TT-ZO: Tensor-Train Enhanced Zeroth-order Fine-tuning of Large Language Models

Yifan Yang, Kai Zhen, Athanasios Mouchtaris, Siegfried Kunzmann , Zheng Zhang

Abstract

Fine-tuning large language models (LLMs) has achieved remarkable performance in recent years across various natural language processing tasks, while the backpropagation graph consumes a substantial proportion of memory. To address this issue, the recently proposed Memory-efficient Zeroth-order (MeZO) methods attempt to fine-tune LLMs using only forward passes. Nonetheless, a significant performance drop and increased risk of divergence prevent widespread adoption of this method. In this paper, we propose the Tensor-Train enhanced zeroth-order (TT-ZO) framework, specifically designed to improve the performance and convergence of the ZO method. To further accelerate the forward pass of the Tensor-Train adaptation, we introduce a new contraction method for the tensorized layer. To address the divergence problem in large-scale model fine-tuning using ZO methods, we propose an adaptive query number schedule that ensures convergence with detailed theoretical analysis. Additionally, we present substantial experimental results on Roberta-large and Llama-2-7B models, demonstrating the effectiveness of our proposed ZO training framework in terms of accuracy, memory cost, and convergence speed. Further experimental results suggest that the proposed adaptive query schedule could be successfully applied to other ZO fine-tuning methods to improve their convergence and performance.

1 Introduction

Fine-tuning large language models (LLMs) has demonstrated outstanding performance in addressing numerous natural language processing tasks, such as natural language understanding (Kenton and Toutanova, 2019; Liu et al., 2019), question-answering (Rajpurkar et al., 2018), and summarization (Zhang et al., 2024). However, as the size of LLMs increases, the training process consumes increasingly more GPU memory, where a main part of memory consumption is the backpropagation graph. To reduce the memory and computational costs during training, various parameter-efficient fine-tuning methods have been proposed (Hu et al., 2021; Houlsby et al., 2019). Despite these PEFT methods significantly reducing training memory by updating only a portion of the parameters, the overall memory cost remains high due to the persistent use of a backpropagation graph. This cost is difficult to mitigate with PEFT methods, as backpropagation remains essential.

To further reduce the memory overhead, (Malladi et al., 2023) proposed the Memory-efficient Zeroth-order (MeZO) method for the fine-tuning of LLMs, which shows over 10$\times$ memory reduction in the case of full model fine-tuning. Difference from the first-order (FO) method (e.g. SGD (Amari, 1993), AdamW (Loshchilov and Hutter, 2018)) that calculates the gradient through the backpropagation, the MeZO method estimates the gradient by the difference of loss values obtained by two forward passes. However, an increasing risk of divergence has been observed for the MEZO methods (Gautam et al., 2024) and a large gap exists between FO and MEZO methods.

To address the divergence issue, various first-order (FO) optimization techniques have been adapted for zeroth-order (ZO) scenarios, such as the ZO-SVRG method (Gautam et al., 2024) and the ZO-Admm method (Jiang et al., 2024). However, these approaches fail to accommodate the specific needs of ZO methods, and they introduce considerable memory overhead due to the optimizer state without significantly enhancing ZO fine-tuning performance. Given the dimensionality-related nature of ZO convergence rates, (Liu et al., 2024) propose a sparse-ZO method, which generates a sparse mask for parameters with small values to reduce the number of trainable parameters needed for ZO optimization. Nevertheless, the number of parameters remains substantially higher than that required by most parameter-efficient fine-tuning (PEFT) methods, and the selection of pruned parameters is based solely on experimental outcomes.

In this paper, we explore improving the convergence and performance of ZO fine-tuning methods by reducing the number of trainable parameters using PEFT techniques. While Malladi et al. have attempted to employ LoRA (Hu et al., 2021) and prefix-tuning (Hu et al., 2021) methods in the MEZO framework, the performance improvements are still limited compared to full model fine-tuning. To further enhance performance, we propose the Tensor-Train Enhanced Zeroth-order (TT-ZO) framework to improve both performance and convergence stability, which includes fast-forward tensorized adapters. Our contributions can be summarized as follows:

- We introduce the Tensor-Train Enhanced Zeroth-order (TT-ZO) framework, designed to enhance ZO estimation accuracy and reduce memory costs by significantly decreasing the number of training parameters.
- A new contraction method is introduced to accelerate the forward pass of tensorized adapters by over 20%.
- We develop an adaptive query number schedule, which sub-linear increases the query numbers to address the persistent divergence problem in ZO fine-tuning.
- We provide a comprehensive theoretical analysis of the convergence of our proposed method, examining the convergence dynamics of ZO fine-tuning in relation to the loss landscape, problem dimensions, and query schedule.
- Experimental results demonstrate that our proposed method achieves superior accuracy across various models and tasks while also enhancing computational efficiency.

