Accurate and Efficient Fine-Tuning of Quantized Large Language Models Through Optimal Balance in Adaptation

Ao Shen, Zhiquan Lai*, Qiang Wang, Xionglve Li, Lizhi Zhang, Dongsheng Li, Jiaxin Li
National Key Laboratory of Parallel and Distributed Computing, College of Computer Science and Technology, National University of Defense Technology, Changsha, China
{shenao, zqlai}@nudt.edu.cn

Abstract

Large Language Models (LLMs) have demonstrated impressive performance across various domains. However, the enormous number of model parameters makes fine-tuning challenging, significantly limiting their application and deployment. Existing solutions combine parameter quantization with Low-Rank Adaptation (LoRA), reducing memory usage but causing performance degradation. Additionally, converting fine-tuned models to low-precision representations further degrades performance. In this paper, we identify an imbalance in fine-tuning quantized LLMs with LoRA: overly complex adapter inputs and outputs versus low effective trainability of the adapter, leading to underfitting during fine-tuning. Thus, we propose Quantized LLMs fine-tuning with Balanced Low-Rank Adaptation (Q-BLoRA), which simplifies the adapter's inputs and outputs while increasing the adapter's rank to alleviate underfitting during fine-tuning. For low-precision deployment, we propose Quantization-Aware fine-tuning with Balanced Low-Rank Adaptation (QA-BLoRA), which aligns with the block-wise quantization and facilitates quantization-aware fine-tuning of low-rank adaptation based on the parameter merging of Q-BLoRA. Both Q-BLoRA and QA-BLoRA are easily implemented and offer the following optimizations: (i) Q-BLoRA consistently achieves state-of-the-art accuracy compared to baselines and other variants; (ii) QA-BLoRA enables the direct generation of low-precision inference models, which exhibit significant performance improvements over other low-precision models. We validate the effectiveness of Q-BLoRA and QA-BLoRA across various models and scenarios. Code has been made available at `https://github.com/xiaocaigou/qbaraqahira`.

*Corresponding author.

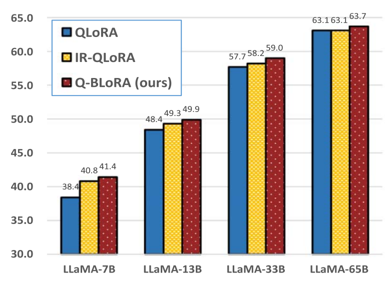
1 Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Workshop et al., 2022; Touvron et al., 2023a; Bubeck et al., 2023) have demonstrated remarkable performance across a wide spectrum of Natural Language Processing (NLP) tasks (Wei et al., 2022a), establishing new benchmarks in various domains. These models can be fine-tuned for specific applications, significantly enhancing their versatility and adaptability (Brown et al., 2020; Devlin et al., 2018; Zhao et al., 2023). However, the fine-tuning process requires substantial memory resources, limiting the accessibility of State-Of-The-Art (SOTA) NLP technologies.

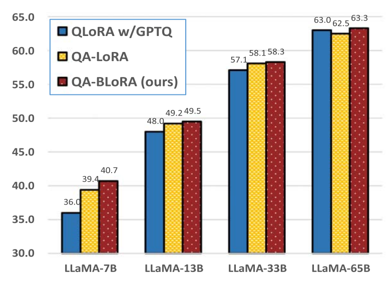
To address this challenge, existing solutions combine parameter quantization with Low-Rank Adaptation (LoRA) fine-tuning (Dettmers et al., 2024; Xu et al., 2023b; Qin et al., 2024), leveraging the strengths of both techniques to significantly reduce memory overhead. Initially, Post-Training Quantization (PTQ) (Dettmers et al., 2023; Frantar et al., 2023; Lin et al., 2023; Xiao et al., 2023; Wei et al., 2022b; Wu et al., 2023) is applied to convert the pre-trained model into a low-precision representation, thereby reducing the memory required for model loading. Subsequently, LoRA (Hu et al., 2021) is applied for Parameter-Efficient Fine-Tuning (PEFT), introducing a small number of new learnable parameters while keeping most pre-trained parameters fixed, thereby reducing the memory needed for parameter updates.

However, despite significantly reducing the memory usage, the combination of parameter quantization and LoRA fine-tuning leads to performance degradation. For example, in the MMLU test, fine-tuning LLaMA-13B with such approaches leads to an accuracy drop of up to 1.3% compared to full fine-tuning (Dettmers et al., 2024). While recent research attempts to recover lost accuracy through information retention techniques (Qin et al., 2024), these approaches often

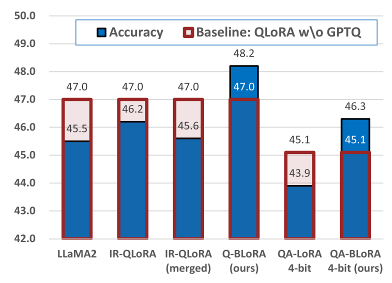
Transactions of the Association for Computational Linguistics, vol. 13, pp. 861–877, 2025. https://doi.org/10.1162/tacl.a.23
Action Editor: Doug Downey. Submission batch: 7/2024; Revision batch: 2/2025; Published 8/2025.



(a) The comparison of Q-BLoRA in the scenario of a 4-bit pre-trained model with a 16-bit adaptation scenario.



(b) The comparison of QA-BLoRA in the scenario of a 4-bit inference model scenario.



(c) The comparison on LLaMA2 model, where only our Q-BLoRA and QA-BLoRA outperformed their respective baselines.

Figure 1: The comparison of 5-shot MMLU accuracy (%) based on the LLaMA and LLaMA2 family. All models are fine-tuned on the Alpaca dataset. Full results are provided in Table 1 and Table 2.

compromise deployment efficiency. Moreover, in low-precision deployment scenarios, compressing the fine-tuned model into a low-precision representation (Frantar et al., 2023) or directly employing Quantization-Aware Training (QAT) (Xu et al., 2023b) requires additional PTQ operations and often struggles to achieve satisfactory performance.

In this paper, we identify an imbalance in fine-tuning quantized LLMs with LoRA: overly complex adapter inputs and outputs versus low effective trainability of the adapter, leading to underfitting during fine-tuning. To address this issue, we reduce the dimensions of the input and output to simplify the adapter's complexity, while increasing the rank of the low-rank matrices to enhance the adapter's representational capacity. The compression and restoration of the inputs and outputs are accomplished through non-parameterized operations with negligible computational overhead. Consequently, we propose Quantized LLMs fine-tuning with Balanced Low-Rank Adaptation (Q-BLoRA), which alleviates the underfitting during fine-tuning and achieves lossless merging of adapter parameters with the pre-trained model.

For low-precision deployment scenarios, we propose Quantization-Aware fine-tuning with Balanced Low-Rank Adaptation (QA-BLoRA), which builds upon the adapter parameter merging of Q-BLoRA. By aligning with the block-wise quantization of the pre-trained model, QA-BLoRA facilitates quantization-aware fine-tuning of low-rank adaptation, enabling the direct generation of low-precision inference models.

Both Q-BLoRA and QA-BLoRA are easy to implement, achieving excellent performance without compromising the original efficiency advantages. We evaluated our methods on the LLaMA and LLaMA2 model families (Touvron et al., 2023a,b), as well as the Mistral (Jiang et al., 2023) and Gemma (Team et al., 2024) models. Figure 1a demonstrates that Q-BLoRA consistently achieves SOTA accuracy compared to other fine-tuning methods using 4-bit pre-trained models with 16-bit adapters. Figure 1b highlights that QA-BLoRA outperforms other 4-bit inference models, even surpassing many benchmarks with 16-bit adapters. On LLaMA2, as shown in Figure 1c, only Q-BLoRA and QA-BLoRA surpass the baseline, showcasing their generalizability across diverse model architectures. Moreover, Q-BLoRA and QA-BLoRA do not require additional PTQ operations or extra trainable parameters, preserving the efficiency of both the fine-tuning process and the inference model. Comprehensive evaluations on the MMLU benchmark, question-answering tasks, and evaluations using GPT-4 / ChatGPT confirm the consistent effectiveness of our Q-BLoRA and QA-BLoRA.

