6-6-6	67.23	49.66	76.98	82.75	78.79	46.00	80.47	50.05	73.16	10.03
		4-5-5	5-5-5	66.57	49.57	76.43	81.62	77.98	45.40	80.09	49.39	72.06	9.47
w/ QLoRA	32	4-16-16	16-16-16	67.47	49.83	77.02	83.24	78.92	46.20	80.58	50.77	73.24	16.85
w/ GSQ-Tuning	32	4-8-8	8-8-8	67.49	49.83	76.98	83.15	78.94	45.60	80.79	51.07	73.56	11.42
		4-7-7	7-7-7	67.38	50.17	77.06	82.81	78.99	45.40	80.79	50.46	73.40	10.87
		4-6-6	6-6-6	67.35	49.83	77.06	83.09	78.89	46.00	80.47	50.31	73.16	10.31
		4-5-5	5-5-5	66.65	48.38	76.18	82.08	78.07	45.60	80.36	49.74	72.77	9.76
w/ QLoRA	64	4-16-16	16-16-16	67.61	49.66	77.23	83.30	78.95	45.40	80.74	51.59	73.24	17.42
w/ GSQ-Tuning	64	4-8-8	8-8-8	67.48	49.57	77.40	82.87	78.88	46.20	80.90	50.72	73.32	11.99
		4-7-7	7-7-7	67.43	49.74	77.27	82.91	78.89	46.00	80.90	50.61	73.09	11.44
		4-6-6	6-6-6	67.35	49.66	77.27	82.75	78.66	79.05	80.90	50.97	73.16	10.89
		4-5-5	5-5-5	66.97	49.91	76.60	81.87	78.15	46.20	80.41	49.54	73.09	10.33
w/ QLoRA	128	4-16-16	16-16-16	67.61	50.34	77.40	83.55	78.89	46.00	80.85	50.92	72.93	18.56
w/ GSQ-Tuning	128	4-8-8	8-8-8	67.62	50.34	77.06	83.18	78.96	46.40	80.69	50.92	73.40	13.14
		4-7-7	7-7-7	67.57	50.43	77.36	83.06	79.05	45.60	80.85	51.28	72.93	12.58
		4-6-6	6-6-6	67.53	50.43	77.31	83.15	78.81	45.80	80.58	50.97	73.16	12.03
		4-5-5	5-5-5	67.10	49.49	76.81	82.08	78.22	46.40	80.03	50.56	73.24	11.48
w/ QLoRA	256	4-16-16	16-16-16	67.91	50.77	77.36	83.64	78.88	46.60	80.74	51.69	73.64	20.85
w/ GSQ-Tuning	256	4-8-8	8-8-8	67.84	51.11	77.06	83.82	78.80	46.40	80.69	52.00	72.85	15.42
		4-7-7	7-7-7	67.74	50.77	77.31	83.79	78.84	46.00	80.63	51.89	72.69	14.87
		4-6-6	6-6-6	67.68	50.77	77.19	83.49	78.82	46.00	80.58	51.38	73.24	14.32
		4-5-5	5-5-5	67.22	50.85	75.84	82.11	78.21	46.00	80.36	50.92	73.48	13.76
w/ QLoRA	512	4-16-16	16-16-16	67.94	50.60	77.48	83.88	79.00	46.40	80.74	52.05	73.40	25.43
w/ GSQ-Tuning	512	4-8-8	8-8-8	67.92	51.02	77.27	83.27	79.04	46.40	81.01	51.79	73.56	20.00
		4-7-7	7-7-7	67.90	51.19	77.15	83.79	78.82	46.80	80.69	51.79	73.01	19.45
		4-6-6	6-6-6	67.82	51.02	77.02	83.85	78.93	46.20	80.90	51.54	73.09	18.89
		4-5-5	5-5-5	67.39	50.94	76.68	82.29	78.39	46.20	80.41	51.69	72.53	18.34

Table 12: 0-shot commonsense QA accuracy (%) across different bits and rank on llama2-70B.

