Prior-Driven Zeroth-Order Optimization for Scalable and Memory-Efficient LLM Fine-Tuning

Anonymous EMNLP submission

Abstract

Fine-tuning large language models (LLMs) has demonstrated exceptional performance across a variety of natural language processing (NLP) tasks. However, the increasing scale of these models imposes significant memory overhead during backpropagation. While zeroth-order (ZO) optimization mitigates this issue by estimating gradients through forward passes and Gaussian sampling, its random sampling strategy introduces variance that scales linearly with the number of parameters, leading to slow convergence and suboptimal performance. We propose a novel gradient estimation framework that utilizes the computation of a guiding vector, which is derived from Gaussian sampling to direct perturbations for approximating gradients. By incorporating this prior knowledge into the perturbation process, our method significantly accelerates convergence compared to traditional ZO approaches. Additionally, we investigate whether a greedy strategy can yield similar enhancements in gradient estimation, providing further insights into the optimization process. Theoretical analysis indicates that the proposed gradient estimator achieves a more substantial alignment with the true gradient direction, thereby improving optimization efficiency. Comprehensive experiments conducted across LLMs of varying scales and architectures demonstrate that our method could integrates seamlessly into diverse optimization frameworks, delivering faster convergence and substantial performance improvements compared to existing methods.

1 Introduction

The emergence of fine-tuning techniques for large language models (LLMs) has revolutionized natural language processing (NLP), enabling state-of-the-art performance in tasks such as text generation and question answering (Brown et al., 2020; Achiam et al., 2023). However, as LLMs are scaled up, the computational and memory demands during full fine-tuning increase exponentially. A significant bottleneck arises during backpropagation (Rumelhart et al., 1986), which requires the storage of intermediate activations and gradients, leading to substantial memory overhead. In recent years, memory-efficient training strategies, such as parameter-efficient fine-tuning (PEFT) (Hu et al., 2022; Houlsby et al., 2019; Li and Liang, 2021), have emerged as promising alternatives by selectively updating only a subset of model parameters. Despite these advancements, memory efficiency remains limited: experiments on OPT-13B (Zhang et al., 2022) indicate that full fine-tuning and PEFT still consume 12× and 6× more GPU memory than inference, respectively (Malladi et al., 2023).

To address these challenges, researchers have investigated alternative optimization paradigms that reduce memory requirements while preserving model performance. Zeroth-order (ZO) optimization has emerged as a promising candidate, substituting backpropagation with gradient estimation through Gaussian sampling and forward passes (Malladi et al., 2023). ZO methods significantly alleviate computational burdens by eliminating the necessity to store intermediate activations, which are a primary source of memory overhead. Recent advancements focus on accelerating convergence and minimizing gradient variance, with innovations such as sparse perturbation strategies enhancing computational efficiency (Liu et al., 2024; Guo et al., 2024). Hybrid frameworks further integrate ZO principles with established techniques: coupling ZO with the Adam optimizer (Guo et al., 2024) enhances stability, while Hessian-aware gradient estimation (Zhao et al., 2024) improves accuracy by incorporating second-order curvature information. Concurrent efforts combine ZO with Parameter-Efficient Fine-Tuning (PEFT) frameworks, such as low-rank adaptations (Hu et al., 2022) and tensorized adapters (Yang et al., 2024), to minimize the number of trainable parameters,

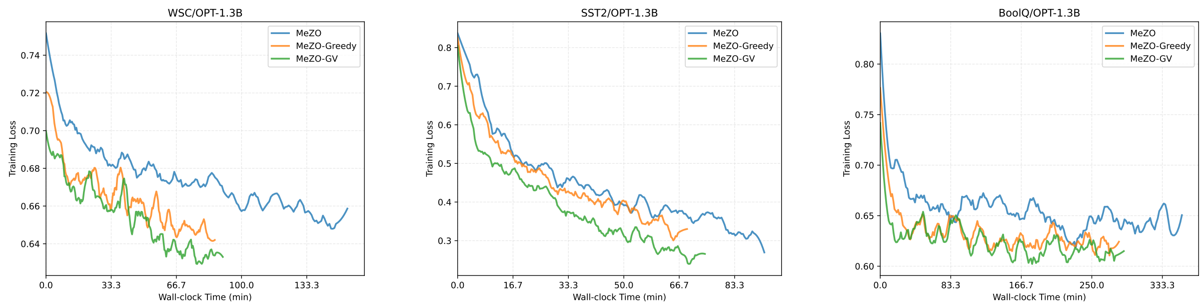


Figure 1: The training loss curves for the WSC, SST-2, and BoolQ tasks are evaluated using the OPT-1.3B model. Our proposed methods (MeZO-Greedy and MeZO-GV) are compatible with MeZO. For full fine-tuning, a learning rate of 2e-7 is employed. All experiments are conducted with a consistent batch size of 16 to ensure uniformity across evaluations.

demonstrating progress toward scalable and flexible optimization.

A fundamental challenge in zeroth-order (ZO) optimization arises from the inherent limitations of conventional gradient estimators, which typically rely on random Gaussian perturbations, such as those used in MeZO. Our work explicitly acknowledges that achieving perfect unbiasedness in the estimation of the ZO gradient is theoretically infeasible in practice due to the presence of the finite difference parameter ϵ and the necessity of approximating expectations over random perturbations. Motivated by this inherent limitation, we propose to intentionally deviate from the standard Gaussian perturbation scheme by incorporating prior-informed perturbations.

