QuZO: Quantized Zeroth-Order Fine-Tuning for Large Language Models

Jiajun Zhou[1,3†] **Yifan Yang**[1], **Kai Zhen**[2], **Ziyue Liu**[1], **Yequan Zhao**[1],
Ershad Banijamali[2], **Athanasios Mouchtaris**[2], **Ngai Wong**[3], **Zheng Zhang**[1]
[1]University of California, Santa Barbara, [2]Amazon AGI
[3]The University of Hong Kong
{jjzhou,nwong}@eee.hku.hk, zhengzhang@ece.ucsb.edu

Abstract

Language Models (LLMs) are often quantized to lower precision to reduce the memory cost and latency in inference. However, quantization often degrades model performance, thus fine-tuning is required for various down-stream tasks. Traditional fine-tuning methods such as stochastic gradient descent and Adam optimization require backpropagation, which are error-prone in the low-precision settings. To overcome these limitations, we propose the Quantized Zeroth-Order (QuZO) framework, specifically designed for fine-tuning LLMs through low-precision (e.g., 4- or 8-bit) forward passes. Our method can avoid the error-prone low-precision straight-through estimator, and utilizes optimized stochastic rounding to mitigate the increased bias. QuZO simplifies the training process, while achieving results comparable to first-order methods in FP8 and superior accuracy in INT8 and INT4 training. Experiments demonstrate that low-bit training QuZO achieves performance comparable to MeZO optimization on GLUE, Multi-Choice, and Generation tasks, while reducing memory cost by 2.94× in LLaMA2-7B fine-tuning compared to quantized first-order methods.

1 Introduction

Large Language Models (LLMs) have achieved state-of-the-art performance in natural language processing, impacting various science and engineering fields. However, deploying and fine-tuning LLMs consumes significant hardware resources because of their huge model size. To address this issue, extensive research has focused on LLM quantization (Brown et al., 2020; Yuan et al., 2024). Notable approaches include post-training quantization (Yao et al., 2022; Wu et al., 2023), quantization-aware training (Bhalgat et al., 2020; Liu et al., 2023c; Nagel et al., 2021), and fully quantized training (Choukroun et al., 2019; Xi et al., 2023; Markidis et al., 2018). Post-training quantization can effectively reduce the latency and memory costs of inference, but often leads to a significant accuracy drop in low-precision formats, although various techniques (Shao et al., 2023; Xiao et al., 2023; Lin et al., 2023; Liu et al., 2023c) can partially mitigate this issue. Quantization-aware training (Liu et al., 2023a) offers better accuracy, but is more expensive due to the use of high-precision computational graphs. Truly quantized training methods employ low-precision gradients, activation, and weights to reduce hardware costs (Wang et al., 2018b; Banner et al., 2018; Micikevicius et al., 2017). However, implementing truly quantized training requires advanced hardware and software support for both forward and backward passes. Meanwhile, the straight-through estimator (Yin et al., 2019), which is commonly used for quantized gradient estimations, often causes unstable and inaccurate results in low-bit training.

In practice, LLM users may afford only a low-cost LLM inference engine (e.g., an edge FPGA or embedded system) with limited precision (e.g., INT8 or INT4). This paper asks the following question: *Can we leverage inference-only quantized hardware to fine-tune low-bit LLMs while achieving good performance?* This seems challenging because (1) inference-only hardware lacks sufficient memory bandwidth and storage to retain intermediate activations required for backpropagation, and (2) the Straight-Through Estimator (STE) introduces increasing gradient approximation errors in lower-bit formats (Malinovskii et al., 2024).

The recent MeZO (Malladi et al., 2024) enables memory-efficient zeroth-order (ZO) fine-tuning for LLMs, but suffers from an avoidable performance drop compared to first-order (FO) methods due to the bias and variance of ZO gradient estimation. In this paper, we show that *a quantized zeroth-order optimizer (QuZO) can achieve better accuracy than its first-order counterparts in a low-precision set-*

[†]Work undertaken during the visit at UC Santa Barbara

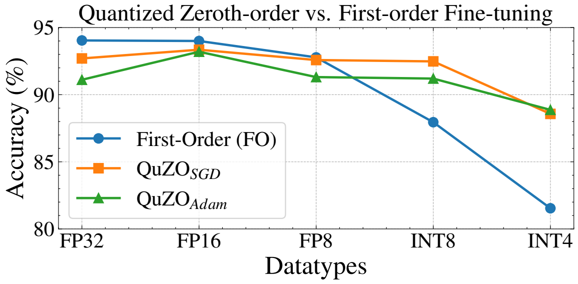


Figure 1: The proposed QuZO provides higher fine-tuning accuracy than first-order (FO) methods in ultra-low precision on the RoBERTa-Large model.

ting. Fig. 1 shows that both the QuZO and FO methods experience accuracy drops as the quantization precision decreases, which is well expected. However, QuZO consistently outperforms FO methods when the quantization precision is INT8 or below. Unlike traditional FO quantized training that depends on error-prone STE (Yin et al., 2019), our QuZO optimizer is more resistant to quantization error. Our contributions are summarized below.

- We identify the challenge of naive quantized ZO training, and propose a stochastic quantized perturbation method with theoretical soundness to reduce bias in quantized ZO gradient estimation.
- We introduce the implementation of QuZO as a plugin that integrates seamlessly with a quantized LLM inference engine, enabling accurate fine-tuning of low-bit LMs without backpropagation.
- We provide detailed numerical analysis about the proposed gradient estimator and the QuZO training framework. We show the benefit of our quantized ZO gradient estimator and the better training behavior of QuZO in low-bit LLM fine-tuning (especially INT4-format trainig).
- We apply QuZO to fine-tune 4/8-bit LLMs using both full-model fine-tuning and Low-Rank Adaptation (LoRA). QuZO achieves much better accuracy than quantized first-order training while reducing the memory cost by $1.4\times - 2.94\times$.

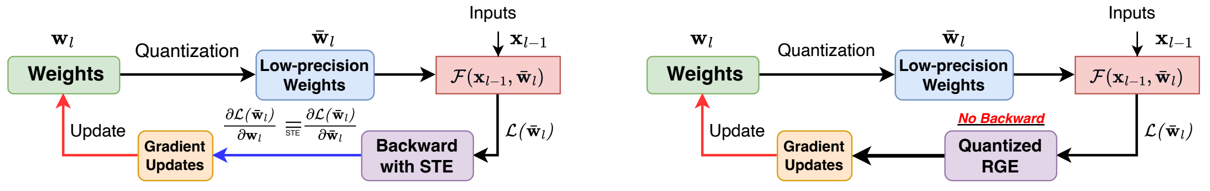
2 Related Work

Zeroth-order method. Zero-order (ZO) optimization techniques use forward passes for gradient estimation. Since backpropagation is not required during training, ZO methods reduce memory consumption significantly compared to a FO method. MeZO (Malladi et al., 2024) employed a memory-efficient ZO stochastic gradient descent (ZO-SGD) algorithm to efficiently fine-tune LLMs exceeding 60 billion parameters, leveraging parameter-efficient approaches (Yang et al., 2024b; Liu et al., 2021) like LoRA (Hu et al., 2021). Other ZO methods include ZO-SGD (Ghadimi and Lan, 2013) and ZO-Sign-SGD (Liu et al., 2018) using sign-based gradient estimation, the ZO-Adam (Chen et al., 2019) optimizer exploiting momentum information, and parameter-efficient methods like AdaZeta (Yang et al., 2024a). Sparse MeZO (Liu et al., 2024) employs a sparse perturbation for LLM fine-tuning. FP16 ZO training (Zhang et al., 2024) performs well but still faces memory bottlenecks. Recent ZO quantization introduces fixed-point 16-bit but fails at 8-bit (Feng et al., 2024). However, we overcome the challenges of lower-precision quantization and enable accurate fine-tuning of LLMs below 8-bit quantization.

Quantization of LLMs. Various quantization methods have been developed to reduce the memory and computing cost of LLMs. LLM.int8() (Dettmers et al., 2022) reduces the precision of model weights while keeping outliers in FP16. SmoothQuant (Xiao et al., 2023) introduces a fine-grained quantization method that supports INT8 operations exclusively. QLLM (Liu et al., 2023a) addresses the outlier problem via employing an adaptive channel reassembly technique. LLM-QAT (Liu et al., 2023c) employs quantization-aware training with a data-free strategy to achieve 4-bit quantization. Furthermore, the 4-bit training (Xi et al., 2023) and QLoRA (Dettmers et al., 2024) methods leverage a Hadamard Transform and a novel NF4 datatype, respectively, to accelerate training while preserving performance. While prior quantized training methods rely on backpropagation for gradient updates, our QuZO method eliminates the error-prune STE-based back propagation and used low-bit inference for truly quantized fine-tuning.