Method	**rank**	LLMs branch	low-rank branch	**Avg.**	**ARC-c**	**ARC-e**	**BoolQ**	**HellaS.**	**OBQA**	**PIQA**	**SCIQ.**	**WinoG.**	**Mem. (G)**
LLaMA2-70B	-	16-16-None	w/o	70.68	56.91	80.05	85.78	83.59	48.60	82.48	48.67	79.40	137.42
w/ QLoRA	16	4-16-16	16-16-16	71.72	58.62	81.44	86.39	83.92	49.80	83.03	50.46	80.11	63.90
w/ GSQ-Tuning	16	4-8-8	8-8-8	71.65	58.62	81.23	86.36	83.87	49.60	83.19	50.41	79.95	49.17
		4-7-7	7-7-7	71.63	58.87	81.57	86.24	83.89	49.20	83.19	50.46	79.64	47.44
		4-6-6	6-6-6	71.58	58.62	81.36	86.15	83.84	49.60	82.97	50.41	79.64	45.72
		4-5-5	5-5-5	71.02	57.34	80.56	85.93	83.75	49.00	82.59	49.33	79.64	43.99
w/ QLoRA	32	4-16-16	16-16-16	71.84	59.13	81.82	86.27	83.88	49.20	83.03	51.02	80.35	64.87
w/ GSQ-Tuning	32	4-8-8	8-8-8	71.78	59.04	81.90	86.33	83.89	49.00	83.19	51.07	79.79	50.17
		4-7-7	7-7-7	71.76	59.30	81.61	86.18	83.98	49.00	83.19	51.02	79.79	48.44
		4-6-6	6-6-6	71.60	58.96	81.36	86.15	83.87	48.80	83.03	51.02	79.64	46.72
		4-5-5	5-5-5	71.26	57.59	80.85	86.15	83.93	49.00	83.13	50.00	79.40	44.99
w/ QLoRA	64	4-16-16	16-16-16	72.22	59.81	82.20	86.51	83.89	50.40	83.13	51.48	80.35	66.82
w/ GSQ-Tuning	64	4-8-8	8-8-8	72.20	59.90	82.32	86.51	83.90	50.20	83.08	51.59	80.11	52.17
		4-7-7	7-7-7	72.18	59.81	82.28	86.39	83.88	50.20	83.13	51.54	80.19	50.44
		4-6-6	6-6-6	72.10	59.39	82.15	86.51	83.94	50.00	83.30	50.92	80.58	48.71
		4-5-5	5-5-5	71.70	58.87	81.48	85.90	83.91	49.60	82.81	50.67	80.43	46.98
w/ QLoRA	128	4-16-16	16-16-16	72.39	60.67	82.37	86.88	84.05	49.20	83.19	52.15	80.66	70.96
w/ GSQ-Tuning	128	4-8-8	8-8-8	72.37	60.75	82.49	87.00	83.94	49.40	83.08	52.15	80.19	56.16
		4-7-7	7-7-7	72.32	60.41	82.45	86.94	83.94	49.00	83.08	52.15	80.58	54.43
		4-6-6	6-6-6	72.28	59.81	82.45	86.91	83.99	49.60	83.35	51.89	80.27	52.70
		4-5-5	5-5-5	71.85	59.47	81.90	86.48	83.82	48.20	83.08	51.02	80.82	50.97

Table 13: 0-shot commonsense QA accuracy (%) across different bits and rank on llama3-3B.