To this end, we introduce the Guiding Vector-Augmented Zeroth-Order (GV-ZO) method, which utilizes prior knowledge to direct the perturbation process. Our approach iteratively estimates a guiding vector through adaptive Gaussian sampling, thereby dynamically aligning the perturbation direction with the expected true gradient. Additionally, we propose a prior-informed greedy perturbation strategy, which further illustrates the effectiveness of integrating prior knowledge in gradient estimation through empirical evaluation.

Theoretically, we demonstrate that our guiding vector-augmented and greedy-enhanced zeroth-order optimization frameworks achieve significantly stronger directional alignment with the true gradient compared to conventional ZO methods (see Section 5). This improved alignment ensures that each optimization step contributes more effectively to the convergence dynamics (see Appendix C.1). Empirical experiments conducted on diverse LLM architectures and scales show that our method not only converges faster (see Figure 1) but also yields substantial performance improvements over existing approaches. Furthermore, on the OPT-13B model, GV-based approaches consistently achieve state-of-the-art performance across all 11 benchmark tasks, outperforming traditional zeroth-order optimization methods. When compared to gradient-based baselines, GV-based methods exhibit superior results on 9 out of 11 tasks, demonstrating a strong balance between efficiency and accuracy. These results highlight the robustness and adaptability of our framework. Notably, our method employs a plug-and-play design, allowing for seamless integration into a wide range of optimization pipelines. This makes it a versatile and practical solution for optimizing modern large language models (LLMs), particularly in resource-constrained environments.

2 Background

2.1 Simultaneous Perturbation Stochastic Approximation (SPSA)

The Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 1992) is a zeroth-order optimization method used to approximate the gradient of scalar-valued functions $f(\boldsymbol{x})$ where $\boldsymbol{x} \in \mathbb{R}^d$. The SPSA gradient estimate employs finite differences along random Gaussian directions:

$$\hat{\nabla} f(\boldsymbol{x}) = \frac{1}{q} \sum_{i=1}^{q} \left(\frac{f(\boldsymbol{x} + \mu \boldsymbol{u}_i) - f(\boldsymbol{x} - \mu \boldsymbol{u}_i)}{2\mu} \right) \boldsymbol{u}_i, \tag{1}$$

where q represents the number of function evaluations, $\mu > 0$ denotes the perturbation step size, and $\boldsymbol{u}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ are random direction vectors.

As $\mu \to 0$, the finite difference converges to the directional derivative $f'(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{u}^\top \nabla f(\boldsymbol{x})$. This results in an unbiased gradient estimator:

$$\mathbb{E}_{\boldsymbol{u}}[f'(\boldsymbol{x}, \boldsymbol{u})\boldsymbol{u}] = \mathbb{E}_{\boldsymbol{u}}[\boldsymbol{u}\boldsymbol{u}^\top \nabla f(\boldsymbol{x})] = \nabla f(\boldsymbol{x}), \tag{2}$$

making SPSA particularly effective for high-dimensional optimization tasks, such as fine-tuning LLMs.

2.2 Memory-Efficient ZO-SGD (MeZO)

Given a labeled dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{D}|}$, minibatch $\mathcal{B} \subset \mathcal{D}$, and a loss function $\mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, the SPSA gradient estimate is expressed as follows:

$$\hat{\nabla}\mathcal{L}(\boldsymbol{\theta}; \mathcal{B}) = \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \boldsymbol{z}; \mathcal{B}) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \boldsymbol{z}; \mathcal{B})}{2\epsilon} \boldsymbol{z}, \tag{3}$$

where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ represents a random perturbation vector, and $\epsilon > 0$ denotes the perturbation scale. The estimator $\hat{\nabla}\mathcal{L}(\boldsymbol{\theta}; \mathcal{B}) \approx \boldsymbol{z}\boldsymbol{z}^\top \nabla \mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$ requires only two forward passes, facilitating memory-efficient optimization. This serves as the foundation for Zeroth-Order Stochastic Gradient Descent (ZO-SGD):

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \hat{\nabla}\mathcal{L}(\boldsymbol{\theta}; \mathcal{B}_t), \tag{4}$$

where $\mathcal{B}_t$ represents the t-th minibatch and η denotes the learning rate, ZO-SGD mitigates the memory overhead associated with backpropagation by substituting exact gradients with SPSA estimates. For more related work, see Appendix B.

3 Our Proposed Method

The proposed method is a plug-and-play strategy designed for seamless integration into any zeroth-order optimization algorithm that employs stochastic perturbation for gradient estimation. The guiding vector mechanism and the greedy perturbation strategy are intentionally architecture-agnostic, ensuring broad compatibility with various optimization frameworks. This inherent flexibility allows the proposed method to be easily adapted to diverse optimization techniques without necessitating significant modifications to the underlying process. To rigorously demonstrate the effectiveness and generality of our approach, we have integrated the proposed mechanisms into two prominent zeroth-order optimization algorithms—MeZO (Malladi et al., 2023) and SubZero (Yu et al., 2024)—and conducted comprehensive experiments to evaluate their performance across a range of models and tasks.