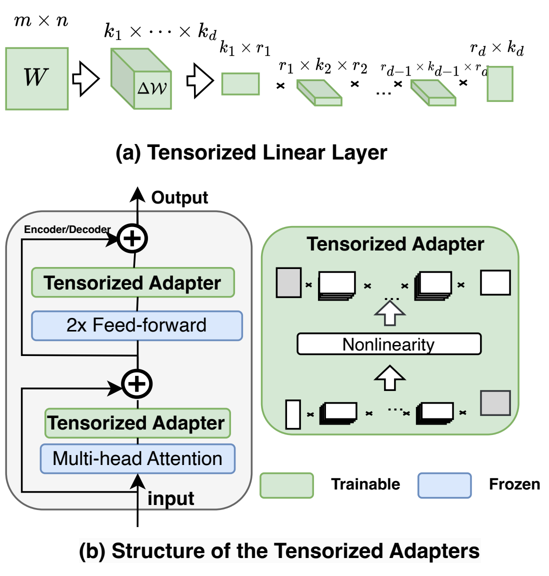

Figure 1: Illustration for tensorized linear layer and tensorized adapters

2 Background

2.1 Parameter-Efficient Fine-tuning

In recent years, various works related to parameter-efficient fine-tuning (PEFT) methods have been proposed. Beyond the most widely used methods like Adapters (Houlsby et al., 2019) and LoRA (Hu et al., 2021), there are also methods exploring ultra-low trainable parameter solutions (Zaken et al., 2022; Li and Liang, 2021; Liu et al., 2022). For instance, the BitFit method (Zaken et al., 2022) fine-tunes large language models (LLMs) by optimizing only the bias terms. The IA3 method (Liu et al., 2022) reduces trainable parameters by rescaling inner activations with learned vectors.

However, these methods struggle to perform well when scaled up to large-scale models. To address these challenges, Yang et al. propose the LoRETTA method (Yang et al., 2024), which includes high-performance tensorized adapters capable of handling large-scale models. Unfortunately, the original tensorized adapters are slow during the forward pass, due to a contraction process involving a sequence of small tensor factors.

2.2 Tensorized Adapters

The tensorized adapters method, also studied in (Yang et al., 2024), is built upon tensorized linear layers. We illustrate the general idea of tensorized linear layers in Fig. 1(a). The weight matrix in tensorized linear layers is represented in the Tensor-Train (TT) format. To obtain the TT format from a weight matrix $\boldsymbol{W}$, we first reshape $\boldsymbol{W}$ into a d-way tensor $\mathcal{W}$. Then, we could represent $\mathcal{W}$ as a sequence of tensor factors $[\mathcal{G}_1, \cdots, \mathcal{G}_d]$ (Oseledets, 2011), where $\mathcal{G}i \in \mathbf{R}^{r \times k_i \times r}$ and r is a constant tensor rank. During the forward pass, the sequence of tensor factors is contracted back into a weight matrix as:

$$\boldsymbol{W} = \text{Reshape}(\prod_{i=1}^{d} \mathcal{G}_i[r_{i-1}, k_i, r_i]). \qquad (1)$$

It is important to note that during the fine-tuning process, we only store and update the sequence of tensor factors in the tensorized layer.

The structure of tensorized adapters is shown in Fig. 1 (b). Each tensorized adapter contains two tensorized layers with a non-linear layer in between. For each encoder/decoder block, the tensorized adapters are attached after the attention and feed-forward layer. In this paper, the pre-trained weight is frozen, and only the parameters within adapters are updated. The tensorized adapters significantly reduce trainable parameters compared to traditional adapters with linear layers. For example, in a model with a hidden size of 768, traditional adapters contain approximately $2 \times 768 \times 64 \approx 98K$ parameters with a bottleneck size of 64. In contrast, tensorized adapters contain only $\sum_{i=1}^{6}(5^2 \cdot 8) = 1.2K$ parameters, given a tensor shape k_i of $[8, 8, 8, 8, 8, 8]$ and a rank of 5. This demonstrates that tensorized adapters are an excellent tool for achieving a high compression ratio within the adapter structure.

Compared to other high PEFT methods with more trainable parameters, tensor adapters are better suited for zero-order (ZO) fine-tuning. The high compression ratio of tensorized adapters leads to a significant reduction in trainable parameters, while the inclusion of a non-linear layer preserves their representation ability. The main challenge with tensorized adapters is the need to perform sequence contraction during the forward pass along a sequence of tensor factors. Since the ZO method requires one more forward pass than backpropagation, traditional adapters become slow due to the contraction issue. In the following sections, we will introduce a new fast-forward tensorized adapter that addresses this issue by parallel contracting the small tensor cores.

3 Methods

Even though previous work has explored the possibility of using PEFT adapters during ZO fine-tuning, few studies have designed specific PEFT frameworks that consider the key factors influencing ZO fine-tuning accuracy. In this section, we first introduce some basic knowledge of the ZO gradient estimator. Then, we present our TT-ZO method, a powerful framework designed to improve the performance of ZO LLM fine-tuning with two main components: 1) a fast-forward, low-parameter PEFT adapter that reduces the number of trainable parameters and improves the speed of the forward pass, and 2) an adaptive query number schedule to address the common issue of divergence in many ZO fine-tuning cases. Finally, we provide a theoretical analysis of the convergence rate of our proposed ZO algorithm, demonstrating the improved convergence rate theoretically.

3.1 Zeroth-order Estimation

Traditional ZO estimation has been widely studied in both convex and non-convex optimization setups. There exist ZO optimization algorithms such as ZO-SGD (Ghadimi and Lan, 2013; Malladi et al., 2023), ZO-Adam (Chen et al., 2019), and ZO-SVRG (Liu et al., 2018). Consider a supervised dataset $\mathcal{D} = (x_i, y_i)$ with a size of D and mini-batch $\mathcal{B}$ with the size of B. The Randomized Zeroth-order Gradient Estimation (RGE) is given as follows.