Method	**rank**	**#Bits**	**Avg.**	**ARC-c**	**ARC-e**	**BoolQ**	**HellaS.**	**OBQA**	**PIQA**	**SCIQ.**	**WinoG.**	**Mem. (G)**
LLaMA3-3B	-	16-16-None	64.13	46.25	74.62	77.68	76.01	44.20	79.11	46.11	69.06	6.42
w/ QLoRA	16	4-16-16	65.05	47.53	75.17	78.59	76.09	44.00	79.54	49.44	70.09	6.42
w/ GSQ-Tuning	16	8-8-8	65.10	47.53	74.71	78.35	75.99	45.00	79.65	49.28	70.32	3.57
		7-7-7	64.96	47.18	75.21	78.10	75.98	44.80	79.27	49.95	69.38	3.34
		6-6-6	64.87	46.84	73.78	78.07	75.88	45.80	79.22	49.39	70.01	3.11
		5-5-5	63.97	46.76	72.64	75.78	74.95	45.20	79.05	48.62	68.75	2.88
w/ QLoRA	32	4-16-16	65.24	47.27	75.04	78.87	76.11	44.60	79.76	49.95	70.32	6.54
w/ GSQ-Tuning	32	8-8-8	65.45	48.12	74.71	78.38	76.14	46.00	79.71	49.64	70.96	3.69
		7-7-7	65.43	47.35	74.20	78.99	75.84	46.00	79.92	49.59	71.59	3.46
		6-6-6	65.01	47.44	74.62	78.65	76.03	44.00	79.60	50.05	69.69	3.23
		5-5-5	64.00	44.97	73.32	75.29	74.95	44.60	79.27	48.93	70.24	3.00
w/ QLoRA	64	4-16-16	65.69	47.14	74.75	79.50	76.46	45.50	79.63	50.26	71.32	6.78
w/ GSQ-Tuning	64	8-8-8	65.60	48.12	74.24	79.72	76.00	45.80	79.60	49.69	71.67	3.93
		7-7-7	65.47	47.78	74.71	79.51	76.09	45.80	79.60	49.80	70.48	3.70
		6-6-6	65.39	47.70	74.58	79.24	76.05	44.60	79.60	50.41	70.96	3.47
		5-5-5	64.18	45.14	72.69	75.20	75.27	46.40	79.65	48.62	70.48	3.24
w/ QLoRA	128	4-16-16	65.84	48.24	74.91	79.78	76.27	45.52	79.77	50.48	71.79	6.76
w/ GSQ-Tuning	128	8-8-8	65.79	48.12	74.83	80.28	75.96	45.80	79.54	50.61	71.19	4.41
		7-7-7	65.69	48.04	74.87	79.79	76.08	45.00	79.49	50.61	71.67	4.18
		6-6-6	65.58	47.87	74.54	80.09	76.05	45.40	79.38	50.10	71.27	3.95
		5-5-5	64.46	46.50	72.77	75.99	75.31	46.60	79.00	48.98	70.56	3.72
w/ QLoRA	256	4-16-16	66.12	48.33	75.00	80.94	76.37	45.61	79.97	51.13	71.64	7.61
w/ GSQ-Tuning	256	8-8-8	66.19	48.55	75.13	80.76	76.14	47.00	79.38	50.72	71.82	5.37
		7-7-7	65.96	48.46	75.08	80.43	76.04	45.60	79.76	50.72	71.59	5.13
		6-6-6	65.90	48.38	74.16	79.94	75.81	46.80	79.43	50.87	71.82	4.90
		5-5-5	64.59	46.33	72.60	76.51	75.57	46.40	79.60	49.39	70.32	4.67
w/ QLoRA	512	4-16-16	66.59	49.26	75.20	81.99	76.06	46.74	79.49	51.71	72.27	9.73
w/ GSQ-Tuning	512	8-8-8	66.52	49.49	74.92	81.28	75.89	47.60	79.49	51.59	71.90	7.28
		7-7-7	66.33	48.89	74.75	81.41	76.06	47.00	79.54	51.74	71.27	7.05
		6-6-6	66.31	48.55	75.51	80.80	76.42	46.00	79.60	51.64	71.98	6.82
		5-5-5	64.86	47.44	73.15	76.85	75.62	47.00	79.33	49.18	70.32	6.59

Table 14: 0-shot commonsense QA accuracy (%) across different bits and rank on llama3-8B.