3.1 Memory-efficient ZO with Guiding Vector

In this work, we propose Memory-Efficient Zeroth-Order Optimization with Guiding Vectors (**MeZO-GV**), an advanced zeroth-order optimization algorithm designed to efficiently optimize high-dimensional parameters $\theta \in \mathbb{R}^d$ in scenarios where gradient computations are either infeasible or computationally expensive. The algorithm builds upon the traditional MeZO framework by introducing a guiding vector $\boldsymbol{v}$ that directs parameter updates toward more promising regions of the loss landscape. This guiding vector is computed using a perturbation-based exploration strategy, which significantly enhances convergence speed and optimization performance compared to standard zeroth-order methods.

The MeZO-GV algorithm iteratively updates the model parameters θ over a fixed step budget T. At each iteration t, MeZO-GV begins by sampling a minibatch $\mathcal{B}_t$ from the dataset $\mathcal{D}$ and generating a random seed s to ensure in-place operation. The guiding vector $\boldsymbol{v}$ is derived from a set of M perturbations $\{\boldsymbol{z}_i\}_{i=1}^{M}$, where each $\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is a random perturbation vector generated using a unique seed $s_i = \text{Hash}(s \oplus i)$. The perturbations are evaluated on the loss function $\mathcal{L}$, and the top αM perturbations with the lowest losses are selected as the elite group $\mathcal{O}_{\text{top}}$, while the remaining form the non-elite group $\mathcal{O}_{\text{bottom}}$. The guiding vector $\boldsymbol{v}$ is computed as:

$$\boldsymbol{v} = \frac{1}{|\mathcal{O}_{\text{top}}|} \sum_{\boldsymbol{z}_i \in \mathcal{O}_{\text{top}}} \boldsymbol{z}_i - \frac{1}{|\mathcal{O}_{\text{bottom}}|} \sum_{\boldsymbol{z}_i \in \mathcal{O}_{\text{bottom}}} \boldsymbol{z}_i, \tag{5}$$

Using the guiding vector $\boldsymbol{v}$, MeZO-GV estimates the directional gradient $\hat{\nabla}\mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$ via:

$$\hat{\nabla}\mathcal{L}(\boldsymbol{\theta}; \mathcal{B}) = \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \boldsymbol{v}; \mathcal{B}) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \boldsymbol{v}; \mathcal{B})}{2\epsilon} \boldsymbol{v}, \tag{6}$$

where $\epsilon > 0$ is the perturbation scale, this estimator approximates the gradient as $\hat{\nabla}\mathcal{L}(\boldsymbol{\theta}; \mathcal{B}) \approx \boldsymbol{v}\boldsymbol{v}^\top \nabla \mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$. This approach requires only two forward passes and eliminates the need for backpropagation, thereby facilitating memory-efficient optimization. The parameters $\boldsymbol{\theta}$ are updated according to Equation 4. By leveraging the guiding vector $\boldsymbol{v}$, MeZO-GV allows the algorithm to concentrate on the most promising directions for parameter updates, resulting in faster convergence and improved optimization performance. The complete algorithmic implementation is provided in Appendix E.1.

3.2 Memory-efficient ZO with Greedy Perturbation

In addition to the guiding vector mechanism, we propose another Memory-efficient ZO with Greedy Perturbation (**MeZO-Greedy**) strategy as a complementary optimization component to further enhance the performance of the optimization process. MeZO-Greedy functions as an independent mechanism that actively explores the most promising update directions at each iteration. Specifically, the algorithm generates a set of M candidate perturbations $\{\boldsymbol{z}_i\}_{i=1}^{M}$, where each $\boldsymbol{z}_i$ is sampled from a predefined distribution. The greedy selection process then identifies the optimal perturbation $\boldsymbol{z}^*$ that minimizes the loss function in the vicinity of the current parameters:

$$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z}_i} \mathcal{L}(\boldsymbol{\theta} + \epsilon \boldsymbol{z}_i; \mathcal{B}), \tag{7}$$

where ϵ controls the exploration radius, and $\mathcal{B}$ represents the current mini-batch of data, the selected perturbation $\boldsymbol{z}^*$ encapsulates the most favorable direction for parameter updates based on immediate feedback from the loss landscape, effectively capturing the local geometry of the optimization surface.

Building upon this selected direction, we calculate an independent gradient estimate using a symmetric difference approximation:

$$\hat{\nabla}^* \mathcal{L}(\boldsymbol{\theta}; \mathcal{B}) = \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \boldsymbol{z}^*; \mathcal{B}) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \boldsymbol{z}^*; \mathcal{B})}{2\epsilon} \boldsymbol{z}^*, \tag{8}$$

Then the parameters $\boldsymbol{\theta}$ are updated using the Equation 4. The complete algorithmic implementation is presented in Appendix E.2.