Definition 1 (Randomized Zeroth-order Gradient Estimation (RGE)). Given a model with parameters $\boldsymbol{w} \in \mathbf{R}^d$ and the corresponding loss function $\ell(\boldsymbol{w}; \mathcal{B})$, the RGE over a batch of data $\mathcal{B}$ is given by:

$$\hat{\nabla} f(\boldsymbol{w}) = \sum_{q=1}^{Q} \frac{\ell(\boldsymbol{w} + \epsilon u_q; \mathcal{B}) - \ell(\boldsymbol{w} - \epsilon u_q; \mathcal{B})}{2\epsilon} u_q \tag{2}$$

where Q is the maximum query number, $u_q \sim \mathcal{N}(0, \boldsymbol{I}_d)$ is the vector-wise random perturbation for each query q, and ϵ is a scaling factor for the perturbation.

In the case of ZO LLMs fine-tuning, we use the RGE to estimate the gradient with two forward passes, instead of the backpropagation used in the FO fine-tuning. Thus, there is no existing backpropagation graph and the speed-up of the forward passes is more importance under the ZO case.

Algorithm 1 TT-ZO

```
Input: Parameters $\boldsymbol{w}$, loss function $\ell(\cdot)$, scaling factor $\epsilon$, maximum query $M$, learning rate $\eta$.
1:  Calculate query growing rate
    $\alpha \leftarrow \log(M)/\log(K)$
2:  for $k = 1, \cdots, K$ do
3:      Each epoch start: $q \leftarrow (\frac{k}{\lceil D/B \rceil})^{\alpha}$
4:      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \boldsymbol{z}, \boldsymbol{z}\, \mathcal{N}(0,1)$
5:      $\ell_{+} \leftarrow \ell(\boldsymbol{\theta}, B)$
6:      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - 2\epsilon \boldsymbol{z}, \boldsymbol{z}\, \mathcal{N}(0,1)$
7:      $\ell_{-} \leftarrow \ell(\boldsymbol{\theta}, B)$
8:      $\hat{\nabla}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \frac{1}{q} \sum_{i=1}^{q} \left[ \frac{\ell_{+} - \ell_{-}}{\mu} \mathbf{u}_i \right]$
9:      Reset the random seed for generating $\boldsymbol{z}$
10: end for
```

3.2 TT-ZO Framework

Previous ZO fine-tuning methods, such as MEZO, typically estimate the gradient for a large number of trainable parameters simultaneously using ZO estimation. This approach results in high variance due to the dimension-related nature of ZO estimation accuracy. Although techniques like LoRA and prefix tuning can reduce the number of trainable parameters, no work has explored parameter-efficient fine-tuning with a ZO optimizer. Additionally, as mentioned above, we have observed an increased risk of divergence when using a ZO optimizer during fine-tuning. To address these issues, we propose our TT-ZO framework to improve performance and solve the instability problem of the vanilla MEZO method. Our framework includes the following components:

Parameter-Efficient and Fast-Forward Tensorized Adapters The Parameter-efficient issue has been widely studied in the FO cases, where people often freeze the pre-trained model parameters and fine-tune the LLMs by adding trainable adapters along with the frozen pretrain weights. Since the ZO estimation accuracy is dimension-dependent, the parameter efficiency issue is becoming more important in the ZO fine-tuning case. Thus, we consider injecting the ultra-low parameter tensorized adapters in our TT-ZO framework to reduce the number of trainable parameters while retaining the performance.

As we have mentioned, ZO fine-tuning mainly relies on gradient estimation with two forward passes at each step. Thus, the speed of the forward pass is a crucial factor for the overall speed of ZO fine-tuning. Instead of using the sequential contraction method during the forward pass as in previous work, we propose new parallel contraction and resource-saving methods. These methods divide the sequence of tensor factors into several groups to enable parallel processing and avoid the presence of high-dimensional tensors. Taking a bipartite case as an example, the contraction process in Eq. (1) is replaced by:

$$\boldsymbol{W} = \mathrm{R}(\prod_{i=1}^{\lfloor \frac{d}{2} \rfloor} \mathcal{G}_i \prod_{j=\lfloor \frac{d}{2} \rfloor + 1}^{d} \mathcal{G}_j),$$

where $\mathcal{G}_i$ represents the i-th tensor factor and $\mathrm{R}(\cdot)$ represents the reshape operation. For larger models, we can tripartite or quadripartite the list of tensor factors to further accelerate the inference speed for the tensorized method. Corresponding CUDA optimizations can further improve training efficiency in the future.

Adapterive Query Adjustment for ZO estimation. As we mentioned, the training process for existing ZO methods is unstable, where the case-by-case divergence problem often happens in large-size models. In previous work, the influence of query numbers for the estimation accuracy has been discussed among optimization fields (Chen et al., 2019; Jiang et al., 2024), the fixed number of queries largely harms the training efficiency of ZO LLMs fine-tuning, as the fine-tuning time for LLMs are often long and naively increasing the number of perturbation timely scale up the training time. In this paper, we consider a simple but effective sublinear increasing query number adjustment schedule, where the number of queries at each time step is calculated as $Q_k := (\frac{k}{\lceil D/B \rceil})^\alpha$ with a fixed increasing rate α. This adjustment schedule successfully solved all divergence problems we observed and performed faster than the traditional way to solve the divergence problem by increasing the batch size.