2 Related Work

Quantization of LLMs. Quantization is a crucial strategy for compressing LLMs, enhancing efficiency and scalability by reducing the bit-width of parameters and activations. Given the high training costs of LLMs, the literature focuses mainly on PTQ, which converts the pre-trained model to lower bit-width representations (Dettmers et al., 2022; Frantar et al., 2023; Yao et al., 2022; Wu et al., 2023). Handling outliers in computation poses a significant challenge, as outliers are critical yet introduce substantial quantization errors. Several approaches address outliers by treating them separately (Dettmers et al., 2022; Xiao et al., 2023; Wei et al., 2022b) or by employing calibration data to minimize post-quantization errors (Frantar et al., 2023; Lin et al., 2023). However, the performance of quantized models is still affected, and additional computational overhead is often required.

LoRA. PEFT introduces a small set of learnable parameters without updating the majority of the pre-trained parameters, thereby reducing the memory required for updates. Among these, LoRA (Hu et al., 2021; Valipour et al., 2022), as a prevalent PEFT approach, introduces two low-rank matrices and uses their product as an adapter. This method significantly reduces the number of trainable parameters for downstream tasks. Furthermore, after fine-tuning, the adapter parameters can be seamlessly integrated with the pre-trained model, thereby eliminating the need for additional computational overhead of the adapter. Recent studies have further advanced LoRA (Hayou et al., 2024; Jiang et al., 2024; Zi et al., 2023; Zhang et al., 2023b; Biderman et al., 2024; Tian et al., 2024) and attempted to combine it with model compression techniques (Zhang et al., 2023a; Dettmers et al., 2024; Mao et al., 2024).

Combination of LoRA and Quantization. Quantization can compress pre-trained models into low-precision representations, while LoRA reduces the memory required for fine-tuning. Therefore, integrating both approaches allows for the full utilization of their respective advantages. Typically, QLoRA (Dettmers et al., 2024) proposes an efficient fine-tuning method that utilizes 4-bit NormalFloat representations for pre-trained parameters, followed by LoRA-based fine-tuning, significantly reducing memory usage but yielding a suboptimal model. QA-LoRA (Xu et al., 2023b) compresses the pre-trained model using GPTQ (Frantar et al., 2023) and then fine-tunes it using QAT, allowing the fine-tuned adapter parameters to merge into the low-bit-width representation of the pre-trained model. However, GPTQ requires additional computation, and the QAT method of QA-LoRA introduces significant information loss, leading to degraded model performance. IR-QLoRA (Qin et al., 2024) enhances model performance through Information Calibration Quantization (ICQ) for pre-trained model and Information Elastic Connection (IEC) during fine-tuning. Nevertheless, ICQ demands extra computation, and the adapters using IEC are no longer computationally equivalent to the base layer, preventing lossless merging with pre-trained parameters.

3 The Proposed Approach

3.1 Preliminaries

Baseline: Low-Bit Quantization with Low-Rank Adaptation. We begin our notation system with LoRA (Hu et al., 2021), which utilizes the product of two matrices to approximate parameter updates during fine-tuning, thus allowing efficient fine-tuning. Let $\mathbf{W}$ denote the pre-trained parameter matrix of size $D_{in} \times D_{out}$, and x represent the input feature vector of length D_{in}. The output feature y of length D_{out} is computed as $y = \mathbf{W}^\top x$. LoRA introduces two matrices, $\mathbf{A}$ and $\mathbf{B}$, with dimensions $D_{in} \times D_{\text{rank}}$ and $D_{\text{rank}} \times D_{out}$, respectively, where $D_{\text{rank}} \ll min(D_{in}, D_{out})$. This allows the product $\mathbf{AB}$ to form a low-rank matrix of the same size as $\mathbf{W}$ using only a small number of parameters. During fine-tuning, the computation is adjusted to $y = \mathbf{W}^\top x + s \cdot \triangle\mathbf{W}^\top x = \mathbf{W}^\top x + s \cdot (\mathbf{AB})^\top x$, where s is the non-learned coefficient for weight tuning. The large parameter matrix $\mathbf{W}$ remains frozen, while only $\mathbf{A}$ and $\mathbf{B}$ are updated, significantly reducing the number of trainable parameters. After fine-tuning, the adapter parameters can be merged with the pre-trained parameters for inference as $\mathbf{W}' = \mathbf{W} + s \cdot \triangle\mathbf{W} = \mathbf{W} + s \cdot \mathbf{AB}$.

Quantization can build upon LoRA by using low bit-width representations for pre-trained parameters, further reducing the memory required for fine-tuning (Dettmers et al., 2024). A widely adopted quantization method is block-wise quantization (Frantar et al., 2023; Dettmers et al., 2024), which divides the parameter matrix $\mathbf{W}$ into smaller blocks and quantizes each block independently. For each block, the quantization can be expressed as:

$$\tilde{w} = \alpha \cdot \hat{w} + \beta = \alpha \cdot \left\lceil \frac{w - \beta}{\alpha} \right\rfloor + \beta, \quad (1)$$

where $\hat{w}$ is the N-bit representations of w, and $\lceil\cdot\rfloor$ denotes the rounding operation, which maps the value to the low bit-width representation space. The parameters α and β are a pair of quantization factors along with the block parameters for dequantization. When applying min-max quantization, $\alpha = (w_{max} - w_{min})/(2^N - 1)$ and $\beta = w_{min}$. For absmax quantization, which primarily targets symmetric distributions around zero, $\alpha = \mathrm{absmax}(\mathrm{w})/(2^N - 1)$ and $\beta = 0$. Combined LoRA and quantization, the calculations of fine-tuning can be expressed as:

$$y = \tilde{\mathbf{W}}^\top x + s \cdot (\triangle \mathbf{W})^\top x = \tilde{\mathbf{W}}^\top x + s \cdot (\mathbf{AB})^\top x,$$

where $\tilde{\mathbf{W}}$ is the quantized parameters of the pre-trained model. By this approach, fine-tuning of LLMs can be accomplished with a minimal number of GPUs. However, this fine-tuning approach results in a decrease in accuracy. Additionally, the adapter components $s \cdot \mathbf{AB}$ and pre-trained models $\tilde{\mathbf{W}}$ must utilize high precision (BF16 or FP16) to achieve effective integration. And compressing the fine-tuned model to lower precision further degrades its performance.

Our Objective: Efficient yet Effective Adaptation and Deployment. Our goal is to achieve the following while maintaining the original efficiency: (i) Obtain high-performance fine-tuned models through LoRA-based fine-tuning of quantized LLMs. (ii) Seamlessly integrate the adapter parameters into the pre-trained model after fine-tuning, without incurring additional computational overhead during deployment. Recent work attempts to achieve these two goals, but fail to achieve both, even with additional computation. Specifically, QLoRA (Dettmers et al., 2024) inherits the deployment advantages of LoRA by enabling lossless adapter integration. However, it suffers from significant performance degradation. IR-QLoRA (Qin et al., 2024) aims to achieve better performance through fine-tuning. However, the enhancement of accuracy relies on additional computations and trainable parameters. Moreover, the shortcut-like structure modifies the computational graph, preventing the adapter from effectively integrating with the pre-trained model. QA-LoRA (Xu et al., 2023b) proposes a QAT fine-tuning method that allows the side weights $s \cdot \mathbf{AB}$ to merge into the quantized pre-trained model $\tilde{\mathbf{W}}$. However, the quantization of pre-trained model (Frantar et al., 2023) require the computation for calibration, and the accuracy of the fine-tuned model remains far from the limits regarding accuracy.

Therefore, we propose fine-tuning methods for quantized LLMs that are tailored to both high-precision floating-point deployment scenarios (e.g., GPUs) and low-precision deployment scenarios (e.g., edge devices). Our approach aims to simultaneously achieve high model performance and efficient deployment, while maintaining the high memory efficiency of the fine-tuning process.