4 Experiments and Analysis

LLM fine-tuning tasks and models For all experiments, we consider the SuperGLUE (Wang et al., 2019) dataset collection, which includes CB (De Marneffe et al., 2019), COPA (Roemmele et al., 2011), MultiRC (Khashabi et al., 2018), RTE (Bar-Haim et al., 2014), WiC (Pilehvar and Camacho-Collados, 2019), WSC (Levesque, 2011), BoolQ (Clark et al., 2019), and ReCoRD (Zhang et al., 2018). Additionally, we incorporated SST-2 (Socher et al., 2013) and two question-answering (QA) datasets: SQuAD (Rajpurkar et al., 2016) and DROP (Dua et al., 2019). We also conduct experiments on two representative language models of varying sizes. For OPT (Zhang et al., 2022), we test the OPT-1.3B, OPT-13B, and OPT-30B models, while for Llama2 (Touvron et al., 2023), we evaluate the Llama2-7B-hf and Llama2-13B-hf models. For specific details and the experimental setup, please refer to the Appendix A.

We evaluate zeroth-order (ZO) large language model (LLM) fine-tuning using two sets of metrics: accuracy and efficiency. Accuracy measures the fine-tuned model's test data performance on specific tasks. Efficiency encompasses various measurements, including memory efficiency (e.g., peak memory usage or GPU cost) and convergence speed.

4.1 Medium-sized Language Models

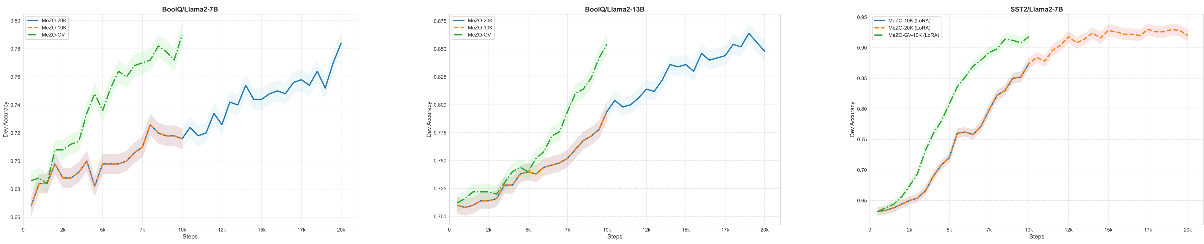
As shown in Table 1, the experimental results demonstrate that GV-based methods, particularly MeZO-GV, consistently outperform both vanilla MeZO and baseline approaches across a wide range of tasks. This highlights that our proposed method achieves significant performance improvements. By leveraging guiding vectors, MeZO-GV enhances fine-tuning efficiency, achieving significant performance gains in classification tasks (e.g., +3.8% on SST-2), multiple-choice tasks (e.g., +5.0% on COPA), and generation tasks (e.g., +3.2% on SQuAD). Notably, MeZO-GV excels in complex scenarios, such as WSC (+3.9% improvement) and MultiRC (+5.3% improvement), where vanilla MeZO and baseline methods exhibit limited effectiveness. Additionally, the proposed method demonstrates significantly accelerated convergence rates, as illustrated in Appendix C.1. For instance, on SST-2 and WSC, MeZO-GV achieves performance comparable to vanilla MeZO at 20,000 steps in just 6,000 and 1,000 steps, respectively. These results highlight MeZO-GV's ability to stabilize the optimization process while effectively adapting to diverse task requirements, establishing it as a robust and memory-efficient fine-tuning framework.

4.2 Large Language Models

With the promising results from OPT-1.3B, we scale the model to larger sizes and architectures to further validate the proposed methods. As shown in Table 2, the experimental results on OPT-13B demonstrate that GV-based methods, such as MeZO-GV and SubZero-GV, consistently outperform their non-GV counterparts and baseline ap-

Table 1: Comparison of average task performance across different methods on OPT-1.3B over three rounds. Results are reported for zero-shot, in-context learning (ICL), and MeZO-based methods, including variants with guiding vectors (GV), LoRA, and prefix tuning. The best performance for each task is highlighted in **bold**.

Task Type	— classification —							— multiple choice —		— generation —	
Task	SST2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	SQuAD	DROP
Zero-shot	53.6	53.1	39.3	44.9	43.3	53.5	45.4	73.0	70.5	27.2	11.2
ICL	80.0	53.4	44.6	59.4	46.2	50.3	46.3	69.0	71.0	58.7	20.5
MeZO(FT)	89.2	57.4	**71.4**	62.5	56.7	57.2	53.3	73.0	70.9	72.0	21.9
MeZO-GV(FT)	93.0	60.6	69.6	64.4	60.6	58.0	58.6	**78.0**	72.0	75.2	24.1
MeZO(LoRA)	90.8	61.7	71.2	63.4	58.7	60.2	57.0	74.0	71.5	77.5	23.1
MeZO-GV(LoRA)	**93.5**	62.8	70.5	**64.8**	**62.5**	**60.7**	60.6	76.0	72.4	78.7	24.4
MeZO(Prefix)	90.1	65.7	69.6	63.0	60.6	56.0	59.1	71.0	70.4	76.0	23.2
MeZO-GV(Prefix)	92.1	**66.8**	70.9	64.5	60.8	58.2	**62.7**	74.0	**72.7**	**78.8**	**24.8**



(a) (b) (c)

Figure 2: Validation Accuracy on SST2 and BoolQ Tasks for Llama2-7B and Llama2-13B. All experiments are conducted with a batch size of 16. For LoRA-based methods, the learning rate is set to 1e-4, while for full-parameter methods, the learning rate is set to 5e-7.