3 The QuZO Fine-Tuning Method

We start with a high-level introduction to our QuZO framework. Given a quantized LLM inference model, QuZO uses a low-bit ZO optimizer to update quantized model parameters directly during training. We assume that the forward pass $\mathbf{x}_l = \mathcal{F}(\mathbf{x}_{l-1}, \bar{\mathbf{w}}_l)$ computes the output of the l-th layer using the quantized weight matrix $\bar{\mathbf{w}}_l$ and the previous-layer feature $\mathbf{x}_{l-1}$, as shown in Fig. 2 (b). With just a few forward passes, our QuZO framework uses quantized RGE (see Section 3.2)


Inputs
$\mathbf{w}_l$
Quantization
$\bar{\mathbf{w}}_l$
$\mathbf{x}_{l-1}$
Weights
Low-precision Weights
$\mathcal{F}(\mathbf{x}_{l-1}, \bar{\mathbf{w}}_l)$
Update
$\frac{\partial \mathcal{L}(\bar{\mathbf{w}}_l)}{\partial \mathbf{w}_l} \underset{\text{STE}}{=} \frac{\partial \mathcal{L}(\bar{\mathbf{w}}_l)}{\partial \bar{\mathbf{w}}_l}$
$\mathcal{L}(\bar{\mathbf{w}}_l)$
Gradient Updates
Backward with STE
Inputs
$\mathbf{w}_l$
Quantization
$\bar{\mathbf{w}}_l$
$\mathbf{x}_{l-1}$
Weights
Low-precision Weights
$\mathcal{F}(\mathbf{x}_{l-1}, \bar{\mathbf{w}}_l)$
Update
No Backward
$\mathcal{L}(\bar{\mathbf{w}}_l)$
Gradient Updates
Quantized RGE

(a) Quantized First-Order (FO) Training

(b) Quantized Zeroth-Order (ZO) Training

Figure 2: Computational graphs for quantized first-order (FO) and zeroth-order (ZO) training.

to estimate ZO gradients, eliminating the need for BP in model updates. This approach fundamentally differs from existing quantized training methods shown in FigFig. 2 (a), which uses STE in the BP to approximate quantized gradient $\frac{\partial \mathcal{L}(\bar{\mathbf{w}})}{\partial \bar{\mathbf{w}}_l}$. Our method avoids the straight-through estimator (STE) (Yin et al., 2019) used in truly quantized FO training, enabling high-accuracy training on a low-precision hardware platform.

In the following, we first show the challenges of ZO-SGD in the quantized setting, and then propose a solution to address this fundamental challenge.

3.1 Challenges of Quantized ZO Training

Standard ZO-SGD uses a randomized gradient estimator (RGE) (Nesterov and Spokoiny, 2017; Ghadimi and Lan, 2013) to approximate a full-precision gradient. Specifically, given full-precision model parameters $\mathbf{w} \in \mathbb{R}^d$, a loss function $\mathcal{L}(\mathbf{w}, \mathcal{B})$ and a minibatch of dataset $\mathcal{B}$, RGE computes the gradient as:

$$\begin{aligned}\nabla \hat{\mathcal{L}}(\mathbf{w}) &= \sum_{i=1}^{n} \frac{\mathcal{L}_{\mathcal{B}}(\mathbf{w} + \epsilon \mathbf{u}_i) - \mathcal{L}_{\mathcal{B}}(\mathbf{w} - \epsilon \mathbf{u}_i)}{2n\epsilon} \mathbf{u}_i \\ &\approx \frac{1}{n} \sum_{i=1}^{n} \mathbf{u}_i \mathbf{u}_i^T \nabla \mathcal{L}_{\mathcal{B}}(\mathbf{w}), \qquad (1)\end{aligned}$$

where ϵ is a scaling factor, $\{\mathbf{u}_i\}_{i=1}^n$ are i.i.d. samples drawn from certain distributions with a unit variance (e.g., a standard Gaussian distribution). While $\nabla \hat{\mathcal{L}}(\mathbf{w})$ differs from the true gradient $\nabla \mathcal{L}_{\mathcal{B}}(\mathbf{w})$, its expectation serves as a good gradient estimator because

$$\begin{aligned}\mathbb{E}\left[\nabla \hat{\mathcal{L}}(\mathbf{w})\right] &\approx \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\left(\mathbf{u}_i \mathbf{u}_i^T\right) \nabla \mathcal{L}_{\mathcal{B}}(\mathbf{w}) \\ &= \nabla \mathcal{L}_{\mathcal{B}}(\mathbf{w}). \qquad (2)\end{aligned}$$

This statistical property ensures the asymptotical convergence of ZO-SGD. Assuming the quantized model parameters $\bar{\mathbf{w}}$ are available and only low-precision hardware is used for inference, the full-precision random perturbation $\mathbf{u}_i$ cannot be directly applied to $\bar{\mathbf{w}}$ due to hardware limitations. To address this, $\mathbf{u}_i$ is replaced with its quantized counterpart $\hat{\mathbf{u}}_i = Q(\mathbf{u}_i)$, leading to a low-precision RGE:

$$\begin{aligned}\nabla \hat{\mathcal{L}}(\bar{\mathbf{w}}) &= \sum_{i=1}^{n} \frac{\mathcal{L}_{\mathcal{B}}\left(\bar{\mathbf{w}} + \epsilon \hat{\mathbf{u}}_i\right) - \mathcal{L}_{\mathcal{B}}\left(\bar{\mathbf{w}} - \epsilon \hat{\mathbf{u}}_i\right)}{2n\epsilon} \hat{\mathbf{u}}_i \\ &\approx \frac{1}{n} \sum_{i=1}^{n} \hat{\mathbf{u}}_i \hat{\mathbf{u}}_i^T \nabla \mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}}). \qquad (3)\end{aligned}$$

Taking the exception values on both sides, we have

$$\begin{aligned}\mathbb{E}\left[\nabla \hat{\mathcal{L}}(\bar{\mathbf{w}})\right] &\approx \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}\left(\hat{\mathbf{u}}_i \hat{\mathbf{u}}_i^T\right) \nabla \mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}}) \\ &\neq \nabla \mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}}) \qquad (4)\end{aligned}$$

Since the quantized perturbation $\hat{\mathbf{u}}_i = Q(\mathbf{u}_i)$ no longer maintains a unit variance, the above naive quantized RGE introduces bias during fine-tuning and may lead to divergence in training.

3.2 Proposed Quantized RGE

We propose a new quantized RGE scheme to address the challenge in the previous subsection.

Stochastic Quantization of $\mathbf{u}_i$. We first define a quantization operation of $Q(\mathbf{u}_i)$ based on stochastic rounding (Connolly et al., 2021):

$$\begin{aligned}Q(\mathbf{u}_i) &= \text{clamp}\Big(SQ, L_{\min}, L_{\max}\Big) + z_0, \\ SQ &= \Big(\lfloor s_u \mathbf{u}_i \rfloor + \text{Ber}(s_u \mathbf{u}_i - \lfloor s_u \mathbf{u}_i \rfloor\Big) \qquad (5)\end{aligned}$$

The stochastic quantization formula $Q(\mathbf{u}_i)$ converts the perturbation $\mathbf{u}_i$ into a low-bit representation by scaling it with a factor s_u as $s_u \mathbf{u}_i$, performing a downward rounding operation $\lfloor s_u \mathbf{u}_i \rfloor$, and applying stochastic up-rounding using a Bernoulli random variable $\text{Ber}(s_u \mathbf{u}_i - \lfloor s_u \mathbf{u}_i \rfloor)$. The resulting quantized value is clamped to the representable range $[L_{\min}, L_{\max}]$ and shifted by the zero point z_0. This stochastic rounding ensures that

$$\mathbb{E}_Q\left[Q(\mathbf{u}_i)\right] = \mathbb{E}\left[\mathbf{u}_i\right]. \qquad (6)$$

We can produce two different quantization results by using two random seeds in the stochastic rounding full-precision $\mathbf{u}_i$:

$$\begin{aligned}
&\mathbf{u}_{i,1} = Q_1(\mathbf{u}_i) = Q(\mathbf{u}_i) \text{ with random seed } i_1;\\
&\mathbf{u}_{i,2} = Q_2(\mathbf{u}_i) = Q(\mathbf{u}_i) \text{ with random seed } i_2;\\
&\mathbf{u}_{i,1} \neq \mathbf{u}_{i,2}. \qquad (7)
\end{aligned}$$