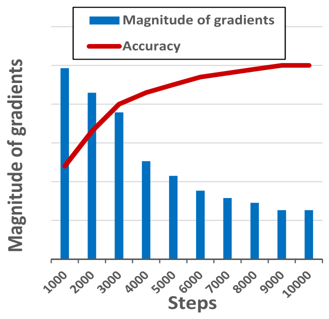
3.2 Q-BLoRA

The Imbalance in Fine-Tuning Quantized LLMs with LoRA. We identify an imbalance in fine-tuning quantized LLMs with LoRA: overly complex adapter inputs and outputs versus low effective trainability of the adapter, leading to underfitting during fine-tuning.

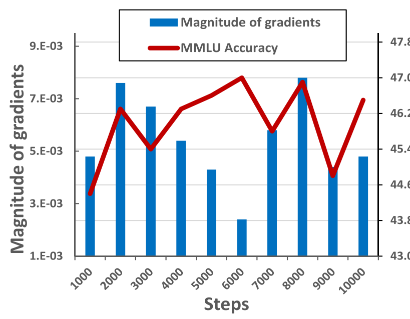
Typically, neural networks require sufficient or even excessive representational capacity to progressively fit and learn knowledge through continuous training. When the model's representational capacity is adequate, the training loss steadily decreases, parameters gradually converge, gradients diminish, and accuracy improves, as shown in Figure 2a. However, when quantized LLMs are fine-tuned using LoRA, as shown in Figure 2b, significant gradients persist even in the later stages of training, and accuracy exhibits noticeable fluctuations. These observations suggest that fine-tuning quantized LLMs with LoRA may result in underfitting.[1]

As shown in Figure 2c, compared to full fine-tuning, the loss when fine-tuning quantized LLMs with LoRA (i.e., QLoRA) remains consistently high and fluctuates, indicating that the model struggles to fit. In contrast, when fine-tuning a non-quantized LLM using LoRA (i.e., LoRA), the loss, despite some fluctuations, decreases to a much lower level. This discrepancy arises because, in LoRA, the pre-trained model has already acquired sufficient knowledge, requiring only minimal adaptation during fine-tuning to fit task-specific knowledge. However, in QLoRA, the quantization process introduces information loss, forcing the adapter to not only learn task-specific knowledge but also compensate for the information lost during quantization. This dual burden significantly increases the difficulty of the fine-tuning process, resulting in underfitting.

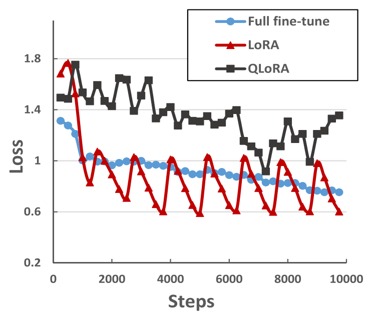
[1]In the case of the Q, K, and V layers in LLaMA2-7B, the rank of the adapter matrices is typically only 16 to 64 (Dettmers et al., 2024), while the input and output vectors have a dimensionality as high as 4096. This contrasts sharply with conventional training/fine-tuning methods, where, during full fine-tuning of LLaMA2-7B, the number of trainable parameters in the Q, K, and V layers is 4096 times that of their input and output dimensions.



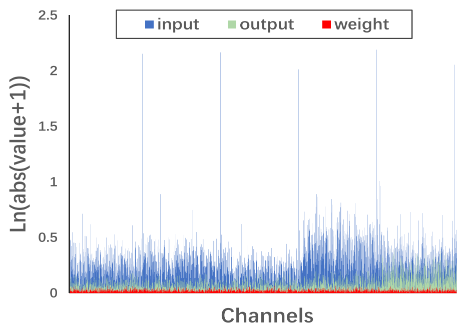
(a) Illustration of gradient and accuracy trends under ideal conditions.



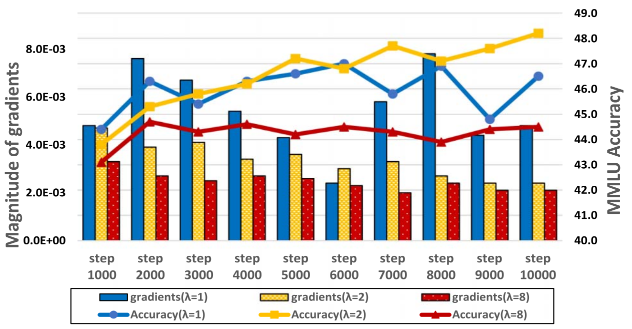
(b) Gradient and accuracy changes during LoRA fine-tuning of quantized LLaMA2-7B.



(c) Loss changes of different methods during fine-tuning of LLaMA2-7B.



(d) Magnitude of the input, output, and weights of a attn.q_proj layer in Llama2-7B.



(e) Gradient and accuracy changes during LLaMA2 fine-tuning with different balancing factors.

Figure 2: Analysis of fine-tuning with low-rank adaptation in quantization scenario.

The above findings indicate that fine-tuning quantized LLMs with LoRA leads to underfitting, which adversely affects the model's performance.

The Rise of Q-BLoRA. Recent studies have proposed potential solutions to address this imbalance problem. Specifically, there is a certain degree of redundancy in the input and output of the layers in LLM models (Xiao et al., 2023; Dettmers et al., 2022; Wu et al., 2023), and not all features from every channel are equally important for the model's performance. Additionally, due to the significant numerical differences across channels (as shown in Figure 2d), in cases of underfitting, excessive parameter updates may even hinder model convergence, leading to performance degradation.

We use λ as the balancing factor, which serves as the compression factor for both the input and output dimensions and as the multiplier for rank expansion. We record the accuracy and gradient values during fine-tuning for various λ. As shown in Figure 2e, when $\lambda = 1$, the input, output and rank of the adapter remain unmodified. In this case, the gradients during fine-tuning remain consistently large, and the accuracy exhibits significant fluctuations. When $\lambda = 2$, the gradients are noticeably reduced and show a decreasing trend, while the accuracy improves. This suggests that the fine-tuning process achieves a better balance, effectively mitigating the underfitting issue. Finally, when $\lambda = 8$, the gradients decrease further, indicating a substantial reduction in the difficulty of fitting. However, at this stage, excessive compression of the input and output dimensions leads to a loss of valuable information, resulting in performance degradation.

These observations indicate that simplifying the adapter's input and output dimensions while expanding its rank can optimize the balance during fine-tuning, alleviating the underfitting problem.

Based on this analysis, we propose Quantized LLMs fine-tuning with Balanced Low-Rank Adaptation (Q-BLoRA), as illustrated in Figure 3. Considering the parameter distribution of LLMs, Q-BLoRA employs NF4 to compress the pre-trained model (Dettmers and Zettlemoyer, 2023). On the adapter side, Q-BLoRA first applies the $\mathsf{AvgPool}(\lambda)$ operation to the input x, transforming it into x' with dimensions D_{in}/λ. The dimensions of the matrices $\mathbf{A}$ and $\mathbf{B}$ in LoRA are subsequently adjusted to $(\frac{D_{in}}{\lambda}, \lambda \cdot D_{\text{rank}})$ and $(\lambda \cdot D_{\text{rank}}, \frac{D_{out}}{\lambda})$, respectively. The output