proaches across a wide range of tasks. In classification tasks, SubZero-GV(FT) achieves **94.7%** accuracy on SST-2, surpassing MeZO(FT) by **2.7%**, whileMeanwhile, SubZero-GV(Prefix) attains **85.7%** accuracy on CB, outperforming ZO-AdaMU(Prefix) by **13.4%**. SubZero-GV(Prefix) achieves **76.2%** accuracy on RTE, marking a **5.4%** improvement over MeZO(Prefix), and scores **65.1%** on MultiRC, leading all compared methods. In generation tasks, SubZero-GV (LoRA) achieves **85.3%** on SQuAD, outperforming MeZO (LoRA) by **1.5%**, while MeZO-GV(LoRA) achieves **32.7%** on DROP, surpassing MeZO (LoRA) by **1.3%**. In multiple-choice tasks, GV-based methods consistently demonstrate advantages: MeZO-GV (Prefix) achieves **90.0%** accuracy on COPA, outperforming MeZO (Prefix) by **3.0%**. Compared to zeroth-order optimization methods, GV-based approaches exhibit superior performance across all 11 tasks. Additionally, when compared to gradient-based methods, GV-based methods excel in 9 out of 11 tasks.

To further validate the effectiveness of the proposed method, we extend our approach to the Llama2-7B model, with the experimental results presented in Table 3. The results demonstrate that our GV-based methods consistently outperform non-GV variants across multiple tasks while also achieving significant efficiency improvements. Specifically, GV-based methods achieve superior performance with only 10,000 training steps, surpassing the results of other methods that are trained for 20,000 steps. GV-based methods exhibit strong performance across various tasks. For instance, MeZO-GV-10k achieves **90.4%** accuracy on SST-2, outperforming both MeZO-10k (**85.3%**) and MeZO-20k (**88.7%**) with half the training steps. Similarly, MeZO-GV-10k (LoRA) achieves **94.3%** accuracy on SST-2, surpassing MeZO-10k (LoRA) (**87.7%**) and MeZO-20k (LoRA) (**93.7%**). On more challenging tasks such as WSC and WIC, GV-based methods demonstrate consistent improvements, achieving **62.5%** and **62.3%** accuracy, respectively, outperforming non-GV methods with fewer training steps. Additionally, we conduct experiments on larger models, including Llama2-13B and OPT-30B, with detailed results provided in Appendix C.1. In Figure 2, we present the curves of training steps versus validation accuracy, which further illustrate the effectiveness of GV-based

Table 2: Average task performance of various methods across three rounds on OPT-13B. Results are reported for zero-shot, in-context learning (ICL), ZO-AdaMU (extends zeroth-order optimization to the Adam algorithm), HiZOO (Hessian matrix-based gradient estimation in ZO optimization), SubZero (decomposes parameter mapping into low-dimensional subspaces), MeZO, and their variants that incorporate guiding vectors (GV), LoRA, and prefix tuning. Fine-tuning using the Adam is also included. The best performance for each task among the zeroth-order optimization methods is highlighted in **bold**.

Task Type	—— classification ——							—— multiple choice ——		—— generation ——	
Task	SST2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	SQuAD	DROP
Zero-shot	58.8	59.6	46.4	59.0	38.5	55.0	46.9	80.0	81.2	46.2	14.6
ICL	87.0	62.1	57.1	66.9	39.4	50.5	53.1	87.0	82.5	75.9	29.6
ZO-AdaMU (2×)	92.1	72.9	67.9	73.0	61.5	60.7	63.0	89.0	83.0	82.4	32.0
ZO-AdaMU (LoRA)	88.0	72.0	71.6	72.6	60.1	56.4	58.9	88.0	83.2	76.8	32.4
ZO-AdaMU (Prefix)	88.0	61.8	72.3	74.9	56.5	58.2	61.9	86.0	82.8	85.2	30.4
HiZOO	92.1	69.3	69.4	67.3	63.5	59.4	61.3	88.0	81.4	81.9	25.0
HiZOO(LoRA)	90.6	67.5	69.6	70.5	63.5	60.2	60.2	87.0	81.9	83.8	25.1
HiZOO(Prefix)	92.0	71.8	69.6	73.9	60.6	60.0	64.8	87.0	81.2	83.2	25.3
MeZO(FT)	91.4	66.1	67.9	67.6	63.5	61.1	60.1	88.0	81.7	84.7	30.9
SubZero(FT)	92.1	74.0	73.2	75.3	65.4	60.8	61.0	88.0	82.3	84.5	32.0
MeZO-GV(FT)	93.9	73.5	71.6	72.5	65.4	61.4	62.5	89.0	82.9	84.9	31.7
SubZero-GV(FT)	**94.7**	74.8	73.9	76.8	64.4	62.7	63.2	89.0	83.1	84.9	31.3
MeZO(LoRA)	89.6	67.9	66.1	73.8	64.4	59.7	61.5	84.0	81.2	83.8	31.4
SubZero(LoRA)	93.8	75.5	71.4	76.1	65.4	60.3	60.3	89.0	81.9	83.7	31.3
MeZO-GV(LoRA)	91.6	72.6	72.8	75.6	**66.3**	60.9	61.9	89.0	82.9	84.9	32.7
SubZero-GV(LoRA)	94.0	75.8	73.8	**77.6**	65.4	63.9	64.1	**90.0**	**83.8**	**85.3**	32.4
MeZO(Prefix)	90.7	70.8	69.6	73.1	60.6	59.9	63.7	87.0	81.4	84.2	28.9
SubZero(Prefix)	91.7	73.6	80.3	76.3	62.1	61.1	63.5	88.0	82.0	83.7	32.0
MeZO-GV(Prefix)	92.4	74.8	73.2	76.6	63.5	61.8	64.4	**90.0**	82.7	84.3	30.9
SubZero-GV(Prefix)	93.1	**76.2**	**85.7**	77.1	64.4	**64.1**	**65.1**	89.0	82.5	85.1	**32.9**
FT	92.0	70.8	83.9	77.1	63.5	70.1	71.1	79.0	74.1	84.9	31.3