The above stochastic quantizations ensure that (1) the expectation of the quantized perturbations $\mathbf{u}_{i,1}$ and $\mathbf{u}_{i,2}$ equals the original perturbation $\mathbf{u}_i$, (2) $\mathbf{u}i, 1$ and $\mathbf{u}_{i,2}$ are conditionally independent to each other. As a result, we have

$$\begin{aligned}
&\mathbb{E}_{Q_1}(\mathbf{u}_{i,1}) = \mathbb{E}_{Q_2}(\mathbf{u}_{i,2}) = \mathbf{u}_i,\\
&\mathbb{E}_{Q_1,Q_2}(\mathbf{u}_{i,1}\mathbf{u}_{i,2}^T) = \mathbb{E}_{Q_1}(\mathbf{u}_{i,1})\mathbb{E}_{Q_2}(\mathbf{u}_{i,2}^T) = \mathbf{u}_i\mathbf{u}_i^T.
\end{aligned}$$

Our Quantized RGE. With the two conditionally independent quantized vectors $\mathbf{u}_{i,1}$ and $\mathbf{u}_{i,2}$ defined in Eq. (7), we propose the following quantized RGE:

$$\nabla\hat{\mathcal{L}}(\bar{\mathbf{w}}) = \sum_{i=1}^{n} \frac{\mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}}+\epsilon\mathbf{u}_{i,1})-\mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}}-\epsilon\mathbf{u}_{i,1})}{2n\epsilon}\mathbf{u}_{i,2} \quad (8)$$

As $\epsilon \to 0$, the RGE result is

$$\nabla\hat{\mathcal{L}}(\bar{\mathbf{w}}) \approx \frac{1}{n}\sum_{i=1}^{n}\mathbf{u}_{i,1}\mathbf{u}_{i,2}^T\nabla\mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}}). \quad (9)$$

The estimation results depend on three random vectors and functions: $\mathbf{u}_i$, Q_1 and Q_2. Taking expectation values on both sides of Eq. (9), we have

$$\begin{aligned}
&\mathbb{E}\left[\nabla\hat{\mathcal{L}}(\bar{\mathbf{w}})\right] \approx \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\mathbf{u_i},Q_1,Q_2}\left[\mathbf{u}_{i,1}\mathbf{u}_{i,2}^T\right]\nabla\mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}})\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}_{\mathbf{u_i}}\left[\mathbb{E}_{Q_1,Q_2}\left[\mathbf{u}_{i,1}\mathbf{u}_{i,2}^T\right]\right]\nabla\mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}})\\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left(\mathbf{u}_i\mathbf{u}_i^T\right)\nabla\mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}})\\
&= \nabla\mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}}). \qquad (10)
\end{aligned}$$

The expectation value of our quantized RGE remains a reliable estimator of the true gradient, which is similar to the full-precision RGE. This indicates that our proposed RGE will ensure asymptotical convergence as in a full-precision ZO method. This theoretical property ensures excellent training performance even in low-precision settings (e.g. INT8 and INT4).

3.3 Implementation of QuZO

Now we present the details of the implementation of the QuZO framework.

Quantized Model Updates. Recall that in full-precision ZO-SGD, the gradient is computed in (1), and the model parameters are updated as

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \cdot \nabla\hat{\mathcal{L}}(\mathbf{w}_t) \quad (11)$$

where $\mathbf{w}_t$ represents the model parameters at iteration t, η_t is the learning rate and $\nabla\hat{\mathcal{L}}(\mathbf{w}_t)$ denotes the estimated gradient of the loss function. Since $\mathbf{w}_t \approx s_w\bar{\mathbf{w}}_t$, and s_w is a scaling factor used in the quantization $\bar{\mathbf{w}}_t = Q\left(\mathbf{w}_t/s_w\right)$, with $Q[\cdot]$ representing the stochastic quantization applied to the parameters. This approximation suggests:

$$\mathbf{w}_{t+1} \approx s_w\left[\bar{\mathbf{w}}_t - \eta_t \cdot \nabla\hat{\mathcal{L}}(\bar{\mathbf{w}}_t)\right] \quad (12)$$

To achieve a *truly quantized* training process suitable for low-precision hardware, the model parameters are updated as:

$$\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t - Q\left[\eta_t \cdot \nabla\hat{\mathcal{L}}(\bar{\mathbf{w}}_t)\right]. \quad (13)$$

To refine the update process, multiple steps can be used. For each query i, we compute

$$\mu_i = \frac{\mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}}+\epsilon\mathbf{u}_{i,1}) - \mathcal{L}_{\mathcal{B}}(\bar{\mathbf{w}}-\epsilon\mathbf{u}_{i,1})}{2\epsilon}. \quad (14)$$

Then the quantized model $\bar{\mathbf{W}}$ is updated as

$$\bar{\mathbf{w}}_{t+1} = \bar{\mathbf{w}}_t - \sum_{i=1}^{n} Q\left(\frac{\eta_t\mu_i}{n}\mathbf{u}_{i,2}\right). \quad (15)$$

Here $\mathbf{u}_{i,2}$ is a second quantized version of $\mathbf{u}_i$ as explained in Eq. (7). Stochastic rounding $Q[\cdot]$ ensures that no additional bias will be introduced when we update the LLM parameters directly at low precision.

Algorithm Flow. The pseudo codes of QuZO are summarized in Algorithm 1. For each query i, two forward passes are performed to determine the sensitivity (μ_i) of the loss function with respect to a quantized perturbation direction $\mathbf{u}_{i,1}$ (lines 5-11). The resulting low-precision gradient associated with each inquiry is obtained by quantizing a scaled version of $\mathbf{u}_{i,2}$, where the sensitivity (μ_i), the learning rate η_t, and the sample size n are taken into account. This low-precision ZO gradient allows us to directly update the quantized LLM model parameters with low-precision hardware.

Algorithm 1 QuZO: Quantized Zeroth-Order Training

```
Require: LLM model parameters $\mathbf{w} \in \mathbb{R}^d$, learning rate $\eta_t$, $T$ is the step, perturbation scaling factor $\epsilon$ and dataset $\mathcal{B}$.
Initial Pre-trained Model to Quantized Model or directly load a quantized model.
$\bar{\mathbf{w}} = \mathrm{Q}(\mathbf{w})$    ◁ Optionally, quantize the model if starting with a full-precision model
for t in $T$ do
    for i in $n$ do
        $\mathbf{u}_{i,1} \leftarrow Q_1(\mathbf{u}_i), \mathbf{u}_i \sim \mathcal{N}(0, \mathbb{I}_d)$    ◁ Quantize the perturbation $\mathbf{u}_i$ with a random seed $i_1$
        $\mathbf{u}_{i,2} \leftarrow Q_2(\mathbf{u}_i)$    ◁ Quantize the perturbation $\mathbf{u}_i$ with a random seed $i_2$
        $\bar{\mathbf{w}}_t \leftarrow \bar{\mathbf{w}}_t + \epsilon \cdot \mathbf{u}_{i,1}$    ◁ Low-bit stochastic perturbation updates $\bar{\mathbf{w}}_t$ using positive scaling
        $\mathcal{L}_1^i \leftarrow \mathcal{F}(\bar{\mathbf{w}}_t, \mathcal{B})$    ◁ First zeroth-order forward pass
        $\bar{\mathbf{w}}_t \leftarrow \bar{\mathbf{w}}_t - 2\epsilon \cdot \mathbf{u}_{i,1}$    ◁ Low-bit stochastic perturbation updates $\bar{\mathbf{w}}_t$ using negative scaling
        $\mathcal{L}_2^i \leftarrow \mathcal{F}(\bar{\mathbf{w}}_t, \mathcal{B})$    ◁ Second zeroth-order forward pass
        $\mu_i \leftarrow (\mathcal{L}_1^i - \mathcal{L}_2^i)/(2\epsilon)$    ◁ Sensitivity w.r.t. the quantized perturbation
        $\bar{\mathbf{w}}_t \leftarrow \bar{\mathbf{w}}_t + \epsilon \cdot \mathbf{u}_{i,1}$    ◁ Recover $\bar{\mathbf{w}}_t$ to its original state
        $\bar{\mathbf{w}}_{t+1} \leftarrow \bar{\mathbf{w}}_t - Q(\frac{\eta_t \mu_i}{n} \mathbf{u}_{i,2})$    ◁ Quantized LLM model update
    end for
end for
return $\bar{\mathbf{w}}$    ◁ Return a quantized model
```

QuZO for LoRA. We can extend the QuZO framework by incorporating low-rank adaptation to allow low-precision parameter-efficient fine-tuning. Our approach uses the model quantization strategies of QLoRA (Dettmers et al., 2024) and LLM.int8() (Dettmers et al., 2022) without modifying the quantized model. QuZO significantly reduces memory overhead by eliminating the storage of FO optimizer states and updating only the low-rank trainable matrices $\mathbf{A} \in \mathbb{R}^{d \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times d}$ using forward passes. In QuZO fine-tuning, the model parameters are quantized and frozen at low precision (e.g. 4 or 8 bits), and we update solely on the low-rank matrices $\mathbf{A}$ and $\mathbf{B}$. The trainable low-rank matrices are quantized (denoted as $Q[\mathbf{A}]$ and $Q[\mathbf{B}]$) in order to match the precision of the LLM . By doing so QuZO training can significantly further reduce the memory cost compared to traditional LoRA for 4/8-bit LLM fine-tuning.