The corresponding optimization algorithm used in the TT-ZO framework is shown in Alg. 1. At the beginning of the fine-tuning process, we fix the maximum query number and calculate the increasing rate α. Then, we utilize this rate to calculate the query number at the beginning of each epoch. Different from the MEZO algorithm, we obtain the gradient used for the model update by taking the average over multiple query results. Note that we fix the query number to be 1 when fine-tuning medium-size models like Roberta-large since the noise of ZO estimation is relatively low when the number of trainable parameters is small. Later, we will show that the choice of a sublinear increasing query number benefits the convergence of the problem when the model size is large, both theoretically and experimentally.

3.3 Theoretical Analysis

In this subsection, we give the theoretical analysis for the TT-ZO framework. Our theoretical analysis highlights why the tensorized adapter can perform better than the LoRA method tried in the previous try. Furthermore, we show a sublinear increasing query number, even with a small increasing rate, can significantly help to improve the convergence rate of the ZO fine-tuning algorithm. Different from the theoretical analysis in the MEZO paper, which focus on the effective rank, we focus on the dimension of the optimized models (number of trainable parameters). As the trainable parameters with PEFT adapters are much smaller than the model size, the theoretical analysis based on the real dimension of the optimization problem helps us to explore the behavior of different PEFT methods.

To align our analysis with the Neural network training case, we conduct the analysis with a non-convex optimization setup. For convenience, we consider an optimization problem with the loss function $\ell(\cdot)$ of our fine-tuning problem as the objective function and study the convergent behavior regarding the steps t, which is simplified from t_k in Alg. 1. First, we list the following assumptions for our analysis:

A1: the loss function $\ell(\cdot)$ have L-lipschitz continuous gradients, where $L \in (0, \infty)$.

A2: At each step t, the graidient $\nabla \ell_k$ has an upper bound of R, which gives $\|\nabla \ell_k\| \le \delta, \forall k \in [1, \cdots, K]$.

Then, we offer the global convergence rate for our TT-ZO algorithm:

Theorem 1. *Under A1 an A2, the convergence rate of the TT-ZO algorithm can be bounded as:*

$$\frac{1}{K}\sum_{k=1}^{K}\mathbf{E}[\nabla \ell_k(\boldsymbol{w}_k)^2] \le \frac{R + \mu^2 L + C(d,\mu)\sum_k \frac{1}{Q_k}}{K\eta} + \frac{L\eta^2\delta + \mu^2 LC(d,\mu)}{2\eta}$$

Proof. Our analysis starts from a lemma that shows the relationship between the degree of the gradient and the gradient noise led by both mini-batch sampling and ZO estimation. Then, we give the bound for the norm of gradient noise to obtain the global convergence rate. Detail of the proof can be found in Appendix A □

According to Theorem 1, we can observe that the bound is related to the query schedule. In our TT-ZO framework, we employ the sublinear adaptive query schedule. For example, we set the query number schedule as $Q_k = \alpha k^{\frac{\beta}{2}}$, gives:

$$\sum_k \frac{1}{Q_k} = \sum_k \sqrt{\frac{1}{k^\beta}} = \frac{2}{2-\beta} k^{1-\frac{\beta}{2}}, \quad (3)$$

Table 1: Comparative analysis of various ZO fine-tuning methods on the Roberta-large models.

Tasks	**SST-2**	**sst-5**	**QNLI**	**MNLI**	**SNLI**	**mr**	**RTE**
FT	91.9	47.5	63.4	70	77.5	88.2	66.4
Zero-Shot	79	35.5	50.9	48.8	50.2	80.2	51.4
LP	76	40.3	57.6	56.5	66	86.6	59.4
BS=16							
MEZO							
MEZO(LoRA)	84.2	44.8	60.3	58.5	65.6	85.7	52.7
TT-ZO	**91.4**	**48.3**	61.3	57.8	69.1	**87**	64.26
BS=64							
MEZO	90.5	45.5	60.5	58.7	68.5	85	64
MEZO(LoRA)	91.3	43		64	**69.7**	**87.4**	**64.9**
TT-ZO	**91.4**	**49.2**	**60.7**	**68.1**	68.7	86.4	63.5

which guarantees a sublinear convergence rate. In contrast, if we use a constant, especially a small constant query number like 1, the upper bound of the convergence rate can never be a small value under the LLMs case we studied. Here, the convergence rate is related to two key factors, the Lipschitz smoothness factor of the loss function L and the dimension of the model d. In the other part of our work, we mainly focus on reducing the model parameters without losing the representation ability of the models.

4 Experiemnts

We conduct comprehensive experiments for the performance of our proposed TT-ZO framework for LLMs with different scales. Specifically, we present the results on both Roberta-large (**?**) and Llama-2-7B (Touvron et al., 2023) models over multiple natural language understanding (NLU) (Socher et al., 2013; Williams et al., 2017; Rajpurkar et al., 2018) and natural language generation (NLG) (Dua et al., 2019; Rajpurkar et al., 2016) tasks. The detail about the experiments setup can be referred to Appendix B.

In this section, we first show our methods outperform a wide variety of baseline ZO fine-tuning methods like MEZO, ZO-LoRA (Malladi et al., 2023), ZO-adam (Jiang et al., 2024), and sparse-ZO (Liu et al., 2024). Then, we give experiments showing the advantage of TT-ZO method can reduce the training memory cost, training FLOPs the hardware efficiency advantages of the proposed TT-ZO method with respect to memory compared with employing other PEFT methods like LoRA. Finally, we show that our adaptive query number schedule can also be broadcasted to ZO fine-tuning with other PEFT methods. All experiments are performed with NVIDIA Tesla A100-40GB GPUs.