Table 3: Task Performance Comparison for Different Methods on Llama2-7B.

Task	SST2	RTE	BoolQ	WSC	WIC
MeZO-10k	85.3	58.1	72.1	60.8	57.8
MeZO-20k	88.7	62.1	80.1	62.1	60.8
MeZO-GV-10k	90.4	64.3	81.3	62.5	62.3
MeZO-10k(LoRA)	87.7	60.6	76.9	58.9	56.3
MeZO-20k(LoRA)	93.7	63.3	79.5	62.5	57.5
MeZO-GV-10k(LoRA)	94.3	65.7	80.7	61.5	61.4

Table 4: Task Performance Comparison of Greedy Strategy for Different Methods on Llama2-7B and OPT-13B

Model	**Task**	**WiC**	**RTE**	**BoolQ**
Llama2-7B	MeZO	60.8	62.1	80.1
	MeZO-Greedy	63.0	63.6	81.9
	MeZO (LoRA)	57.5	63.3	79.5
	MeZO-Greedy (LoRA)	61.9	65.7	80.8
OPT-13B	MeZO	61.1	66.1	67.6
	MeZO-Greedy	61.9	72.2	72.6
	MeZO (LoRA)	60.8	74.0	75.3
	MeZO-Greedy (LoRA)	62.7	75.8	75.9

methods. The curves demonstrate that GV-based methods achieve comparable validation accuracy with significantly fewer training steps compared to non-GV methods, reinforcing their efficiency and performance advantages. These results validate the scalability and robustness of GV-based methods across different model sizes, highlighting their potential for efficient fine-tuning in resource-constrained environments.

4.3 MeZO with Greedy Strategy

In Table 4, we present the test accuracy of various optimization methods, including MeZO, MeZO-Greedy, SubZero, and SubZero-Greedy, applied to the Llama2-7B and OPT-13B models across multiple datasets (e.g., WIC, RTE, BoolQ). The results demonstrate that the Greedy variants (MeZO-Greedy and SubZero-Greedy) consistently achieve higher accuracy compared to their standard counterparts (MeZO and SubZero). For instance, MeZO-Greedy outperforms standard MeZO, and SubZero-Greedy exhibits superior performance over standard SubZero. This trend suggests that Greedy strategies are more effective in optimizing model performance, particularly in resource-constrained

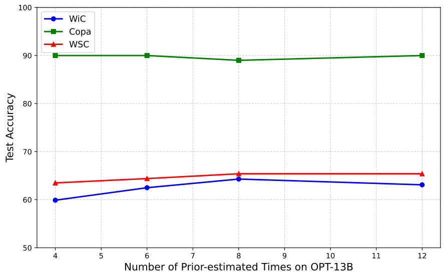
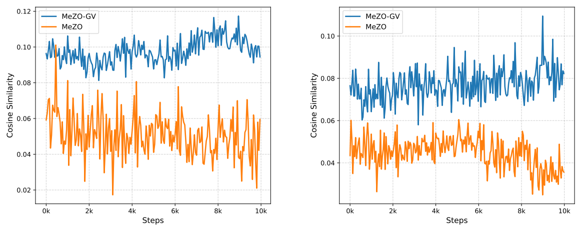


Figure 3: Performance of OPT-13B Model Across three Datasets as a Function of Prior-Estimated Times



(a) SST-2 (b) BoolQ

Figure 4: Cosine similarity between the estimated gradient $\hat{\mathbf{g}}$ and the true gradient $\mathbf{g}$ computed by SGD, on SST-2 and BoolQ using OPT-1.3B in the prefix tuning scheme.

scenarios. Moreover, when combined with techniques like LoRA (Low-Rank Adaptation), the Greedy variants (e.g., MeZO-Greedy (LoRA)) maintain or even enhance accuracy while reducing computational costs. The performance advantage of the Greedy methods is consistent across different datasets and model sizes, demonstrating their robustness and broad applicability. These findings highlight the effectiveness of the Greedy strategies in improving model accuracy and efficiency. Additionally, in Appendix C.2, we provide the training loss convergence curves based on the Greedy strategy, which reveal that perturbations guided by prior knowledge accelerate the model's convergence speed and achieve better performance compared to the original baseline.