3.4 QuZO Analysis

In this subsection, we analyze the quality of gradient estimation in QuZO and its impact to training.

QuZO Gradient Quality. We use a simple encoder-block transformer to analyze the asymptotic behavior of two quantized ZO gradient estimators. Q-RGE1 refers to the quantized estimate in Eq. (3), and Q-RGE2 denotes our proposed estimation in Eq. (8). Although we need only a few inquiries to compute actual ZO gradients, the statistical behavior of a gradient (rather than the value of the individual gradient) decides the training performance. To verify statistical asymptotic behavior, we set $n = 1000$ to perform a Monte Carlo computation to get empirical mean values of Q-RGE1 and Q-RGE2, and then compare them with a full-precision ZO gradient via the ℓ_2 error. As shown in Fig. 3 (a), the expected values of both quantized estimators have larger errors as the precision reduces from INT8 to INT3. However, our method (Q-RGE2) is much more resilient to quantization errors and has a more accurate expected value, since our quantized ZO gradient estimator can avoid the additional bias caused by quantization.

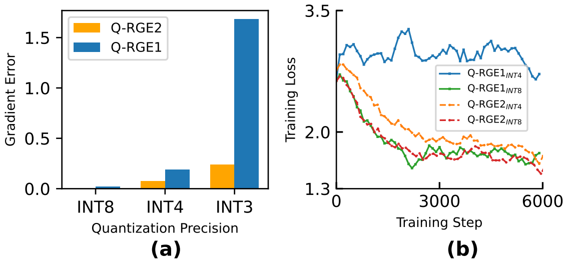


Figure 3: (a) Errors of quantized gradient estimation Q-RGE1 in Eq. (3) and our proposed Q-RGE2 in Eq. (8). (b) Training loss of low-precision ZO optimizer with these two quantized gradient estimators, respectively.

Training Behavior. Figure 3 (b) further shows the training behavior of quantized ZO optimization using these two gradient estimators when fine-tuning the OPT-1.3B model. Experiments are performed on the DROP dataset under 8-bit and 4-bit settings. We observe that our QuZO with Q-RGE2 shows slightly better convergence compared to quantized training using Q-RGE1 in the 8-bit setting. In 4-bit training, our method demonstrates a stable and significantly better training behavior: it achieves a loss similar to 8-bit training, while INT 4 Q-RGE1 causes convergence failures. The above analysis clearly demonstrates the better numerical performance of our QuZO in low-bit LLM fine-tuning.

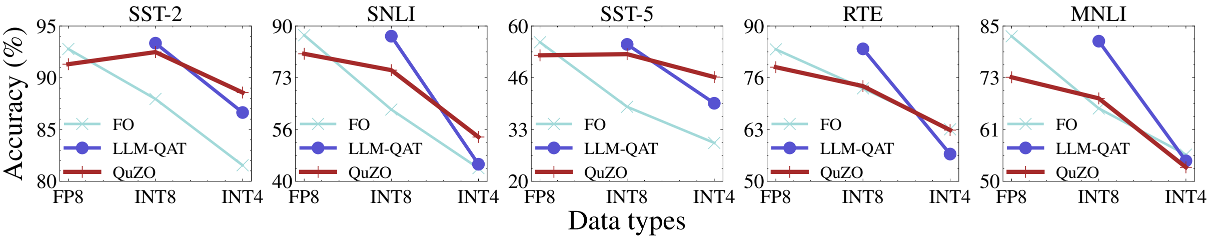


Figure 4: Experimental findings on RoBERTa-large (350M parameters) with prompts reveal that QuZO, leveraging full-parameter tuning, starts to surpass FO and LLM-QAT as precision reduces to INT8 or below.

Table 1: QuZO demonstrates superior performance in full-parameter fine-tuning of LLaMa-2 7B. Note : WaAb quantization configurations, which refer to a-bit weight quantization and b-bit activation quantization.

LLaMa-2 7B Model		Classification				Multiple-Choise		Generation	
Data Precision	Method	SST-2	RTE	WSC	MultiRC	COPA	ReCoRD	SQuAD	DROP
FP W32A32	FO	95.41	63.73	63.46	65.10	86.00	81.00	90.71	51.38
	MeZO	94.80	54.60	58.80	62.60	82.70	70.80	72.50	46.80
FP W8A8	FO	**91.63**	**63.90**	49.00	**58.00**	79.00	72.50	72.68	23.46
	QuZO	91.05	55.59	**65.38**	57.10	**80.00**	**76.80**	**76.38**	**30.17**
INT W8A8	FO	90.81	52.34	61.53	50.60	62.00	74.83	70.13	20.06
	SmoothQuant	91.05	**66.78**	59.51	**61.50**	72.02	79.10	73.07	29.94
	LLM.int8()	88.04	62.56	57.75	55.61	80.02	**80.61**	76.34	20.15
	QuZO	**92.00**	61.01	**63.46**	60.00	**81.00**	79.00	**77.71**	**30.11**
INT/FP W4A8	FO	86.35	47.29	60.57	51.90	62.04	73.21	30.01	10.06
	MinMax	88.1	59.91	41.28	53.21	82.51	80.97	50.07	24.71
	LLM-FP4	91.20	**66.82**	61.38	58.81	**82.90**	**81.25**	51.07	**24.99**
	QuZO	**91.62**	54.87	**62.28**	**60.60**	80.01	78.20	**68.12**	21.80

4 Experiments

In this section, we evaluate the proposed QuZO method on several language models (LMs) with 4-8 bit precision. QuZO demonstrates performance comparable to or better than standard first-order (FO) truly quantized training across various model sizes and tasks, with significantly lower memory usage. We also explore fine-tuning quantized models by combining QLoRA (Dettmers et al., 2024) with QuZO. For hardware costs, QuZO employs a forward-only framework with hardware requirements similar to post-training quantization. In Section 4.3, we compare the memory consumption between truly quantized FO training and QuZO. Furthermore, we employ both medium-size models (e.g. RoBERTa-Large (Liu et al., 2019)) and large decoder-based LMs [e.g. OPT 1.3B (Zhang et al., 2022) and LLaMa-2 7B (Touvron et al., 2023)] in few-shot settings. All experiments were carried out on NVIDIA A100-40GB GPUs. The details of the experimental setup are provided in Appendix A.

4.1 Full-Parameter Quantized Fine Tuning

We first summarize our experiments on full-parameter fine-tuning for medium- and large-scale models. These results demonstrate that QuZO provides a practical approach for accurate fine-tuning of quantized LLMs directly on low-precision hardware, maintaining. For medium-scale models like RoBERTa-Large, QuZO surpasses truly quantized FO fine-tuning in most tasks in the 4-bit precision. For large-scale models such as LLaMA-2, QuZO achieves performance comparable to or better than truly quantized FO fine-tuning, particularly under ultra-low bit configurations. These findings highlight the ability of QuZO to enable low-cost hardware training without compromising performance.

Performance on the RoBERTa-Large model. We evaluate the performance of various methods in the SST-2, SNLI, SST-5, RTE, and MNLI datasets and on the RoBERTa-Large model. The results in Fig. 4 leads to the following observations:

- As expected, all training methods experience accuracy decline as quantization precision decreases. This occurs because the model expressive power declines and the optimization becomes more challenging in lower precision.
- The performance of truly quantized FO fine-tuning drops most significantly due to the increasing errors in the straight-through estimators as precision decreases.
- Quantization-aware training (QAT) can mitigate partially the accuracy drop of truly quantized FO training. As a faked quantized training, QAT

Table 2: Results of parameter-efficient fine-tuning of quantized LLMs.