Method	rank	#Bits	Avg.	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	SCIQ.	WinoG.	Mem. (G)
LLaMA3-8B	-	16-16-None	67.18	53.50	77.74	81.13	79.20	45.00	80.63	47.03	73.24	15.01
w/ QLoRA	16	4-16-16	68.14	54.52	79.50	83.43	78.66	44.80	80.85	50.00	73.32	10.71
w/ GSQ-Tuning	16	8-8-8	68.16	54.61	79.84	83.70	78.58	44.80	80.79	49.85	73.16	7.03
		7-7-7	68.00	54.01	79.29	83.46	78.65	45.00	80.85	49.80	73.01	6.65
		6-6-6	67.74	54.01	78.70	83.09	78.49	44.00	80.90	49.44	73.32	6.26
		5-5-5	66.51	51.54	77.27	81.99	77.00	44.40	78.84	48.46	72.61	5.87
w/ QLoRA	32	4-16-16	68.31	55.55	80.39	83.36	78.65	44.60	81.28	50.05	72.61	11.02
w/ GSQ-Tuning	32	8-8-8	68.45	55.72	80.22	83.43	78.60	45.00	81.18	50.20	73.32	7.23
		7-7-7	68.29	54.95	80.13	83.36	78.53	44.80	81.01	50.20	73.32	6.84
		6-6-6	68.08	55.29	79.29	83.55	78.28	45.80	81.07	49.39	71.98	6.46
		5-5-5	66.48	51.71	77.69	82.11	76.91	44.20	79.43	48.16	71.67	6.07
w/ QLoRA	64	4-16-16	68.45	55.63	80.13	83.67	78.78	44.80	81.28	50.41	72.93	11.64
w/ GSQ-Tuning	64	8-8-8	68.61	55.97	80.22	83.61	78.68	45.20	81.50	50.41	73.32	7.63
		7-7-7	68.57	55.97	80.68	83.73	78.84	45.20	81.01	50.26	72.85	7.24
		6-6-6	68.22	55.55	79.29	83.67	78.47	44.80	80.90	50.05	73.09	6.86
		5-5-5	66.69	54.10	77.99	81.65	77.12	43.80	79.54	47.90	71.43	6.47
w/ QLoRA	128	4-16-16	68.77	56.14	80.56	83.98	79.03	45.60	81.34	50.56	72.93	12.13
w/ GSQ-Tuning	128	8-8-8	68.72	56.57	80.22	83.82	78.80	45.40	81.23	50.41	73.32	8.43
		7-7-7	68.71	56.48	80.18	83.88	78.78	45.80	81.34	50.36	72.93	8.04
		6-6-6	68.67	56.91	79.50	83.79	78.71	46.60	80.52	50.36	73.01	7.66
		5-5-5	66.92	52.47	78.45	82.63	77.22	44.60	79.49	48.52	71.98	7.27
w/ QLoRA	256	4-16-16	69.09	56.74	80.35	84.56	79.02	45.20	81.83	50.92	74.11	13.81
w/ GSQ-Tuning	256	8-8-8	69.04	56.57	80.85	84.07	78.97	45.40	81.45	51.28	73.72	10.03
		7-7-7	69.00	56.83	80.89	84.25	78.96	45.60	81.50	50.46	73.56	9.64
		6-6-6	68.84	56.74	79.80	83.98	78.84	46.40	81.12	50.77	73.09	9.26
		5-5-5	67.54	53.33	78.49	83.21	77.38	44.60	79.98	48.93	73.64	8.87
w/ QLoRA	512	4-16-16	69.18	57.17	80.30	84.65	79.28	46.40	81.07	50.36	74.27	16.81
w/ GSQ-Tuning	512	8-8-8	69.24	56.48	80.47	85.35	79.13	45.40	81.56	51.54	74.03	13.23
		7-7-7	69.16	56.40	80.68	85.26	79.10	45.80	81.28	51.13	73.64	12.84
		6-6-6	69.01	56.57	80.01	84.56	78.84	45.80	81.23	51.64	73.48	12.45
		5-5-5	67.90	54.38	78.60	83.97	78.08	45.30	80.24	50.00	72.76	12.07

Table 15: 0-shot accuracy comparison with FP8 in different quantization bits in 64 rank setting.

Method	#Bits	Avg.	ARC-c	ARC-e	BoolQ	HellaS.	OBQA	PIQA	SCIQ.	WinoG.	Mem. (G)
LLaMA2-7B	16-16-None	64.13	46.25	74.62	77.68	76.01	44.20	79.11	46.11	69.06	13.20
w/ QLoRA	4-16-16	65.69	47.14	74.75	79.50	76.46	45.50	79.63	50.26	71.32	9.73
w/ FP8	8-8-8	64.46	46.84	73.61	77.83	76.03	44.60	79.65	47.80	69.38	6.88
w/ GSQ-Tuning	8-8-8	65.60	48.12	74.24	79.72	76.00	45.80	79.60	49.69	71.67	6.88
	6-6-6	65.39	47.70	74.58	79.24	76.05	44.60	79.60	50.41	70.96	6.17
	5-5-5	64.18	45.14	72.69	75.20	75.27	46.40	79.65	48.62	70.48	5.81
LLaMA3-8B	16-16-None	67.18	53.50	77.74	81.13	79.20	45.00	80.63	47.03	73.24	15.01
w/ QLoRA	4-16-16	68.45	55.63	80.13	83.67	78.78	44.80	81.28	50.41	72.93	11.71
w/ FP8	8-8-8	66.46	50.77	76.39	81.38	78.19	43.40	79.92	47.29	74.35	7.63
w/ GSQ-Tuning	8-8-8	68.61	55.97	80.22	83.61	78.68	45.20	81.50	50.41	73.32	7.63
	6-6-6	68.22	55.55	79.29	83.67	78.47	44.80	80.90	50.05	73.09	6.86
	5-5-5	66.69	54.10	77.99	81.65	77.12	43.80	79.54	47.90	71.43	6.47