4.4 Impact of the Number of Evaluations

In Figure 3, we illustrate the performance of the OPT-13B model across three datasets—WIC, Copa, and WSC—as the number of evaluations varies from 4 to 12. The Copa and WSC datasets exhibit stable performance with increasing evaluations, suggesting limited sensitivity to additional iterations. In contrast, the WIC dataset demonstrates the most significant improvement, highlighting its stronger dependence on the number of evaluations. These findings reveal that the impact of the number of evaluations varies substantially across datasets, emphasizing the need for dataset-specific optimization strategies. Notably, the experiments indicate that for many datasets, increasing the number of evaluations does not consistently enhance performance; often, only a few iterations are sufficient to achieve robust results. We further support this observation with a theoretical analysis confirming that excessive evaluations are not always beneficial. While additional evaluations may accelerate model convergence, they can also increase the risk of overfitting. Therefore, a balanced approach, carefully tailored to the unique characteristics of each dataset, is essential for achieving optimal performance.

4.5 Directional Alignment Analysis

To quantitatively assess the quality of zeroth-order gradient estimation, we examine the directional alignment between the estimated gradient $\hat{\mathbf{g}}$—obtained via MeZO or MeZO-GV—and the true gradient $\mathbf{g}$, which is computed using stochastic gradient descent (SGD). Specifically, we calculate the expected cosine similarity $\cos(\mathbf{g}, \hat{\mathbf{g}})$ as a measure of alignment quality. Figure 4 illustrates the alignment trends on SST-2 and BoolQ using the OPT-1.3B model under the prefix tuning setting. All methods are trained with a batch size of 16 for 10K steps. As illustrated in Figure 4, MeZO-GV consistently achieves a higher cosine similarity compared to the standard MeZO baseline and closely follows the direction of the true gradient obtained via SGD. These empirical findings provide robust support for our theoretical analysis, which predicts enhanced alignment when perturbations are guided by prior-informed directions.

5 Theory Analysis

To discuss the approximation efficiency of the expectation in Equation 2, we propose the following lemma.

Lemma 1. *Under the ZO setting, assume the dimsension of the problem is d, and the sampling number is k, $z_1, z_2, ..., z_k \sim N(0, I_d)$. Let ϵ be a small positive number, $\delta > 0$ and $S_k = \frac{1}{k}\sum_{i=1}^{k} z_i z_i^T$. When $k = O\left(\frac{1}{\epsilon^2}\log\left(\frac{d}{\delta}\right)\right)$, with probability $p \geq 1-\delta$, we have $\|S_k - I_d\| \leq \epsilon$. Please note that the experimental configuration of $MeZO(K=1)$ is far from the theoretical bound, indicating that its* *approximation efficiency is not satisfactory.*

Lemma 2. *Under the ZO settings, assume the dimension of the optimization problem is* d*, and the sampling number is* k*, where* $z_1, z_2, \ldots, z_k$ *are all standard normal distributions, and* g *is the gradient direction (without loss of generality, assume the length is 1). Define* $V = \frac{1}{k}\sum_i z_i z_i^T g$, $V_{\parallel} = (V^T g)g$, $V_{\perp} = V - V_{\parallel}$*. We have the following conclusions:*

$$ratio_1 = \frac{\|V_{\parallel}\|}{\|V_{\perp}\|} \approx \sqrt{\frac{k}{d-1}}.$$

$$ratio_2 = \frac{\|V_{\parallel}\|}{\|g\|} \approx 1.$$

The **Lemma 2** analyzes the norm ratio between the parallel component (aligned with the true gradient g) and the orthogonal component. For k Gaussian random vectors, the parallel component dominates with a ratio $\|V_{\parallel}\|/\|V_{\perp}\| \approx \sqrt{\frac{k}{d-1}}$, while its length aligns closely with g ($\|V_{\parallel}\|/\|g\| \approx 1$).

Lemma 3. *Under the ZO combined with greedy permutation, assume the dimension of the optimization problem is* d*, and the sampling number is* k*, where* $z_1, z_2, \ldots, z_k$ *are all standard normal distributions, and* g *is the gradient direction (without loss of generality, assume the length is 1). Using decomposition, we have* $z_i = (z_i^T g)g + z_{i,\perp}$, $Y_i = z_i^T g$ *and let* $Y_1 = \min_{1 \le i \le k} Y_i$*, we have its PDF as*

$$f(y) = k(1 - \Phi(y))^{k-1}\phi(y).$$

Now $V = z_1 z_i^T g = (Y_1 g + z_{1,\perp})(Y_1 g + z_{1,\perp})^T g = Y_1^2 g + Y_1 z_{1,\perp}$, $V_{\parallel} = Y_1^2 g$, $V_{\perp} = Y_1 z_{1,\perp}$*. We have the following conclusions:*

$$ratio_1 = \frac{\|V_{\parallel}\|}{\|V_{\perp}\|} = \frac{|Y_1|}{|z_{1,\perp}|} \approx \frac{|Y_1|}{\sqrt{d-1}} \approx \frac{\sqrt{2\log(k)}}{\sqrt{d-1}}.$$

$$ratio_2 = \frac{\|V_{\parallel}\|}{\|g\|} = Y_1^2 \approx 2\log(k).$$

The norm ratio of the parallel component to the true gradient satisfies ratio$_2 = \frac{\|V_{\parallel}\|}{\|g\|} \approx 2\log(k)$, enhancing the dominance of the parallel component, making the ZO estimation more accurate along the gradient direction.