Model	Methods	Gradient	SST-2	MultiRC	ReCoRD	SQuAD	DROP
8bit LLaMa2-7B	FO (LoRA)	INT8	91.97	45.60	80.10	56.13	20.93
	MeZO (LoRA)	FP32	88.17	**63.60**	**80.60**	**86.96**	**37.23**
	QuZO (LoRA)	INT8	**90.36**	60.00	80.80	79.97	36.11
8bit OPT-1.3B	FO (LoRA)	INT8	**91.97**	55.30	70.50	**71.92**	18.35
	MeZO (LoRA)	FP32	89.56	**57.30**	**70.80**	68.11	18.53
	QuZO (LoRA)	INT8	88.76	55.90	70.20	66.53	**23.17**
4bit OPT-1.3B	FO (LoRA)	INT4	53.89	55.55	17.20	28.20	10.00
	MeZO (LoRA)	FP32	**90.48**	54.10	**69.50**	**70.93**	20.53
	QuZO (LoRA)	INT4	87.84	**56.20**	68.90	68.13	**21.92**

still needs backpropagation and updates LLM model parameters in full precision. Therefore, it remains memory-intensive and less suitable for resource-constrained low-precision hardware.

- In contrast, the performance of QuZO is **most resilient to the decreased precision**, and it works the best in a very low-precision (e.g., INT4). This is because (1) QuZO can bypass the error-prone straight-through estimator that is used in truly quantized FO training, and (2) the quantized RGE in Eqn.(8) can eliminate the bias caused by quantized perturbations.

Performance of QuZO on LLaMA Models. We further apply QuZO to fine-tune the LLaMa-2 model, evaluating it on SuperGLUE (Wang et al., 2019) and generation tasks. Table 1 shows that QuZO outperforms its truly quantized FO counterparts on all multichoice and generation tasks under FP W8A8 quantization (i.e. FP8 for both weights and activations). Under the INT W8A8 quantization, QuZO outperforms SmoothQuant, LLM.int8(), and truly quantized FO methods in 4 out of 8 tasks. For 4-bit quantized FO training, uniform quantization yields the worst accuracy, but advanced methods such as LLM-FP4 improve performance. LLM-FP4 (Liu et al., 2023b) and its baseline MinMax use FP W4A8 quantization and achieve a slight improvement in accuracy, particularly for multichoice tasks. Our QuZO method maintains strong performance under W4A8 quantization with mixed-datatype (see Appendix B), achieving the best results in 4 out of 8 tasks. SmoothQuant, LLM.int8(), MinMax, and LLM-FP4 have introduced efficient quantization methods that enhance performance. However, they are memory-intensive as they require fine-tuning using a FO optimizer.

Table 3: Performance Comparison of QuZO on the LLaMa-2 13B Model

Model (#Bit)	Methods	ReCoRD	SQuAD	DROP
LLaMa2-13B (8-Bit)	FO	81.70	63.23	25.90
	MeZO	82.10	63.71	25.20
	QuZO	**82.20**	**78.19**	**37.61**
LLaMa2-13B (4-Bit)	FO	82.00	62.27	25.31
	MeZO	**82.30**	62.62	25.33
	QuZO	82.10	**73.79**	**27.32**

4.2 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning methods like QLoRA (Dettmers et al., 2024) reduce memory usage with 4-bit precision compared to standard training but still rely on AdamW (Loshchilov, 2017), which requires backpropagation. QuZO improves inference efficiency and memory savings, achieving a 5.47× reduction in maximum memory cost compared to QLoRA in fine-tuning the 4-bit OPT-1.3B model (details in Appendix C).

Our QuZO framework applies the LoRA (rank set as 8), allowing fine-tuning with far fewer trainable parameters than full-model tuning, significantly reducing memory consumption, and accelerating convergence. Table 2 highlights the performance of QuZO with low-bit perturbation and gradient configurations for different tasks and models. For the OPT-1.3B model, QuZO utilizes INT8 RGE gradients with INT4 perturbations. Despite the introduction of low-bit gradients, QuZO achieves competitive or superior performance compared to full-precision MeZO with LoRA in most tasks and demonstrates strong robustness in 4-bit fine-tuning, while truly quantized FO shows poor accuracy in 4-bit training. Furthermore, QuZO reduces $2-5.47\times$ memory consumption compared to fully quantized FO methods in Table 15(see Appendix C). For the LLaMa2-7B model, QuZO achieves performance comparable to full-precision MeZO while allowing fine-tuning across all five

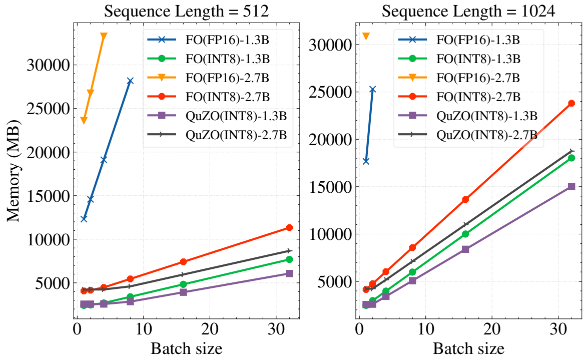


Figure 5: Peak memory usage of FP16 and INT8 training on the OPT 1.3B/2.7B model with sequence lengths of 512 (left) and 1024 (right).

tasks on a single GPU. In contrast, truly quantized FO methods encounter out-of-memory (OOM) issues. This result highlights that the low-bit stochastic perturbation of QuZO effectively reduces memory overhead while mitigating the bias caused by quantization errors, making accurate fine-tuning feasible on resource-constrained hardware.

Fine-Tuning 13B LMs. Table 3 presents the performance comparison of QuZO fine-tuning against other methods with LoRA, including First-Order (FO) and MeZO, on the LLaMa-2 13B model under 8-bit and 4-bit quantization. The evaluation is conducted on three datasets: ReCoRD, SQuAD, and DROP, which assess reading comprehension and reasoning ability. The results indicate that QuZO consistently outperforms MeZO and FO, particularly in SQuAD and DROP, demonstrating its ability to better retain performance in a quantized setting. In the 8-bit setting, QuZO achieves a significant improvement. In the 4-bit setting, the trend remains similar, highlighting the robustness of QuZO in handling more aggressive quantization.

4.3 Memory Efficiency

We follow Table 5 of (Zhang et al., 2024) to provide the theoretical analysis for different optimizers. As shown in Table 4, our QuZO demonstrates significant memory reduction compared to truly quantized FO fine-tuning at the same precision.

We further compare the empirical memory costs of full fine-tuning the LLaMA-2 7B model in Table 5. Specifically, in the MultiRC task, QuZO (8-bit) reduces memory usage by 1.43×, while QuZO (4-bit) achieves a 1.39× reduction compared to their truly quantized FO counterparts with the same precision. Similarly, in the SQuAD task, QuZO (8-bit) reduces memory consumption by 2.94×, and QuZO (4-bit) achieves a 2.89× reduction relative to FO-SGD at the same precision.

Table 4: Comparison of peak memory consumption during full-model fine-tuning. Note: model storage (Weight Mem.) and dynamic allocations for gradients (Dynamic Mem.). $|\mathbf{w}|$ and $|\mathbf{a}|$ denote memory usage for model parameters and intermediate parameters, respectively, with l representing a specific layer.

Method	Weight Mem.	Dynamic Mem
Full Precision Optimizer		
FO-SGD	$\lvert\mathbf{w}\rvert$	$\sum_l \max\{\lvert\mathbf{a}\rvert, \lvert\mathbf{w}\rvert\}$
MeZO	$\lvert\mathbf{w}\rvert$	$\max_l \lvert\mathbf{w}\rvert$
Optimizer with Low Precision Model		
FO(8-bit)	$\lvert\mathbf{w}\rvert/4$	$\sum_l \max\{\frac{\lvert\mathbf{a}\rvert}{4}, \frac{\lvert\mathbf{w}\rvert}{4}\}$
FO(4-bit)	$\lvert\mathbf{w}\rvert/8$	$\sum_l \max\{\frac{\lvert\mathbf{a}\rvert}{8}, \frac{\lvert\mathbf{w}\rvert}{8}\}$
QuZO(8-bit)	$\lvert\mathbf{w}\rvert/4$	$\max_l \frac{\lvert\mathbf{w}\rvert}{4}$
QuZO(4-bit)	$\lvert\mathbf{w}\rvert/8$	$\max_l \frac{\lvert\mathbf{w}\rvert}{8}$

Table 5: Total memory consumption (GB) for different optimizers on LLaMa-2 7B.