4.1 Bert-based Models

We initially conducted experiments on the Roberta-large models on tasks including single-sentence tasks like SST-2, SST-5, natural language inference tasks including QNLI, MNLI, SNLI, RTE, and sentiment analysis dataset like Movie Reviews (MR). We summarize the test results in Tbale. 1 and compare the downstream task performance comparison between first-order AdamW method (Loshchilov and Hutter, 2018), zero-shot (Brown et al., 2020), linear probing (Kumar et al., 2021) and the proposed TT-ZO method. We run the experiment under 16-shot setup with 16 data samples in each classes of the datasets and collect the best test accuracy erevy 500 steps with 1000 datasamples for all experiments. Following conclusions have been reached.

TT-ZO shows stronger performance than other ZO fine-tuning methods. According to our observation in Tbale. 1, TT-ZO method ourperfrom other ZO fine-tunin, like MEZO, ZO-LoRA. Compared with ZO-LoRA method, which also envolve the PEFT adapters, TT-ZO outperforms over 6 out of 7 tests in 16 batch size cases and 5 out of 7 in the 64 batch size case. This is leads by the fact that LoRA has more trainable parameters than the tensozied adapter, which

Table 2: Comparative analysis of various ZO fine-tuning methods on the llama-2-7B.

llama-2-7B	RTE	CB	BoolQ	MultiRC	COPA	ReCoRD	SQuAD	DROP
Zero-Shot	49.5		65.1	55.8	59.7			
ICL	54.5		67.4	58.7	84.4			
LP								
FT	61.73	66.1	84.6	45.4	86	81.1	90.71	51.38
LoRA	85.56	67.86	84.8	85	81	79.4	90.56	51.48
MEZO								
MEZO(LoRA)	59	75	69.8			71.2		
TT-ZO	66.2	73	64.4					

is more difficult to converge according to the dimension-related converge rate proved in Section 3.3.

TT-ZO demonstrates improved convergence. Compared with LoRA method, LoRETTA method can converge well under the 16 batch size case. Note that it's reasonable to see the 16 batch size case outperform the 64 batch size method if the fine-tuning process converges well since we are considering a 16-shot training setup. This also indicate that ZO estimation is not working well on LoRA method due to the large training batch size problem.

4.2 Llama-3 Models

In the previous section, we demonstrated encouraging results using the tensorized adapter method to enhance the performance of ZO fine-tuning. In this section, we evaluate the performance of the TT-ZO framework on the large-scale Llama-2-7B model. Unlike the experiments on the Roberta-large models, we incorporate the adaptive query schedule method proposed in our TT-ZO framework to address the commonly observed divergence issue in ZO fine-tuning fields. To increase the challenge of our experiments, we employ a low-data resource setup with datasets from SuperGLUE (Wang et al., 2019) and generation tasks such as SQuAD (Rajpurkar et al., 2016) and DROP (Dua et al., 2019). Our experimental setup follows the prompted-based fine-tuning strategy used in the MEZO paper (Malladi et al., 2023). The results are presented in Table 2 and Figure **??**. Based on these experimental outcomes, the following conclusions are drawn:

TT-ZO Method Demonstrates Superior Performance Over Traditional ZO Fine-Tuning. The TT-ZO framework exhibits outstanding performance across a range of tasks, surpassing all ZO baselines such as MEZO and ZO-LoRA methods. Additionally, TT-ZO significantly outperforms traditional inference-only methods like ICL and Zero-shot. Remarkably, the TT-ZO method approaches the results of first-order (FO) methods, which consume $12\times$ more memory.

TT-ZO Method Effectively Addresses Divergence Issues in ZO Fine-Tuning. According to Figure **??**, a serious divergence issue is evident among various ZO fine-tuning methods, such as MEZO, ZO-LoRA, and ZO-LoRETTA, particularly in the fine-tuning for SST-2 and xxx tasks, where they fail to decrease training loss before achieving convergence. In contrast, the adaptive query schedule within the TT-ZO framework successfully resolves this divergence issue, enhancing training outcomes. Subsequent analyses demonstrate that by sublinearly increasing the query number, TT-ZO converges faster than when addressing the convergence problem with increased batch sizes.

TT-ZO Method Exhibits Robust Performance Across Diverse Tasks. Further testing of the TT-ZO method on generative tasks like SQuAD and DROP reveals that TT-ZO consistently achieves high performance. Moreover, compared to the FO method, TT-ZO supports a $16\times$ larger batch size using the same number of GPUs, showcasing its efficiency and scalability.

4.3 Hardware Efficiency

In this section, we study the hardware efficiency of the TT-ZO method on Llama-2-7B from three perspectives: training flops, average step time, and peak memory consumption. Previously, a normal way to solve the divergence problem and improve the training performance is to increase the batch size. Here, we show that our adaptive query number schedule method is a good alternative to large batch-size training from the hardware efficiency perspective. We compare TT-ZO with three baseline methods (MEZO, ZO-LoRA, FO tuning), and two variations of TT-ZO methods (TT-ZO with fixed query of 1 (TT-ZO$_{fix_query}$), TT-ZO with sequential factors contraction (TT-ZO$_{seq}$). We show the comparison result in Table **??** and Fig. **??**. From the observation in these results, we reach the following conclusions.

TT-ZO reduce reaches a 12× reduction compared with FT method. The TT-ZO method only require 16GB GPU memory for the fine-tuning of SST-2 dataset on Llama-2-7B model. COmparable, the FT method need xxGB memory.