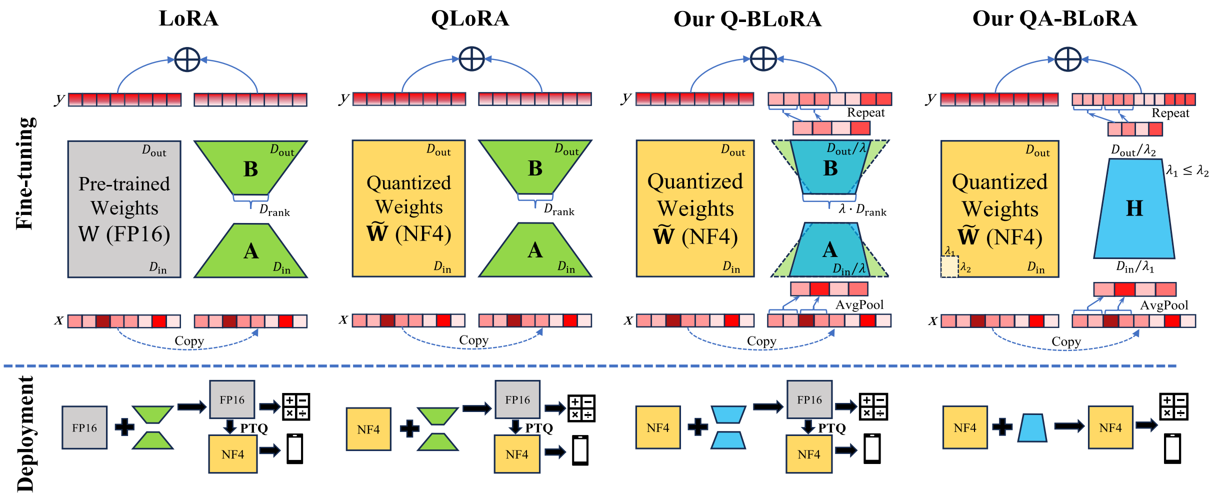


Figure 3: An illustration of our Q-BLoRA and QA-BLoRA. Compared to LoRA and QLoRA, our Q-BLoRA achieves a better fine-tuning balance by increasing the rank of adaptation and using non-parameter operators to change the dimension of input and output. Our QA-BLoRA employs a single matrix for adaptation, enabling the fine-tuning of an efficient model ready for inference.

$$
\begin{aligned}
(\triangle \mathbf{W}^{in}_{BLoRA})^\top x^{pool} &= \begin{pmatrix} w^{in}_{11} & w^{in}_{21} & \cdots & w^{in}_{\frac{D_{in}}{\lambda_1},1} \\ w^{in}_{12} & w^{in}_{22} & \cdots & w^{in}_{\frac{D_{in}}{\lambda_1},2} \\ \vdots & \vdots & \ddots & \vdots \\ w^{in}_{1,D_{out}} & w^{in}_{2,D_{out}} & \cdots & w^{in}_{\frac{D_{in}}{\lambda_1},D_{out}} \end{pmatrix} \begin{pmatrix} \frac{\sum_{i=1}^{\lambda_1} x_i}{\lambda_1} \\ \frac{\sum_{i=\lambda_1+1}^{2\lambda_1} x_i}{\lambda_1} \\ \vdots \\ \frac{\sum_{i=D_{in}-\lambda_1+1}^{D_{in}} x_i}{\lambda_1} \end{pmatrix} \\
&= \frac{1}{\lambda_1} \begin{pmatrix} w^{in}_{11} \sum_{i=1}^{\lambda_1} x_i + w^{in}_{21} \sum_{i=\lambda_1+1}^{2\lambda_1} x_i + \cdots + w^{in}_{\frac{D_{in}}{\lambda_1},1} \sum_{i=D_{in}-\lambda_1+1}^{D_{in}} x_i \\ \vdots \\ w^{in}_{1,D_{out}} \sum_{i=1}^{\lambda_1} x_i + w^{in}_{2,D_{out}} \sum_{i=\lambda_1+1}^{2\lambda_1} x_i + \cdots + w^{in}_{\frac{D_{in}}{\lambda_1},D_{out}} \sum_{i=D_{in}-\lambda_1+1}^{D_{in}} x_i \end{pmatrix}
\end{aligned}
\tag{2}
$$

y' with dimension D_{out}/λ is computed as $y' = (\mathbf{AB})^\top x'$. Next, the output y' undergoes the repeat_interleave(λ) operation, repeating elements and producing the final adapter output y, with dimensions restored to D_{out}.

In this process, Q-BLoRA reduces the input and output dimensions of the adapter by a factor of λ, which not only reduces the complexity of fine-tuning but also alleviates fluctuations in the adapter's input and output, mitigating the negative effects of excessive parameter updates. Simultaneously, it expands the rank of the adapter to enhance its effective representational capacity. By achieving an optimal balance between input/output dimensions and rank, Q-BLoRA mitigates underfitting without increasing the number of trainable parameters. For a detailed analysis of the choice of λ, refer to Section 4.3.

Merging Q-BLoRA Into a Pre-Trained Model. A notable advantage of LoRA (Hu et al., 2021) and QLoRA (Dettmers et al., 2024) is that the low-rank adaptation $\triangle \mathbf{W}$ has the same dimensions as the pre-trained model $\mathbf{W}$ or $\tilde{\mathbf{W}}$. This allows for merging into the pre-trained model, making inference no longer require additional computation from the adapter. However, the adaptation weight of our Q-BLoRA $\triangle \mathbf{W}_{BLoRA}$ is in shape $(D_{in}/\lambda, D_{out}/\lambda)$, necessitating the design of methods to integrate Q-BLoRA effectively.

Let's start by analyzing the calculation when Q-BLoRA is *just compressing the input*, and denote the balancing factor at this stage as λ_1. The AvgPool(λ_1) operation changes $x = (x_1, x_2, \ldots, x_{D_{in}})^\top$ of length D_{in} to $x^{pool} = (\frac{\sum_{i=1}^{\lambda_1} x_i}{\lambda_1}, \frac{\sum_{i=\lambda_1+1}^{2\lambda_1} x_i}{\lambda_1}, \ldots, \frac{\sum_{i=D_{in}-\lambda_1+1}^{D_{in}} x_i}{\lambda_1})^\top$ of length D_{in}/λ_1, and the adaptation weight $\triangle \mathbf{W}^{in}_{BLoRA}$ is in shape $(D_{in}/\lambda_1, D_{out})$. The computation can be described as Equation (2).

Equation (2) can be equivalent to the baseline where the multiplication is between $\triangle \mathbf{W}$ in shape

Algorithm 1 Q-BLoRA Pseudocode in the PyTorch-like style.

```
# lambda: the balancing factor; W_hat: the quantized weight; scaling: the scaling factor for adaptation
# D_in, D_out: the input and output dimensions of the pretrained model; D_rank: the dimension of low-rank
    adaptation of QLoRA
qblora_pool = nn.AvgPool1d(lambda)
qblora_A = nn.Linear(lambda * D_rank, D_in / lambda)
qblora_B = nn.Linear(D_out / lambda, lambda * D_rank)
def qblora_forward(x, W_hat, alpha, beta, qblora_A, qblora_B):
    W_tilde = dequantize(W_hat, alpha, beta)
    result = x @ W_tilde
    result += qblora_pool(x) @ qblora_A.transpose(0, 1) @ qblora_B.transpose(0, 1)
    result = scaling * torch.repeat_interleave(result, lambda, dim = 2)
    return result
def get_delta_weight(qblora_A, qblora_B):
    delta_weight = torch.repeat_interleave(qblora_A @ qblora_B, lambda, dim=0), lambda, dim=1) / lambda
    delta_weight = torch.repeat_interleave(delta_weight, lambda, dim=1)
    return delta_weight
```

(D_{in}, D_{out}) and x of length D_{in}, under the condition that $w^{in}_{m,n} = \lambda_1 w_{\lambda_1 m+1,n} = \lambda_1 w_{\lambda_1 m+2,n} = \cdots = \lambda_1 w_{\lambda_1 m+\lambda_1,n}$ for any $m \in [0, D_{in}/\lambda_1) \cap \mathbb{Z}$ and $n \in [0, D_{out}] \cap \mathbb{Z}$ is satisfied.

When *only the Q-BLoRA output is considered*, we get the output $y^{re} = (y^{re}_1, y^{re}_2, \cdots, y^{re}_{D_{out}/\lambda_2})$ of length D_{out}/λ_2. Then, the repeat_interleave operation expands y^{re} to $y = (y^{re}_1, y^{re}_1, \cdots, y^{re}_1, y^{re}_2, y^{re}_2, \cdots, y^{re}_2, \cdots, y^{re}_{D_{out}/\lambda_2})$ of length D_{out}. Similarly, it can be equivalent to the baseline when $w^{out}_{m,n} = w_{m,\lambda_2 n+1} = w_{m,\lambda_2 n+2} = \cdots = w_{m,\lambda_2 n+\lambda_2}$ for any $m \in [0, D_{in}) \cap \mathbb{Z}$ and $n \in [0, D_{out}/\lambda_2) \cap \mathbb{Z}$.