Method	MultiRC (GB)	SQuAD (GB)
FO-SGD (8-bit)	11.66	21.29
FO-SGD (4-bit)	6.28	10.73
QuZO (8-bit)	8.15	7.24
QuZO (4-bit)	4.52	3.71

To verify hardware efficiency, we profile the memory usage of our QuZO method with INT8 CUDA kernels, comparing it to the peak memory consumption of INT8 and FP16 tensor-core GEMM implementations in full parameter tuning. In practice, QuZO achieves up to a 7.8× memory reduction with an INT8 model compared to the first-order FP16 trainning, as shown in Fig 5.

5 Conclusion

This work has proposed a Quantized Zeroth-Order (QuZO) method for truly qantized training of LLMs without using back propagation. We have identified the challenge of quantized ZO training, and proposed an new quantized ZO gradient to mitigate the bias in low-precision settings. Since QuZO can avoid the error-prone straight-trough estimator, it can achieve better performance than first-order truly quantized training in low-bit settings. The superior performance of QuZO in low-bit (e.g., INT8 and INT4) training has been shown by a variety of fine-tuning experiments on the OPT-1.3B and LLaMA2-7B models. Our QuZO method is intrinsically hardware efficient for fine-tuning LLMs on low-bit resource-constrained hardware.

Limitations

The presented QuZO method can significantly impact practical LLM deployment. We have not yet implemented the real quantized training framework using low-precision kernels during training, as this requires much engineering effort. For instance, adding a minimal hardware block to an LLM inference accelerator can enable resource-efficient fine-tuning, making on-device learning of LLMs accessible and affordable for many downstream users. Additionally, QuZO can greatly reduce the latency and energy cost of fine-tuning due to its capability to directly use an ultra low-bit LLM inference accelerator. This will enable the deployment of LLMs in many resource-constrained scenarios, such as autonomous systems and robots.

References

Ron Banner, Itay Hubara, Elad Hoffer, and Daniel Soudry. 2018. Scalable methods for 8-bit training of neural networks. *Advances in neural information processing systems*, 31.

Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. 2020. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 696–697.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David Cox. 2019. Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization. *Advances in neural information processing systems*, 32.

Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. 2019. Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE.

Michael P Connolly, Nicholas J Higham, and Theo Mary. 2021. Stochastic rounding and its probabilistic backward error analysis. *SIAM Journal on Scientific Computing*, 43(1):A566–A585.

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.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Chen Feng, Shaojie Zhuo, Xiaopeng Zhang, Ramchalam Kinattinkara Ramakrishnan, Zhaocong Yuan, and Andrew Zou Li. 2024. Stepping forward on the last mile. *arXiv preprint arXiv:2411.04036*.

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*.

Natalia Frumkin, Dibakar Gope, and Diana Marculescu. 2023. Jumping through local minima: Quantization in the loss landscape of vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16978–16988.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Saeed Ghadimi and Guanghui Lan. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the first international conference on Human language technology research*.

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*.

Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. 2019. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4350–4359.

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.

Jangwhan Lee, Minsoo Kim, Seungcheol Baek, Seok Joong Hwang, Wonyong Sung, and Jungwook Choi. 2023. Enhancing computation efficiency in large language models through weight and activation quantization. *arXiv preprint arXiv:2311.05161*.

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

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

Shih-yang Liu, Zechun Liu, Xijie Huang, Pingcheng Dong, and Kwang-Ting Cheng. 2023b. Llm-fp4: 4-bit floating-point quantized transformers. *arXiv preprint arXiv:2310.16836*.

Sijia Liu, Pin-Yu Chen, Xiangyi Chen, and Mingyi Hong. 2018. signsgd via zeroth-order oracle. In *International Conference on Learning Representations*.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yong Liu, Zirui Zhu, Chaoyu Gong, Minhao Cheng, Cho-Jui Hsieh, and Yang You. 2024. Sparse mezo: Less parameters for better performance in zeroth-order llm fine-tuning. *arXiv preprint arXiv:2402.15751*.

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. 2023c. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Vladimir Malinovskii, Denis Mazur, Ivan Ilin, Denis Kuznedelev, Konstantin Burlachenko, Kai Yi, Dan Alistarh, and Peter Richtarik. 2024. Pv-tuning: Beyond straight-through estimation for extreme llm compression. *arXiv preprint arXiv:2405.14852*.

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. 2024. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36.

Ben Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, S Agarwal, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 1.

Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S Vetter. 2018. Nvidia tensor core programmability, performance & precision. In *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pages 522–531. IEEE.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. 2021. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*.

Yurii Nesterov and Vladimir Spokoiny. 2017. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

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

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. 2018b. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

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*.

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, 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.

Yifan Yang, Kai Zhen, Ershad Banijamali, Athanasios Mouchtaris, and Zheng Zhang. 2024a. Adazeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 977–995.

Yifan Yang, Jiajun Zhou, Ngai Wong, and Zheng Zhang. 2024b. Loretta: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3161–3176.

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.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. 2019. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*.

Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, Yan Yan, et al. 2024. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*.

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

Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D Lee, Wotao Yin, Mingyi Hong, et al. 2024. Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark. *arXiv preprint arXiv:2402.11592*.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

Jiajun Zhou, Jiajun Wu, Yizhao Gao, Yuhao Ding, Chaofan Tao, Boyu Li, Fengbin Tu, Kwang-Ting Cheng, Hayden Kwok-Hay So, and Ngai Wong. 2023. Dybit: Dynamic bit-precision numbers for efficient quantized neural network inference. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Appendix

A Experiments Setup

We first conduct experiments with RoBERTa-large on sentiment classification and natural language classification tasks. We follow prior works (Malladi et al., 2024) in low data resource settings which can be sampling k examples per class for k = 16 or 512. QuZO is running for 100k steps and the first order fine-tuning for 5 epochs. We also conducted experiments on a smaller set of tasks (Wang et al., 2018a) that includes entailment, span sentiment analysis, and topic classification. These tasks include perceptual analysis (SST-2 and SST-5 (Socher et al., 2013)), Question Classification (TREC (Hovy et al., 2001)),, and natural language reasoning (MNLI, SNLI, and RTE (Bowman et al., 2015; Williams et al., 2017; Rajpurkar et al., 2018)). The metrics we used for the GLUE benchmark are summarized in Table 6.

Table 6: Metrics that we use to evaluate GLUE Benchmark for BERT-based Model.

Task Name	Metric
SST-2	Accuracy
SST-5	Accuracy
MNLI	Matched Acc.
SNLI	Accuracy
TREC	Accuracy
RTE	Accuracy

Subsequently, we selected several SuperGLUE tasks (Wang et al., 2019), encompassing classification (CB, BoolQ, WSC) and multiple-choice (COPA and ReCoRD), alongside two additional question-answering tasks (SQuAD (Rajpurkar et al., 2016) and DROP (Dua et al., 2019)). To intensify the challenge, we operated under the few-shot setting, randomly sampling 1,000 examples for training, 500 for validation, and 1,000 for testing. We followed the prompt settings outlined in Appendix D of (Malladi et al., 2024) to adapt classification tasks into language model tasks. The evaluation metrics used are summarized in Table 7. All experiments were conducted using the AdamW optimizer (Loshchilov and Hutter, 2018).

A.1 Hyperparameters

As observed in some LLM fine-tuning literature, zeroth-order (ZO) optimization typically shows consistent performance improvement with training steps. However, the number of forward passes significantly affects computational costs. To optimize resource usage, we limit the training steps to 10k for the RoBERTa-Large model on the SST-2, SST-5, TREC, MNLI, and SNLI datasets. In Table 8, our method primarily use a batch size of 64 and experiment with different learning rates for RoBERTa-Large fine-tuning (Fig. 4). Since first-order (FO)-based methods use the Adam optimizer, both FO and LLM-QAT (Liu et al., 2023c) experiments utilize smaller batch sizes and larger learning rates compared to ZO tuning. We use the hyperparameters in Table 8 for the RoBERTa-Large model. Note that even though we run all experiments for 5 epochs, further learning steps may help to improve the performance of our proposed methods further.

Table 7: Metrics that we use to evaluate SuperGLUE and generations tasks.

Task Name	Metric
CB	F1
BoolQ	Accuracy
WSC	F1
COPA	Accuracy
ReCoRD	F1
SQuAD	F1
DROP	F1

Table 8: The hyperparameter grids used for RoBERTa-Large experiments.