The adaptive query schedule used in TT-ZO method performs faster and better can save up to 50% memory. To relax the divergence problem encountered in the large scale ZO fine-tuning, the precious ZO algorithm only consider reducing the learning rate and increasing the batch size. Here we shows that the TT-ZO framework we propose perfrom faster than even increasing the batch size. Also, by remaining a small batch size of 16, we achieve a memory reduction of 50% compared a batch size of 64 case to relax the divergence problem. At the same time, our method achieve much higher fine-tuning performance compared with the increasing batch size and reducing learning rate, even with over 4× less training steps.

The parallel contraction method helps to improve the training speed. After using the parallel contraction method, the average step time is reduce by 20% compared with the traditional tensorized adapter, which helps to further improve the training efficiency of the proposed TT-ZO method.

The TT-ZO method has further potential in reducing the training time by observing the train g Flops. We observe that the training flops of the propose TT-ZO method is much lower than the MEZO method. Evne though the training time is similar at this point, further CUDA optimization may largely improving the training efficiency since the parallel contraction of lists of small tensor factors has not been fully optimized.

4.4 Other PEFT Methods

Our adaptive query schedule can also be applied on other PEFT methods for the stable ZO fine-tuning. We can observe from Fig. **??**, the adaptive query schedule can welly solve the divergen problem of LoRA method when the batch size is relative small, which helps to improve both the memory efficiency and training performance.

4.5 Query and Learning Rate Schedule

5 Conclusion

In this paper, we propose an enhanced zeroth-order fine-tuning framework with tensor-train decomposition, named TT-ZO, which outperforms other ZO fine-tuning methods across various tasks and models. To design specific PEFT adapters suitable for ZO optimizers, we introduce fast-forward tensor adapters and demonstrate that freezing the layer norm is crucial for convergence. Theoretical analysis confirms that our proposed methods are guaranteed to converge faster than those using other PEFT adapters. Extensive experimental results indicate that our method achieves better outcomes with faster convergence. Furthermore, we explore the potential of using our adaptive query schedule, showing that it can be widely adapted to other ZO fine-tuning methods to address common divergence issues.

References

Shun-ichi Amari. 1993. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196.

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.

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*.

Tanmay Gautam, Youngsuk Park, Hao Zhou, Parameswaran Raman, and Wooseok Ha. 2024. Variance-reduced zeroth-order methods for fine-tuning language models. *arXiv preprint arXiv:2404.08080*.

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

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

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

Shuoran Jiang, Qingcai Chen, Youcheng Pan, Yang Xiang, Yukang Lin, Xiangping Wu, Chuanyi Liu, and Xiaobao Song. 2024. Zo-adamu optimizer: Adapting perturbation by the momentum and uncertainty in zeroth-order optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18363–18371.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Ananya Kumar, Aditi Raghunathan, Robbie Matthew Jones, Tengyu Ma, and Percy Liang. 2021. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

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

Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. 2018. Zeroth-order stochastic variance reduction for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31.

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*.

Sharon L Lohr. 2021. *Sampling: design and analysis*. Chapman and Hall/CRC.

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

Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. 2023. Fine-tuning language models with just forward passes. *arXiv preprint arXiv:2305.17333*.

Ivan V Oseledets. 2011. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.

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*.

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.

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

Yifan Yang, Jiajun Zhou, Ngai Wong, and Zheng Zhang. 2024. Loretta: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. *arXiv preprint arXiv:2402.11417*.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2024. Benchmarking large language models for news summarization. *Transactions of the Association for Computational Linguistics*, 12:39–57.

A Proof of Theorem 1

We start from the update rule of the ZO-SGD optimizer, which gives $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta\nabla\hat{\ell}_k(\boldsymbol{w}_k)$. By using Taylor's theorem on the loss, we have:

$$\ell(\boldsymbol{w}_{k+1}) = \ell(\boldsymbol{w}_k - \eta\nabla\hat{\ell}_k(\boldsymbol{w}_k)) \tag{4}$$

$$= \ell(\boldsymbol{w}_k) - \eta\nabla\hat{\ell}_k(\boldsymbol{w}_k)^\top\nabla\ell(\boldsymbol{w}_k) + \frac{\eta^2}{2}\nabla\hat{\ell}_k(\boldsymbol{w}_k)^\top\nabla\ell(\boldsymbol{w}_k)^2\nabla\hat{\ell}_k(\boldsymbol{w}_k) \tag{5}$$

Taking expectation on both side, gives:

$$\mathbf{E}[\ell(\boldsymbol{w}_{k+1})] = \mathbf{E}[\ell(\boldsymbol{w}_k)] - \eta\mathbf{E}[\nabla\hat{\ell}_k(\boldsymbol{w}_k)^\top\nabla\ell(\boldsymbol{w}_k)] + \frac{\eta^2}{2}\mathbf{E}[\nabla\hat{\ell}_k(\boldsymbol{w}_k)^\top\nabla\ell(\boldsymbol{w}_k)^2\nabla\hat{\ell}_k(\boldsymbol{w}_k)] \tag{6}$$

$$\overset{\text{(a)}}{\leq} \mathbf{E}[\ell(\boldsymbol{w}_k)] - \eta\mathbf{E}[\nabla\ell_k(\boldsymbol{w}_k)^2] + \frac{\eta^2 L}{2}\mathbf{E}[\nabla\hat{\ell}_k(\boldsymbol{w}_k)^2], \tag{7}$$

where (a) is given by the Assumption A1.