Together, we can get the $\triangle\mathbf{W}$ in shape (D_{in}, D_{out}), which is equivalent to the fine-tuning using Q-BLoRA. This equivalent $\triangle\mathbf{W}$ can be divided into $\frac{D_{in}D_{out}}{\lambda_1\lambda_2}$ blocks of size (λ_1, λ_2), each with the same value of elements. Our Q-BLoRA can be implemented by inserting/modifying a few lines of code, as described in Algorithm 1.

3.3 QA-BLoRA

In some application scenarios, low-precision models are required for inference without relying on high-precision formats such as FP16 or BF16. To address this, HuggingFace[2] recommends PTQ after fine-tuning. While this approach is feasible, PTQ inevitably leads to some accuracy degradation. In contrast, QAT adapts to low precision during the training process, thereby improving performance. Therefore, the optimal solution is to directly obtain a low-precision inference model through quantization-aware fine-tuning.

Since the quantized pre-trained model is represented using a triplet $(\hat{w}, \alpha, \beta)$ in a block-wise fashion, as shown in Equation (1), where $\hat{w}$ is a discrete series corresponding to the rounding operation $\lceil\cdot\rfloor$, and α and β are high-precision floating-point numbers, it considered continuous. For the adapter weight $\triangle\mathbf{W}$ to merge into the quantized pre-trained model, the values of $(\triangle w - \beta)/\alpha$ within each block must align with the discrete series defined by the rounding operation. Meeting this condition is challenging, especially since each block corresponds to different parameters α and β.

Thus, we need to relax the constraints. A feasible solution is to make all $\triangle w$ within each block equal, allowing $\triangle w$ to be viewed as an adjustment to β, thereby enabling quantization-aware fine-tuning of LoRA. As analyzed in Section 3.2, this can be easily achieved through the parameter merging of Q-BLoRA, wherein every block of size (λ_1, λ_2) within the $\triangle\mathbf{W}$ of Q-BLoRA has equal elements. We only need to adjust λ_1 and λ_2 so that it aligns to the size of the quantization block. The detailed process is shown in Figure 4.

Based on this analysis, we propose Quantization-Aware fine-tuning with Balanced Low-Rank Adaptation (QA-BLoRA). This approach compresses the adapter's inputs and outputs to align with block-wise quantization, thereby achieving quantization-aware fine-tuning for LoRA. Typically, pre-trained models use block quantization with sizes of 4×8 or 8×8 (Frantar et al., 2023;

[2]`https://huggingface.co/blog/overview-quantization-transformers`.

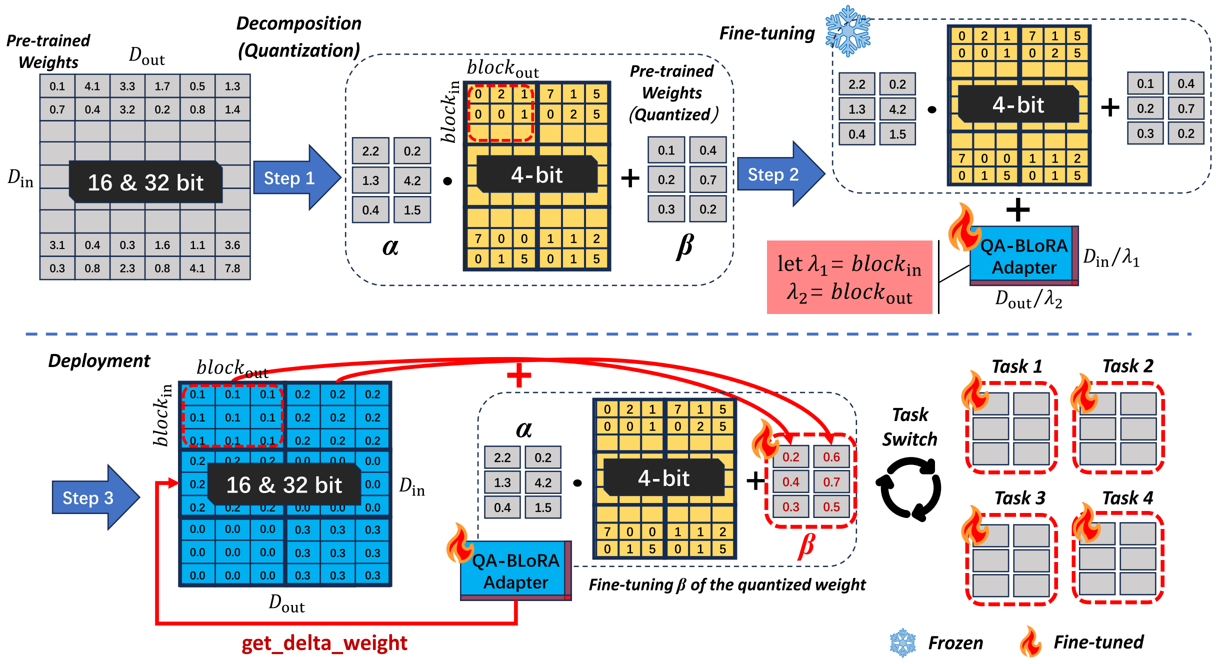


Figure 4: Quantization-aware fine-tuning of LoRA for low-precision deployment and task switching.

Algorithm 2 QA-BLoRA Pseudocode in the PyTorch-like style.

```
# lambda_1, lambda_2: the compressing factor of input & output; W_hat: the quantized weight
# D_in, D_out: the input & output dimensions of the pretrained model; scaling: the scaling factor for adaptation
qablora_pool = nn.AvgPool1d(lambda_1)
qablora_H = nn.Linear(D_out / lambda_2, D_in / lambda_1)
def qablora_forward(x, W_hat, alpha, beta, qablora_H):
    W_tilde = dequantize(W_hat, alpha, beta)
    result = x @ W_tilde
    result += qablora_pool(x) @ qablora_H.transpose(0, 1)
    result = scaling * torch.repeat_interleave(result, lambda_2, dim = 2)
    return result
def get_new_beta(qablora_H, beta):
    new_beta = torch.repeat_interleave(qablora_A, lambda_1, dim=0) / lambda_1
    new_beta = scaling * torch.repeat_interleave(delta_weight, lambda_2, dim=1) + beta
    return new_beta
```

Dettmers and Zettlemoyer, 2023). When both the input and output are significantly simplified, we employ a single matrix as the adapter to enhance the fine-tuning capability (Jiang et al., 2024). Furthermore, as shown in Figure 2d, the input contains more information (including more outliers) than the output. Consequently, we set $\lambda_1 \leq \lambda_2$, prioritizing the preservation of information in the input. For an explanation of QA-BLoRA, see Figure 3. For further analysis of λ_1 and λ_2, see Section 4.3. Our QA-BLoRA can also be implemented by inserting/modifying a few lines of code, as described in Algorithm 2.

4 Experiments

4.1 Settings

We establish Q-BLoRA and QA-BLoRA on the LLaMA and LLaMA2 model families (Touvron et al., 2023a,b), the Mistral (Jiang et al., 2023) and Gemma (Team et al., 2024) models, using Alpaca (Taori et al., 2023) and Flan v2 (Longpre et al., 2023) datasets. For evaluation, we utilize the Massively Multitask Language Understanding (MMLU) (Hendrycks et al., 2020), the zero-shot CommonsenseQA benchmarks—e.g., HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020),