Experiment	**Hyperparameter**	**Values**
FO	Batch size	$\{8, 16\}$
	Learning rate	$\{1e{-}5, 1e{-}6\}$
LLM-QAT	Batch size	$\{8, 16\}$
	Learning rate	$5e{-}6$
QuZO	Batch size	$\{16, 64\}$
	Learning rate	$\{1e{-}6, 1e{-}7\}$
	ϵ	$1e{-}5$
	Weight decay	$\{0, 0.1\}$

Regarding the LLaMa-2 7B model, we use the hyperparameters in Table 9. We evaluate the model for around 10-12k training steps and directly use the last checkpoint for evaluation. All first-order (FO) quantization training experiments train for 5 epochs and all QuZO experiments use 12K steps.

Modeling and implementation The model and prompt-tuning process follows a structured ap-

Table 9: The hyperparameter grids used for LLaMA-2 experiments.

Experiment	Hyperparameters	Values
QLoRA	Batch size	$[2, 4, 8, 16]$
	Learning rate	$1e-5, 5e-6, 5e-7$
LLM.int8()	Batch size	$[2, 4, 8, 16]$
	Learning rate	$1e-5, 5e-6, 5e-7$
MeZO	Batch size	$[8, 16]$
	Learning rate	$1e-4, 5e-5, 5e-6$
QuZO	Batch size	$[4, 8, 16]$
	Learning rate	$1e-4, 5e-5, 5e-6$

proach tailored for RoBERTa-large, OPT, and LLaMa-2 models across various tasks. For RoBERTa, a masked language model (MLM) fine-tuning paradigm is used, where prompts incorporate [MASK] tokens that the model learns to predict, with specific label word mappings defining classification outputs. Tasks such as sentiment classification (SST-2, SST-5), topic classification (TREC), and natural language inference (MNLI, SNLI, RTE) utilize template-based prompts adapted from prior works (Gao et al., 2020).

For OPT and LLaMa-2, the tuning process follows GPT-3-style prompting (Mann et al., 2020) and encompasses three task categories: classification, multiple-choice, and question answering (QA). Classification tasks rely on cross entropy loss for label prediction, while multiple-choice and QA tasks utilize teacher forcing to train on correct outputs. During inference, classification and multiple-choice predictions are determined using the average log-likelihood per token, whereas QA responses are generated through greedy decoding. Additionally, in-context learning with 32-shot examples is employed to maintain stable results.

For classification tasks, RoBERTa uses linear probing, while OPT and LLaMa employ LM head tuning to refine task-specific representations. This fine-tuning framework ensures consistent evaluation across datasets and models, leveraging structured prompts to enhance adaptability in both low-data and fully supervised settings.

Full Parameter Tuning Performance of QuZO on OPT Models We further evaluate our method on the OPT-1.3B model. The activation functions of OPT models are generally more sensitive to quantization errors compared to the LLaMA model, posing some challenges for LLM quantization. In Table 10, our QuZO method outperform post-training quantization methods such as QLLM and SmoothQuant in 8 out of 11 tasks under the INT W8A8 quantization.

B Quantization Methods

In this section, we present our weight-activation quantization method. Since per-channel activation quantization is incompatible with efficient GEMM kernels, we employ per-tensor static activation quantization as our coarsest-grained quantization method and per-channel weight quantization as our finer-grained quantization scheme. For post-training quantization (PTQ) methods, we adopt the quantization configuration from SmoothQuant and evaluate their W8A8 quantization under our low data resource setting. Additionally, we reproduce LLM-FP4 (Liu et al., 2023b) using their open-source codebases and evaluate the same tasks within their frameworks, noting that it requires significant time for datatype searching. To ensure a fair comparison, we reduce the calibration size to 8.

B.1 Weight-only Quantization

Throughout this work, we focus initially on both weight and activation quantization. This approach can introduce significant quantization errors and lead to accuracy degradation. To address this, we further evaluate weight-only quantization on several tasks, as detailed in Table 11. Our findings indicate that weight-only quantization yields better performance compared to combined weight and activation quantization. There are some related work that only do weight quantization for LLMs (i.e GPTQ (Frantar et al., 2022)). But it converts the quantized weight to FP16 on the fly during inference and lead to speed up.

B.2 Hybrid Datatype Support

Mixed Datatypes Support. Assigning the same low-bit datatype to both weights and activations in QuZO can lead to accuracy degradation due to the limited precision of 4-bit integers compared to floating-point formats, with activation functions being particularly sensitive to quantization errors. While QLoRA introduced the NF4 datatype to mitigate this issue, our QuZO framework takes it a step further by assessing quantization errors (Jung et al., 2019) for hybrid formats at the same precision. This mixed-datatype fine-tuning in quantized ZO training effectively preserves performance even under 4-bit quantization. Existing works (Liu et al.,

Table 10: Performance comparisons for weights and activations quantization on the OPT-1.3B model.

OPT-1.3B Model		Classification							Multiple-Choise		Generation	
Data Precision	Method	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	SQuAD	DROP
INT W8A8	QLLM	82.45	55.59	66.07	**63.00**	63.46	52.35	**56.81**	71.01	59.90	61.49	15.80
	LLM.int8	53.66	53.79	41.07	46.32	42.31	58.46	45.72	**75.00**	70.22	67.14	10.33
	SmoothQuant	75.01	52.34	37.51	48.20	44.23	57.83	53.41	71.03	68.81	69.42	11.22
	QuZO(FT)	**91.38**	**55.61**	**67.85**	62.30	**63.46**	**60.03**	55.91	74.00	**70.81**	**73.88**	**21.82**

Table 11: Weight-only Quantization experiments conducted on LLaMa-2 7B model.

LLaMa-2 7B Model		Classification					Multiple-Choise		Generation	
Data Precision	Method	SST-2	RTE	CB	BoolQ	MultiRC	COPA	ReCoRD	SQuAD	DROP
INT-W4A32	QuZO(FT)	92.43	60.28	60.71	**65.50**	59.60	83.00	79.00	**82.78**	37.31
INT-W8A32	QuZO(FT)	92.77	**62.81**	**71.42**	64.00	60.70	**83.00**	**81.00**	80.93	40.25
FP-W8A32	QuZO(FT)	**93.69**	61.37	66.07	63.72	**60.91**	81.01	79.60	80.93	**37.86**

2023c; Zhou et al., 2023) also incorporate this into their quantization strategy but require customized hardware to support the specific datatype. In our quantization algorithm, we use a set of quantization grids $\mathbf{b} = \{b_1, b_2, \ldots, b_i\}$ and apply the quantization operation $Q_b(w)$ to map a full-precision scalar w to a quantized value as follows:

$$Q_b(\mathbf{w}) = b_i, \quad i = \arg\min_i |w - b_i|.$$

This notation indicates that the parameter w is quantized to the closest quantization grid point b_i. We define the corresponding quantization error as:

$$\mathbb{E}_b(\mathbf{w}) = Q_b(\mathbf{w}) - \mathbf{w}. \tag{16}$$

We use the mean squared error (MSE) as the metric to quantify the quantization loss:

$$\text{MSE} = \mathbb{E}\big[(\mathbf{w} - Q_b(\mathbf{w}))^2\big]. \tag{17}$$

where w are the FP32 value, and $p(w)$ stands for the probability density function. The neural network weights are a random variable $w \sim p_w(w)$. The quantization range is defined between b_{min} and b_{max}. Our framework selects the data type that minimizes the MSE for each layer and executes the searching algorithm only once before fine-tuning. Based on our data-type search algorithm, we found that INT quantization is more suitable for weight quantization, offering better hardware efficiency. On the other hand, FP quantization is primarily chosen for activation quantization to maintain good accuracy. This quantization selection offers a more accurate QuZO fine-tuning process.

Underflow severely impacts low-bit quantization in LLMs (Lee et al., 2023), associated with rounding zero values that further degrade model performance. Therefore, we propose a hybrid datatype search in Section 4.2 during quantized zeroth-order training, using existing data formats, including integers and floating-points, which are widely used in hardware platforms. We evaluate the LLaMA-2 model using the hybrid datatype detailed in Table 12. Through coarse layer-wise datatype selection, QuZO can boost around 1 to 2% average performance across these 11 tasks in both W4A8 and W8A8 quantization.