Rearrange and we obtain:

$$\eta\mathbf{E}[\nabla\ell_k(\boldsymbol{w}_k)^2] \leq \mathbf{E}[\ell(\boldsymbol{w}_k)] - \mathbf{E}[\ell(\boldsymbol{w}_{k+1})] + \frac{\eta^2 L}{2}\mathbf{E}[\nabla\hat{\ell}_k(\boldsymbol{w}_k)^2] \tag{8}$$

Taking summation over steps $k = 1, \cdots, K$ gives:

$$\sum_{k=1}^{K}\eta\mathbf{E}[\nabla\ell_k(\boldsymbol{w}_k)^2] \leq \mathbf{E}[\ell(\boldsymbol{w}_0) - \ell(\boldsymbol{w}_K)] + \sum_{k=1}^{K}\frac{\eta^2 L}{2}\mathbf{E}[\nabla\hat{\ell}_k(\boldsymbol{w}_k)^2] \tag{9}$$

$$\overset{\text{(a)}}{\leq} \mathbf{E}[\ell_0 - \ell^*] + \mu^2 L + \sum_{k=1}^{K}\frac{\eta^2 L}{2}\mathbf{E}[\nabla\hat{\ell}_k(\boldsymbol{w}_k)^2] \tag{10}$$

$$\overset{\text{(b)}}{\leq} R + \mu^2 L + \sum_{k=1}^{K}\frac{\eta^2 L}{2}\mathbf{E}[\nabla\hat{\ell}_k(\boldsymbol{w}_k)^2], \tag{11}$$

where (a) is using the Lemma 1 in (Liu et al., 2018) that $\ell_\mu(\boldsymbol{w}_0) - \ell_\mu(\boldsymbol{w}_T) \leq \ell_\mu(\boldsymbol{w}_0) - \ell(\boldsymbol{w}_0) + \ell^* - \ell_\mu^* \leq (\ell_\mu(\boldsymbol{w}_0) - \ell^*) + \mu^2 L$ and (b) is given by setting $R := \ell(\boldsymbol{w}_1) - \ell^*$. Now, the key to the bound comes from the last term in the right of the inequation.

To bound the last term, we first represent the noise gradient $\nabla\hat{\ell}_k(\boldsymbol{w}_k)^2$ as a combination of the true gradient and the gradient noise, which gives:

$$\nabla\hat{\ell}_k(\boldsymbol{w}_k) := \nabla\ell_k(\boldsymbol{w}_k) + h_k \tag{12}$$

Then, we begin to bound the last term $\mathbf{E}[\nabla\hat{\ell}_k(\boldsymbol{w}_k)^2]$. Recall from the gradient estimation rule in eq. (2) that

$$\nabla\hat{\ell}_k(\boldsymbol{w}_k) = \frac{1}{B}\sum_{b_i\in\mathcal{B}_k}\hat{g}_k(\boldsymbol{w}_k; b_i) \quad \hat{g}_k(\boldsymbol{w}_k; b_i) = \frac{1}{Q_k}\sum_{j=1}^{Q}{}_k\hat{g}_k(\boldsymbol{w}_k; b_i, u_{i,j}), \tag{13}$$

where there are two sources of gradient noise: a) the gradient noise leads by the mini-batch sampling. b) the gradient noise leads by the presence of ZO gradient estimation. Here we first bound the gradient noise leads by the mini-batch sampling with a fact given in the stochastic gradient descent theory, as the mini-batch estimation is unbiased and the ZO estimation is biased.

Considering the gradient noise h_k in optimizing the target objective with mini-batch sampling:

$$h_k := \|\nabla \ell_k(\boldsymbol{w}_k) - \hat{g}_k(\boldsymbol{w}_k; b_i)\| \tag{14}$$

. For convenience, we consider a general case that the mini-batch $\mathcal{B}_k$ is formed by uniform sampling without replacement and follow the i.i.d. fashion, according to (Lohr, 2021)[Section 2.8, Page 48], it holds:

$$\mathbf{E}[\|h_k\|] = \frac{N-B}{NB}\Lambda^2, \tag{15}$$

where Λ^2 is the sample variance of the gradient $\hat{g}_k(\boldsymbol{w}_k; b_i)$ leads by the mini-batch sampling, which is defined as:

$$\begin{aligned}
\Lambda^2 &= \frac{1}{B-1}\sum_{i=1}^{B} \|\hat{g}_k(\boldsymbol{w}_k; b_i) - \nabla \ell_k(\boldsymbol{w}_k)\| && (16)\\
&= \frac{1}{B-1}\sum_{i=1}^{B} \|\nabla \ell_k^{\mu} + e_k^{i,j} - \nabla \ell_k\|, && (17)
\end{aligned}$$

where e_k is defined as the gradient noise leads by the ZO estimation that $e_k := \nabla \ell^i - \nabla \ell_k^{\mu}$ Finally, we need to bound the variance Λ^2, which is related to the ZO gradient estimation. Taking expectation with respect to the i.i.d. random perturbation vector $\boldsymbol{u}$, we have:

$$\begin{aligned}
\mathbf{E}_{\boldsymbol{u}}[\Lambda^2] &= \mathbf{E}_{\boldsymbol{u}}[\frac{1}{B-1}\sum_{i=1}^{B} \|\nabla \ell_k^{\mu} + e_k^{i} - \nabla \ell_k\|] && (18)\\
&\leq \mathbf{E}_{\boldsymbol{u}}[\frac{1}{B-1}\sum_{i=1}^{B} \|e^i\|] + \frac{1}{B-1}\sum_{i=1}^{B} \|\nabla \ell_k(\boldsymbol{w}) - \nabla \ell_k^{\mu}\| && (19)\\
&\overset{(a)}{\leq} \frac{1}{(B-1)Q^2}\sum_i\sum_j \mathbf{E}_{\boldsymbol{u}}[\|e^{i,j}\|] + \frac{B\mu^2 L}{2(B-1)} && (20)\\
&\overset{(b)}{=} \frac{1}{(B-1)Q}\sum_i \mathbf{E}_{\boldsymbol{u}}[\|e^{i,1}\|]] + \frac{B\mu^2 L}{2(B-1)} && (21)
\end{aligned}$$

where (a) is by using the (Liu et al., 2018)[lemma 1] again and (b) is given by the facts that $\mathbf{E}_{\boldsymbol{u}}[e^{i,j}] = \mathbf{E}_{\boldsymbol{u}}[e^{i,1}]$ as $u_{i,j}$ is i.i.d. samples.

Finally, we need to bound the first term $\mathbf{E}_{\boldsymbol{u}}[e^{i,1}]]$, which gives:

$$\begin{aligned}
\mathbf{E}_{\boldsymbol{u}}[\|e^{i,1}\|]] &= \mathbf{E}_{\boldsymbol{u}}[\|\nabla \ell^{i,1} - \nabla \ell_k^{\mu}\|]] && (22)\\
&\leq \mathbf{E}_{\boldsymbol{u}}[\|\nabla \ell^{i,1} - \nabla \ell_{\mu}^{i,1}\|] + \mathbf{E}_{\boldsymbol{u}}[\|\nabla \ell_{\mu} - \nabla \ell_{\mu}^{i,1}\|] && (23)\\
&\overset{(a)}{\leq} 2d\|\nabla \ell^{i,1}\| + \frac{\mu^2 L^2 d^2}{2} + \mathbf{E}_{\boldsymbol{u}}[\|\nabla \ell_{\mu}\|] + \mathbf{E}_{\boldsymbol{u}}[\|\nabla \ell_{\mu}^{i,1}\|] \overset{(b)}{\leq} 2d\delta^2 + \frac{\mu^2 L^2 d^2}{2} + 2\delta^2 && (24)\\
&\overset{(c)}{\leq} 2d\delta^2 + \frac{\mu^2 L^2 d^2}{2} + 2\delta^2, && (25)
\end{aligned}$$

where (a) follows a similar idea of the proof in (Ghadimi and Lan, 2013)[eq. (3.21)] and (b) is given by using the bound of the gradient in Assumption A2 for the last three terms.

Taking eq. (22) back into eq. (18), then combining eq. (9), eq. (12) and eq. (15), taking the expectation over all randomness and average over the maximum steps K, we obtain:

$$\frac{1}{K}\sum_{k=1}^{K}\eta\mathbf{E}[\nabla\ell_k(\boldsymbol{w}_k)^2] \leq \frac{R}{K} + \frac{\mu^2 L}{K} + \frac{1}{K}\sum_{k=1}^{K}\frac{\eta^2 L}{2}\mathbf{E}[\nabla\hat{\ell}_k(\boldsymbol{w}_k)^2] \tag{26}$$

$$\frac{R}{K} + \frac{\mu^2 L}{K} + \frac{1}{K}\sum_{k=1}^{K}\frac{\eta^2 L}{2}\mathbf{E}[\|\nabla\ell_k(\boldsymbol{w}_k)\| + \|h_k\|] \tag{27}$$

$$= \frac{R}{K} + \frac{\mu^2 L}{K} + \frac{\eta^2 L\delta}{2} + \frac{1}{K}\sum_{k=1}^{K}\frac{\eta^2 L}{2}\frac{N-B}{NB}\Lambda^2 \tag{28}$$

$$\leq \frac{R}{K} + \frac{\mu^2 L}{K} + \frac{\eta^2 L\delta}{2} + \frac{1}{K}\sum_{k=1}^{K}\frac{\eta^2 L}{2}\frac{N-B}{NB}\left(\frac{1}{(B-1)Q}\sum_i \mathbf{E}_{\boldsymbol{u}}[\|e^{i,1}\|] + \frac{B\mu^2 L}{2(B-1)}\right) \tag{29}$$

$$\leq \frac{R}{K} + \frac{\mu^2 L}{K} + \frac{\eta^2 L\delta}{2} + \frac{1}{K}\sum_{k=1}^{K}\frac{\eta^2 L}{2}\frac{N-B}{NB}\left(\frac{\sum_i(2d\delta^2 + \frac{\mu^2 L^2 d^2}{2} + 2\delta)}{(B-1)Q} + \frac{B\mu^2 L}{2(B-1)}\right) \tag{30}$$

$$= \frac{R + \mu^2 L + C(d,\mu)\sum_k \frac{1}{Q_k}}{K} + \frac{L\eta^2\delta + \mu^2 LC(d,\mu)}{2} \tag{31}$$

Divide both side with η, we finish the proof as:

$$\frac{1}{K}\sum_{k=1}^{K}\mathbf{E}[\nabla\ell_k(\boldsymbol{w}_k)^2] \leq \frac{R + \mu^2 L + C(d,\mu)\sum_k \frac{1}{Q_k}}{K\eta} + \frac{L\eta^2\delta + \mu^2 LC(d,\mu)}{2\eta} \tag{32}$$

B Detail of Experiment Setup