Method	#Bits	No extra PTQ	No extra #params	MMLU				
				Hums.	STEM	Social	Other	Avg.
LLaMA-7B	16	–	–	33.3	29.8	37.8	38.0	34.6
QLoRA	4+16&16	Yes	Yes	36.1	31.9	42.0	44.5	38.4
IR-QLoRA	4+16	No	No	38.6	34.6	45.2	45.5	40.8
IR-QLoRA(merged)	16	No	No	37.6	33.2	44.3	44.7	39.4
Q-BLoRA (ours)	4+16&16	**Yes**	**Yes**	43.4	36.1	42.1	46.0	**41.4**
PEQA	4	Yes	Yes	34.9	28.9	37.5	40.1	34.8
QLoRA w/ GPTQ	4	No	Yes	33.8	31.3	37.4	42.2	36.0
QA-LoRA	4	No	Yes	36.6	32.4	44.8	44.9	39.4
QA-BLoRA (ours)	4	**Yes**	**Yes**	43.3	36.0	40.8	44.3	**40.7**
LLaMA-13B	16	–	–	44.0	35.9	53.2	52.9	46.3
QLoRA	4+16&16	Yes	Yes	46.0	37.3	55.8	55.1	48.4
IR-QLoRA	4+16	No	No	47.2	39.0	56.5	55.0	49.3
IR-QLoRA(merged)	16	No	No	47.3	38.8	56.4	55.3	48.5
Q-BLoRA (ours)	4+16&16	**Yes**	**Yes**	52.8	45.6	51.5	55.4	**49.9**
PEQA	4	Yes	Yes	43.0	37.7	53.6	49.0	45.0
QLoRA w/ GPTQ	4	No	Yes	45.4	37.4	55.7	54.3	48.0
QA-LoRA	4	No	Yes	48.4	38.3	54.9	55.2	49.2
QA-BLoRA (ours)	4	**Yes**	**Yes**	51.4	46.2	48.5	53.0	**49.5**
LLaMA-33B	16	–	–	56.2	45.9	67.1	63.9	58.2
QLoRA	4+16&16	Yes	Yes	55.4	46.0	66.4	63.6	57.7
IR-QLoRA	4+16	No	No	56.7	46.7	66.5	63.2	58.2
Q-BLoRA (ours)	4+16&16	**Yes**	**Yes**	61.4	52.2	60.7	64.2	**59.0**
QA-LoRA	4	No	Yes	55.8	46.4	67.0	64.0	58.1
QA-BLoRA (ours)	4	**Yes**	**Yes**	61.6	52.4	57.7	63.2	**58.3**
LLaMA-65B	16	–	–	61.4	51.9	73.6	67.6	63.4
QLoRA	4+16&16	Yes	Yes	60.3	52.7	72.9	67.4	63.1
IR-QLoRA	4+16	No	No	60.1	50.1	74.4	68.7	63.1
Q-BLoRA (ours)	4+16&16	**Yes**	**Yes**	65.2	58.2	64.4	68.6	**63.7**
QA-LoRA	4	No	Yes	60.8	50.5	72.5	66.7	62.5
QA-BLoRA (ours)	4	**Yes**	**Yes**	64.8	58.2	63.8	68.1	**63.3**

Table 1: Accuracy (%) comparison of LLaMA on the MMLU fine-tuned on Alpaca dataset.

WinoGrande (Sakaguchi et al., 2021), ARC (Clark et al., 2018), BoolQ (Clark et al., 2019), and OpenBookQA (Mihaylov et al., 2018)—and evaluations using GPT-4 / ChatGPT, including Vicuna (Chiang et al., 2023), Koala (Vu et al., 2023), WizardLM (Xu et al., 2023a), Self-instruct (Wang et al., 2023), and LIMA (Zhou et al., 2024). We follow the official code and configuration of QLoRA (Dettmers et al., 2024), IR-QLoRA (Qin et al., 2024), QA-LoRA (Xu et al., 2023b), PEQA (Kim et al., 2024), and GPTQ (Achiam et al., 2023).

4.2 Main Results and Efficiency

Comparison Against Recent Competitors of Fine-Tuning LLaMA for MMLU. Table 1 compares our Q-BLoRA and QA-BLoRA with SOTA LoRA-based quantization methods, including QLoRA (Dettmers et al., 2024), QA-LoRA (Xu et al., 2023b), and IR-QLoRA (Qin et al., 2024). We also compare with PEQA (Kim et al., 2024), which does not use LoRA but fine-tunes the quantized scaling instead. The ‘‘4+16’’ bit width indicates the inference model requires a 4-bit pre-trained model togethor with a 16-bit adapter, while ‘‘4+16&16’’ signifies the pre-trained model and adapter can be merged into a single 16-bit model, thereby enhancing inference efficiency.

Comprehensive results in Table 1 demonstrate that Q-BLoRA and QA-BLoRA consistently outperform all comparative methods by a convincing margin in their respective application scenarios. Q-BLoRA achieves the SOTA accuracy across models of various sizes. Compared to the most related method, QLoRA, our Q-BLoRA shows a significant improvement in accuracy. When compared to IR-QLoRA, the most competitive method in terms of accuracy, Q-BLoRA still achieves higher accuracy without requiring

Method	#Bit	MMLU				
		Hums.	STEM	Social	Other	Avg.
LLaMA2-7B	16	43.0	36.4	51.4	52.2	45.5
QLoRA	4+16&16	49.1	36.4	52.6	54.1	47.0
IR-QLoRA	4+16	43.4	36.8	51.9	53.6	46.2
IR-QLoRA(merged)	16	42.4	36.5	51.2	52.9	45.6
Q-BLoRA (ours)	4+16&16	50.0	42.5	51.3	51.2	**48.2**
QLoRA w/ GPTQ	4	47.2	36.2	51.0	51.6	45.1
QA-LoRA	4	42.1	34.4	49.1	50.3	43.9
QA-BLoRA (ours)	4	49.4	37.1	50.7	51.7	**46.3**
LLaMA2-13B	16	53.3	44.1	63.3	61.0	55.3
QLoRA	4+16&16	55.8	44.0	63.6	61.0	55.6
IR-QLoRA	4+16	51.9	43.9	61.9	60.4	54.4
Q-BLoRA (ours)	4+16&16	59.3	48.9	60.2	60.2	**56.2**
QLoRA w/ GPTQ	4	53.7	43.6	62.6	60.0	53.8
QA-BLoRA (ours)	4	57.8	46.8	60.2	59.7	**55.6**

Table 2: Accuracy (%) comparison of LLaMA2 on the MMLU fine-tuned on the Alpaca dataset.

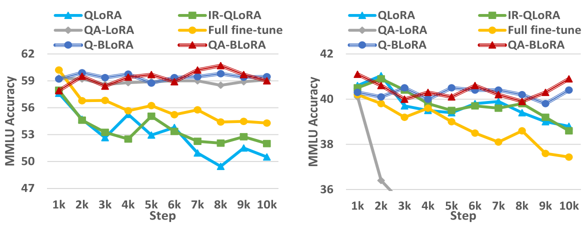
extra optimization for pre-trained model quantization or introducing extra trainable parameters. Moreover, IR-QLoRA alters the adapter's computational graph, preventing lossless merging with the pre-trained model and degrading performance after merging, whereas Q-BLoRA supports lossless merging.

QA-BLoRA can directly produce a 4-bit inference model through fine-tuning, making it more suitable for deployment on edge devices. The results show that QA-BLoRA has a significant accuracy advantage over other low-precision methods, whether QLoRA fine-tuning followed by GPTQ compression or the recent QAT method QA-LoRA. It even surpasses most fine-tuned 16-bit inference models in accuracy. Notably, our QA-BLoRA also does not require extra optimization for pre-trained model quantization or extra trainable parameters.

Performance on LLaMA2. As shown in Table 2, vanilla QLoRA with/without GPTQ exhibit remarkable model generalization capabilities. However, current SOTA variants (i.e., IR-QLoRA and QA-LoRA) show degraded performance. Only our Q-BLoRA and QA-BLoRA achieve accuracy superior to QLoRA. These results indicate the strong generalization across different LLM families for Q-BLoRA and QA-BLoRA.

Performance on Mistral and Gemma. As shown in Figure 5, Q-BLoRA and QA-BLoRA demonstrate significant performance advantages on the latest Mistral-7B and Gemma-2-2B.

It is noteworthy that during fine-tuning, the performance of full fine-tune, QLoRA,



(a) Results on Mistral-7B. (b) Results on Gemma-2-2B.

Figure 5: Comparison of the MMLU accuracy (%) during fine-tuning on the Alpaca dataset.