B.3 Quantized Perturbation

We now explore the ZO gradient quantization, which can accelerate model training without compromising convergence. Using a fully quantized I-BERT (Kim et al., 2021) as an example, we assign low-bit perturbation to update the INT8 model, as shown in Table 13. The accuracy drop is less than 1%, but the memory reduction is around 4-16× for the random perturbation parameters. In the RoBERTa-Large model, we found that 2-bit perturbation performs better, indicating that quantized perturbation does not significantly affect training performance. This is a huge benefit for ZO training since the perturbations are generated and calculated four times for one training step. Current works only focus on sparse parameter perturbations (Liu et al., 2024) for reducing gradient estimation variance in RGE. It introduces the masks and applies them to weight perturbations per step. However, we now consider on hardware-efficient side and use low-precision weight perturbation to do ZO gradient

Table 12: Compared to pure-INT or FP quantized zero-order training, our hybrid datatype (INT and FP) searching algorithm boosts accuracy by 1-2% for most tasks on the LLaMa-2 7B model.

LLaMa-2 7B Model			Classification							Multiple-Choise		Generation		Avg
Method	Datatype	Precision	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	SQuAD	DROP	Performance
QuZO(Ours)	INT	W4A8	89.10	54.87	62.50	66.60	64.42	57.99	60.60	**83.00**	78.20	78.12	31.80	66.10
QuZO(Ours)	INT/FP	W4A8	90.59	59.92	63.71	68.40	64.50	**59.70**	59.30	80.00	78.60	79.89	33.55	67.10
QuZO(Ours)	INT	W8A8	93.00	61.01	64.18	80.00	63.46	52.82	60.01	81.00	79.00	77.71	31.11	67.58
QuZO(Ours)	INT/FP	W8A8	**93.08**	**65.95**	**64.28**	**81.10**	**64.57**	55.17	**60.11**	83.00	**79.60**	**80.74**	**36.58**	**69.47**

Table 13: Evaluate the impact of low-bit perturbation on QuZO training for SST-2 tasks using different models.

Model	Model Precision	Perturbation (#bit)	Performance
I-BERT	INT W8A8	8	92.77
I-BERT	INT W8A8	4	92.48
I-BERT	INT W8A8	2	91.89
RoBERTa-Large	INT W8A8	8	92.48
RoBERTa-Large	INT W8A8	4	91.51
RoBERTa-Large	INT W8A8	2	93.07
LLaMa-2 7B	INT W4A8	8	91.32

Table 14: Number of Perturbations **u** for Different Models at Each Step.

Model	Number of Perturbation
I-BERT	406885467
RoBERTa-Large	407935067

estimation in LLM fine-tuning. We further analyze the memory costs of the perturbation parameters $\mathbf{u} \in \mathbb{R}^d$. At each step, QuZO reuses **u** four times in Algorithm 1. We evaluated the quantized perturbation experiments on the RoBERTa-Large model, and it costs around 1.63 GB of memory to store each **u** during one step. However, quantized perturbation would only cost 110 to 410 MB if we quantize it to 2-bit or 8-bit, respectively, based on our estimated results in the Table 14. Since these results are estimated based on the number of perturbations and storage datatype, real hardware implementation is required to demonstrate the full advantage. We will address this in future work.

Handling outliers. The outliers mainly occur in the activations of transformers and can severely degrade quantization performance if not addressed efficiently (Liu et al., 2023c,a; Lin et al., 2023). To simplify the quantization process without introducing overhead, we propose an outlier detector that can distinguish outliers from normal values. Our outlier detector can automatically select the outlier threshold to determine a suitable ratio α (Outliers/All data), which is normally around 1%. We quantize the normal data using a pre-defined quantization datatype and quantize the outlier data using the same precision FP type. As a signed INT8 quantization example, we designate the binary code 10000000_2 as an outlier label to identify outlier values in the selected tensor array. Consequently, the valid data range becomes $[-127, 127]$, and we utilize an 8-bit floating-point scheme with adaptive biased bits to efficiently quantize these outlier values. It enables efficient quantization of LLMs across various hardware platforms such as CPU and FPGAs using the QuZO method.

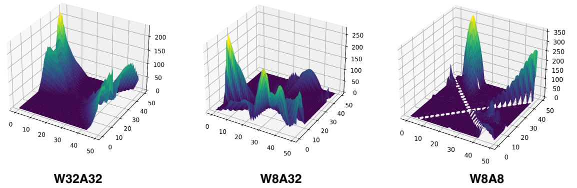


Figure 6: The loss landscape of the RoBERTa-large model under different quantization bits. The notations W and A mean the bits for weights and activation.

Loss Landscape. The effectiveness of ZO fine-tuning for LLMs arises from starting near the optimal loss region. Theoretical analysis in (Malladi et al., 2024) [Lemma 3] links ZO convergence to the low effective rank of Hessian matrix. In quantized training, the Lipschitz smoothness constant L significantly impacts performance (Frumkin et al., 2023). Fig. 6 (See Appendix B) demonstrates the stability of the smoothness of loss function across weight and activation quantization levels, underscoring the effectiveness in low-bit ZO training.

B.4 ZO Gradient Accumulation

Gradient accumulation is a technique for training models where data samples are divided into several batches and calculated sequentially. To fine-tune large models on a single GPU, especially for datasets like DROP that require small batch sizes, we implemented a zeroth-order accumulation method for performing weight updates. Initially, we calculate the gradient without updating the network parameters at each step, accumulating the projected gradient information. After reaching the predefined accumulation steps, the accumulated gradient is used to update the parameters.

We also incorporate prevalent efficiency-enhancing tricks adopted in current zeroth-order optimizers, following the first-order approach to implement our zeroth-order method effectively. This approach allows efficient fine-tuning of large models on a single GPU, leveraging the advantages of gradient accumulation within a QuZO optimization framework.

C Memory Efficiency of QuZO

To demonstrate the hardware efficiency of QuZO, we employ the Cutlass INT8 Kernel to showcase memory efficiency. To fine-tune large models efficiently with limited GPUs, we assess the first-order (FO) method using Fully Sharded Data Parallelism (FSDP) (Zhao et al., 2023) for distributed training. Besides, We believe it can be further reduced if we fully apply the INT engine in each linear and non-linear layer. This could be our next step in the CUDA optimization. Finally, we provide the memory cost of our QuZO method using INT8 CUDA kernels and compare it with the peak memory usage of INT8 and FP16 tensor-core GEMM implementations on full parameter tuning. As the batch size increases from 1 to 32, the memory reduction reaches up to 7.8× when running with an INT8 model compared to FP16 training in Fig. 5. Table 15 provides a comprehensive comparison of memory consumption (in GB) across various tasks when fine-tuning quantized models using QuZO with LoRA $(rank = 8)$. The methods compared include QuZO, LLM.int8(), and QLoRA. Notably, QuZO employs 4-bit perturbations to fine-tune the models, achieving significant memory savings compared to LLM.int8 and QLoRA. For instance, in the OPT1.3B-int4 model, QuZO reduces memory usage by approximately 2.8× on SST-2 (1.72 GB vs. 4.76 GB in QLoRA) and by 5.47× on SQuAD (3.74 GB vs. 20.48 GB in QLoRA). Similarly, for the OPT1.3B-int8 model, QuZO achieves a memory reduction of 1.4× on MultiRC (12.61 GB vs. 23.97 GB in INT8 FO fine tuning).

Table 15: Memory Consumption (GB) Across Models and Methods for Five Tasks. This table compares the memory requirements of different methods (e.g., LLM.int8, QuZO, and QLoRA) across various tasks using two models: OPT1.3B and LLaMa-2 7B. The QuZO method demonstrates significantly lower memory consumption across all models, while LLM.int8() encounters Out of Memory (OOM) issues in some cases.

Model	Methods	SST-2	MultiRC	ReCoRD	SQuAD	DROP
8-bit OPT 1.3B	LLM.int8()	9.01	23.97	6.76	22.09	31.29
	QuZO	3.43	12.61	4.82	7.50	16.42
4-bit OPT 1.3B	QLoRA	4.76	18.15	4.42	20.48	27.23
	QuZO	1.72	6.30	2.41	3.74	11.70
8-bit LLaMa-2 7B	LLM.int8()	31.47	OOM	19.06	OOM	OOM
	QuZO	9.94	25.11	13.04	16.69	31.66

In the 8-bit LLaMa-2 7B model, while LLM.int8 encounters Out-of-Memory (OOM) errors on several tasks, QuZO successfully completes fine-tuning with substantial memory efficiency, using just 9.94 GB on SST-2 compared to 31.47 GB for LLM.int8—a reduction of 3.2×. These results highlight the ability of QuZO to fine-tune quantized models effectively with minimal memory overhead, leveraging 4-bit perturbations for substantial efficiency gains while maintaining compatibility with LoRA architectures. This positions QuZO as a practical choice for resource-constrained fine-tuning in large-scale NLP tasks.