Method	#Bit	MMLU				
		Hums.	STEM	Social	Other	Avg.
LLaMA-7B	16	33.3	29.8	37.8	38.0	34.6
QLoRA	4+16&16	41.4	35.0	49.8	52.0	44.3
IR-QLoRA	4+16	44.2	39.3	54.5	52.9	47.4
Q-BLoRA (ours)	4+16&16	50.8	40.2	48.2	52.8	47.6
QLoRA w/ GPTQ	4	36.5	33.7	46.9	50.3	41.4
QA-LoRA	4	43.9	38.0	54.3	53.0	47.0
QA-BLoRA (ours)	4	50.6	40.0	48.1	52.3	47.2

Table 3: Accuracy (%) comparison of LLaMA on the MMLU fine-tuned on the FLAN v2 dataset.

and IR-QLoRA consistently degrades, while Q-BLoRA and QA-BLoRA maintain stable performance. This could be attributed to the fact that the latest models, such as Mistral and Gemma, have been pre-trained on high-quality data and diverse training strategies, with their model parameters carefully optimized (Jiang et al., 2023; Team et al., 2024; Huang et al., 2024). When further fine-tuned, significant parameter updates may disrupt the original model's convergence. In contrast, Q-BLoRA and QA-BLoRA mitigate overfitting and extreme parameter updates by employing neutral operations on adapter inputs and outputs (i.e., AvgPool and repeat_interleave), which helps preserve the model's original performance. While QA-LoRA applies similar compression operations and achieves good results on Mistral-7B, for the more parameter-efficient Gemma-2-2B, the lower dimensions of the adapter inputs lead to information loss and fine-tuning failure.

These results collectively demonstrate the effectiveness of Q-BLoRA and QA-BLoRA on the Mistral and Gemma models, aligning with the analysis presented in Section 3.2.

Fine-Tune Performance using the Flan v2 Dataset. We evaluated the fine-tuned models using the Flan v2 dataset to validate the performance of our methods on different datasets. As shown in Table 3, both our Q-BLoRA and

Method	#Bit	HellaSwag	ARC-e	ARC-c	PIQA	WinoGrande	BoolQ	OBQA	Avg.
LLaMA-7B	16	56.3	67.3	38.2	78.2	67.1	72.9	28.4	58.3
QLoRA	4+16&16	61.8	75.8	43.6	78.1	68.4	73.7	32.8	62.0
IR-QLoRA	4+16	54.7	76.6	45.1	78.8	72.6	80.6	37.2	63.7
Q-BLoRA (ours)	4+16&16	76.3	57.2	43.8	78.7	70.2	78.4	60.4	**66.4**
QLoRA w/ GPTQ	4	57.4	70.9	41.8	77.6	66.2	73.5	31.2	59.8
QA-LoRA	4	58.6	71.2	43.9	78.0	66.9	79.9	43.0	61.8
QA-BLoRA (ours)	4	76.3	56.6	43.4	79.2	70.2	77.9	58.2	**66.0**
LLaMA2-7B	16	75.0	60.4	40.3	75.2	65.8	75.2	65.6	65.4
QLoRA	4+16&16	75.6	68.9	45.7	77.8	68.6	77.2	72.2	69.4
IR-QLoRA	4+16	75.4	68.9	45.5	77.0	68.4	77.0	69.4	68.8
Q-BLoRA (ours)	4+16&16	76.3	70.0	46.8	78.6	70.2	77.5	69.4	**69.8**
QLoRA w/ GPTQ	4	75.1	66.2	44.5	76.2	67.1	73.5	69.4	67.4
QA-LoRA	4	75.0	65.6	44.2	75.8	66.7	73.2	69.2	67.1
QA-BLoRA (ours)	4	75.6	65.8	48.8	78.7	69.9	77.5	65.4	**68.8**

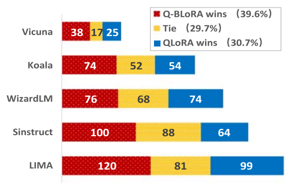
Table 4: Accuracy (%) comparison on the 0-shot commonsense QA.

QA-BLoRA achieve the highest accuracy in their respective application scenarios.

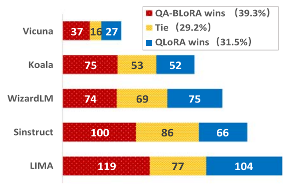
Performance on Commonsense QA. We also evaluate our Q-BLoRA and QA-BLoRA for zero-shot commonsense QA, with results summarized in Table 4. The results are very similar to those on MMLU. Our Q-BLoRA consistently achieves SOTA performance, and our QA-BLoRA also significantly outperforms other methods on 4-bit inference models. Notably, on LLaMA2, only our methods surpass QLoRA and the 4-bit model obtained by QLoRA w/GPTQ, while other methods fail to exceed the benchmarks.

Evaluations using GPT-4/ChatGPT. We present the evaluation results of content generated by Q-BLoRA and QA-BLoRA compared to QLoRA, using GPT-4/ChatGPT as the evaluator. This evaluation simulates human judgment, offering unique insights that complement automated metrics and provide an alternative perspective on the performance of fine-tuned models. Each response is scored by GPT-4/ChatGPT on a 1–10 scale. To mitigate positional bias, the responses are presented in two different orders. The evaluation criteria are defined as follows:

- Win: Outperforms in both orderings or wins in one and ties in the other.
- Tie: Ties in both orderings or wins in one and loses in the other.
- Loss: Performs worse in both orderings or ties in one and loses in the other.



(a) Q-BLoRA vs. QLoRA.



(b) QA-BLoRA vs. QLoRA.

Figure 6: Comparison of GPT-4/ChatGPT evaluations.

The Alpaca dataset and LLaMA-7B are used for fine-tuning in this evaluation. The results in Figure 6 demonstrate that our proposed Q-BLoRA and QA-BLoRA outperform QLoRA in nearly all tests, highlighting the superior performance of Q-BLoRA and QA-BLoRA.

The Efficiency of Q-BLoRA and QA-BLoRA. Table 5 compares the memory usage, fine-tuning time, and accuracy of various fine-tuning methods. The key findings are as follows: (i) Compared to non-quantized fine-tuning, both Q-BLoRA and QA-BLoRA significantly reduce the memory required for fine-tuning while maintaining similar accuracy. (ii) Compared to Q-LoRA, Q-BLoRA not only improves accuracy under similar fine-tuning times but also outperforms IR-QLoRA in terms of both accuracy and fine-tuning time, as well as inference efficiency. (iii) QA-BLoRA further reduces the size of the inference model. Compared to QLoRA with GPTQ, QA-BLoRA achieves a significant improvement in accuracy without requiring additional PTQ operations, and the fine-tuning time is shorter. When compared to QA-LoRA, QA-BLoRA shows significant advantages in both accuracy and

Method	BS	Mem-FT	Mem-Infer.	Time (h)	Acc.
Full fine-tune	–	13.5+32G	13.5G	–	41.0
LoRA	–	13.5+4.9G	13.5G	10.7	41.3
BLoRA	–	13.5+4.9G	13.5G	10.8	41.3
QLoRA	32	3.8+4.9G	13.5G	13.1	38.4
IR-QLoRA	32	3.8+4.9G	13.5+0.2G	14.0	40.8
Q-BLoRA	32	3.8+4.9G	13.5G	13.2	**41.4**
QLoRA w/GPTQ	32	3.8+4.9G	3.8G	13.1+0.1	36.0
QA-LoRA	32	3.8+2.7G	3.8G	0.1+15.8	36.4
QA-BLoRA	32	3.8+4.9G	3.8G	13.0	**40.7**
QA-BLoRA	64	3.8+2.5G	3.8G	12.8	**38.2**

Table 5: Efficiency comparison: block size (BS), memory requirements for fine-tuning (Mem-FT), and inference (Mem-Infer.) (e.g., ‘‘13.5+4.9G’’ indicates 13.5GB for the pre-trained model and 4.9GB for the adapter), fine-tuning time (Time), and accuracy on MMLU with Alpaca (Acc.). All results are reported with one GeForce RTX 3090 GPU.

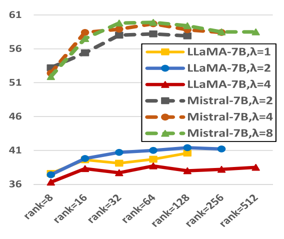
fine-tuning time. (iv) Additionally, the results show that non-quantized BLoRA does not lead to significant accuracy improvements compared to full fine-tuning or LoRA. However, in the case of quantization, our methods significantly improve memory efficiency while maintaining comparable performance. These results indicate that, in the non-quantized case, no underfitting occurs, while in the quantized case, our methods alleviate underfitting and thus achieve better model performance.

These results demonstrate the superior efficiency of Q-BLoRA and QA-BLoRA.

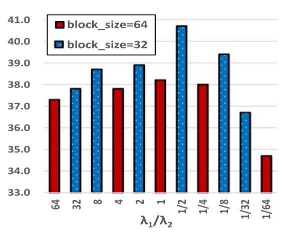
4.3 Ablation Study

The Balancing Factor of Q-BLoRA. Q-BLoRA introduces a new model-specific hyperparameter λ. As shown in Figure 7a, when fine-tuning LLaMA-7B, setting $\lambda = 2$ yields better performance, while for Mistral-7B, $\lambda = 8$ achieves superior results. We determine the value through experiments and recommend $\lambda = 2$ for the LLaMA/LLaMA2 model family and $\lambda = 8$ for Mistral and Gemma models.

As discussed in Figure 5, the SOTA LLMs benefit from high-quality data and meticulously designed training strategies during pre-training, leading to well-converged model parameters. Consequently, conventional fine-tuning approaches may compromise the original model's performance. Referring to the analysis in Figure 2, our method effectively suppresses drastic parameter updates. By employing a larger λ for SOTA LLMs,



(a) Impact of the rank and the scale factor of Q-BLoRA.



(b) Impact of the scale factor of QA-BLoRA.

Figure 7: Ablation analysis.

we can further preserve the original performance, thereby mitigating the negative impact of regular fine-tuning. Additionally, we observe that using a lower learning rate also helps restrain parameter updates and alleviates performance degradation to some extent. Exploring how to integrate our method to develop optimized fine-tuning strategies for SOTA LLMs will be a focus of future work.

The Balancing Factor of QA-BLoRA. In QA-BLoRA, the values of λ_1 and λ_2 are aligned with the block quantization of the pre-trained model. In LLMs, the block size for quantization is typically set to 32 or 64 (Dettmers and Zettlemoyer, 2023; Xu et al., 2023b). Figure 7b shows the fine-tuning results for different quantization blocks (with corresponding values of λ_1 and λ_2) under block sizes of 32 and 64. It can be observed that when the block size is 64, the best performance is achieved with $\lambda_1 = \lambda_2 = 8$, whereas for a block size of 32, the best performance occurs with $\lambda_1 = 4$ and $\lambda_2 = 8$.

The Scale Operators. We use alternative scale operators to study the effects of other non-parameter operators. Besides the AvgPool & repeat_interleave, we also tried truncation & supplement_0 and interval_sampling & interpolation. The specific descriptions are as follows:

- AvgPool & repeat_interleave: see Section 3.2.
- truncation & supplement_0: Truncate the input to retain only the front part; restore the output to the original size using zero-padding.
- interval_sampling & interpolation: Sample values at regular intervals; restore the output to the original size using zero-interpolation.

Operator	Q-BLoRA		QA-BLoRA	
	$\lambda = 2$	$\lambda = 4$	$\lambda_1 = 4, \lambda_2 = 8$	$\lambda_1 = \lambda_2 = 8$
AvgPool&repeat_interleave	**41.4**	**38.2**	**40.7**	**38.2**
truncation&supplement_0	36.6	35.8	37.0	35.9
int_sampling&interpolation	36.8	36.4	36.8	36.0

Table 6: Accuracy (%) comparison for operators on the MMLU fine-tuned on the Alpaca dataset.

As shown in Table 6, other scaling operators are significantly less effective compared to our AvgPool & repeat_interleave operator under different settings. This is due to the presence of outliers in certain channels, which carry high information content. Completely discarding a portion of outliers has a significant impact on performance. Furthermore, as discussed in Section 3.2, the neutral operation of AvgPool & repeat_interleave reduces the gap between values across different channels, thereby lowering the difficulty of fitting. In contrast, other operators not only lose valuable information from outliers, but also fail to effectively balance the values across channels.

5 Conclusion

In this paper, we propose two methods for fine-tuning quantized LLMs: Q-BLoRA and QA-BLoRA. Q-BLoRA addresses underfitting during fine-tuning by balancing the complexity of the adapter's input and output with the rank of the adapter. Additionally, it enables lossless merging of adapter parameters after fine-tuning, resulting in high-precision floating-point inference models. Building on the parameter merging approach of Q-BLoRA, QA-BLoRA achieves quantization-aware fine-tuning of low-rank adaptation by aligning with the block-wise quantization of the pre-trained model. This allows for the direct acquisition of low-precision inference models. Both Q-BLoRA and QA-BLoRA are easy to implement and demonstrate significant superiority across various models and evaluation tasks.

Acknowledgments

We would like to thank the anonymous reviewers and the action editor for their insightful comments. This work is sponsored in part by the National Natural Science Foundation of China (no. 62025208 and 62421002).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*.

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439. `https://doi.org/10.1609/aaai.v34i05.6239`

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

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*.

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, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

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

Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2023. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

Tim Dettmers and Luke Zettlemoyer. 2023. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pages 7750–7774. PMLR.

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*.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2023. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *The Eleventh International Conference on Learning Representations*.

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

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

Edward J. Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Wei Huang, Xudong Ma, Haotong Qin, Xingyu Zheng, Chengtao Lv, Hong Chen, Jie Luo, Xiaojuan Qi, Xianglong Liu, and Michele Magno. 2024. How good are low-bit quantized llama3 models? An empirical study. *arXiv preprint arXiv: arXiv:2404.14047*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Ting Jiang, Shaohan Huang, Shengyue Luo, Zihan Zhang, Haizhen Huang, Furu Wei, Weiwei Deng, Feng Sun, Qi Zhang, Deqing Wang, and Fuzhen Zhuang. 2024. Mora: High-rank updating for parameter-efficient fine-tuning. *arXiv preprint arXiv:2405.12130*.

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.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2023. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Yulong Mao, Kaiyu Huang, Changhao Guan, Ganglin Bao, Fengran Mo, and Jinan Xu. 2024. Dora: Enhancing parameter-efficient fine-tuning with dynamic rank distribution. *arXiv preprint arXiv:2405.17357*.

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. *arXiv preprint arXiv:1809.02789*.

Haotong Qin, Xudong Ma, Xingyu Zheng, Xiaoyang Li, Yang Zhang, Shouda Liu, Jie Luo, Xianglong Liu, and Michele Magno. 2024. Accurate lora-finetuning quantization of llms via information retention. *arXiv preprint arXiv:2402.05445*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106. `https://doi.org/10.1145/3474381`

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.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L. Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu-hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Chunlin Tian, Zhan Shi, Zhijiang Guo, Li Li, and Cheng-Zhong Xu. 2024. Hydralora: An asymmetric lora architecture for efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:9565–9584.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien

Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 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, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. 2022. Dylora: Parameter efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. *arXiv preprint arXiv:2210.07558*.

Thuy Vu, Xuanli He, Gholamreza Haffari, and Ehsan Shareghi. 2023. Koala: An index for quantifying overlaps with pre-training corpora. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 90–98. https://doi.org/10.18653/v1/2023.emnlp-demo.7

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508. https://doi.org/10.18653/v1/2023.acl-long.754

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

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

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel

Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Xiaoxia Wu, Zhewei Yao, and Yuxiong He. 2023. Zeroquant-fp: A leap forward in llms post-training w4a8 quantization using floating-point formats. *arXiv preprint arXiv:2307.09782*.

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.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023a. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Yuhui Xu, Lingxi Xie, Xiaotao Gu, Xin Chen, Heng Chang, Hengheng Zhang, Zhengsu Chen, Xiaopeng Zhang, and Qi Tian. 2023b. Qa-lora: Quantization-aware low-rank adaptation of large language models. In *The Twelfth International Conference on Learning Representations*.

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.

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*.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. 2023a. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. *arXiv preprint arXiv:2305.18403*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023b. Adaptive budget allocation for parameter-efficient fine-tuning. In *International Conference on Learning Representations*. Openreview.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. 2023. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices. *arXiv preprint arXiv:2309.02411*.