Lemma 4. *Under the ZO combined with guding vector, assume the dimension of the optimization problem is* d*, and the sampling number is* k*, where* $z_1, z_2, \ldots, z_k$ *are all standard normal distributions,* g *is the gradient direction with norm 1,* σ *is the ratio of sparks selected to calculate guding vector,* $s = [\sigma * k]$*. Using decomposition, we have* $z_i = (z_i^T g)g + z_{i,\perp}$, $Y_i = z_i^T g$ *and let* $Y_1 < Y_2 < \ldots < Y_k$, $\wedge_1 = \{1, 2, ..., s\}, \wedge_2 = \{k, k-1, k-2, ..., k-s+1\}, \hat{z} = \frac{1}{s}\left(\sum_{i \in \Lambda_1} z_i - \sum_{j \in \Lambda_2} z_j\right)$, $V = \hat{z}\hat{z}^T g = V_{\parallel} + V_{\perp}$*,We have the following conclusions:*

$$ratio_1 = \frac{\|V_{\parallel}\|}{\|V_{\perp}\|} = \frac{2\sqrt{s\log k}}{\sqrt{d-1}}$$

$$ratio_2 = \frac{\|V_{\parallel}\|}{\|g\|} = 8s\log k$$

The ratios ratio$_1 = \frac{2\sqrt{s\log k}}{\sqrt{d-1}}$ and ratio$_2 = 8s\log k$ show that increasing s or k significantly boosts $\|V_{\parallel}\|$, enhancing alignment with the gradient direction.

Table 5 compares the current ZO algorithm and its variants in terms of the gradient-aligned component ratio. Both ZO-Greedy and ZO-GV significantly improve the gradient-aligned component ratio through the greedy strategy and guiding vector, respectively. In particular, ZO-GV achieves the best performance when increasing s and k. In Appendix F, we provide a detailed theoretical proof to support these findings.

Table 5: Comparison of Gradient-aligned Component Ratio

Algorithm Index	ZO	ZO-Greedy	ZO-GV
$\frac{\|V_{\parallel}\|}{\|V_{\perp}\|}$	$\sqrt{\frac{k}{d-1}}$	$O\left(\frac{\sqrt{\log(k)}}{\sqrt{d-1}}\right)$	$O\left(\frac{\sqrt{s\log k}}{\sqrt{d-1}}\right)$
$\frac{\|V_{\parallel}\|}{\|g\|}$	1	$O(\log(k))$	$O(s\log(k))$

6 Conclusion

In this paper, we propose two distinct prior-informed approaches to enhance zeroth-order optimization: a guiding vector-augmented strategy and a greedy perturbation strategy. Both methods leverage prior knowledge to significantly improve optimization performance and efficiency. Theoretically and empirically, our approaches achieve more substantial directional alignment with the true gradient, drastically reducing the number of convergence iterations while maintaining high accuracy. These innovations underscore the effectiveness of prior-guided perturbations, providing scalable and efficient solutions for optimizing LLMs.

Limitations

Despite demonstrating promising results, this study has several limitations that warrant discussion. First, while the proposed prior-informed perturbation strategy proves effective, the research does not systematically investigate which types of prior knowledge are most suitable for different model architectures or task requirements, leaving the optimal selection criteria for future exploration. Second, although designed as a plug-and-play framework and validated against mainstream approaches, the method's generalizability across the full spectrum of existing zeroth-order optimization techniques remains to be comprehensively verified. Third, constrained by computational resources, our empirical validation was limited to models with up to 30 billion parameters, leaving open questions about the method's scalability to hundred-billion or trillion-parameter models and its potential trade-offs in such extreme-scale scenarios. These limitations highlight valuable directions for future research to further strengthen the framework's theoretical foundations and practical applicability.

Ethics Statement

This study raises no ethical concerns, as it involves no human or animal subjects, confidential data, or sensitive materials. All research was conducted using publicly available data, adhering to standard academic integrity guidelines. No personally identifiable information was used, and all sources are properly cited to avoid plagiarism or misrepresentation.

References

OpenAI Josh Achiam, Steven Adler, Sandhini Agarwal, and et al. 2023. Gpt-4 technical report.

Roy Bar-Haim, Ido Dagan, and Idan Szpektor. 2014. Benchmarking applied semantic inference: The PASCAL recognising textual entailment challenges. In *Language, Culture, Computation. Computing - Theory and Technology - Essays Dedicated to Yaacov Choueka on the Occasion of His 75th Birthday, Part I*, volume 8001 of *Lecture Notes in Computer Science*, pages 409–424. Springer.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *NeurIPS*.

HanQin Cai, Yuchen Lou, Daniel Mckenzie, and Wotao Yin. 2021. A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization. *ArXiv*, abs/2102.10707.

Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Konstantinos Parasyris, Jiancheng Liu, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. 2024. Deepzero: Scaling up zeroth-order optimization for deep model training. In *The Twelfth International Conference on Learning Representations*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *ACL*.

Tanmay Gautam, Youngsuk Park, Hao Zhou, Parameswaran Raman, and Wooseok Ha. 2024. Variance-reduced zeroth-order methods for fine-tuning language models. In *Forty-first International Conference on Machine Learning*.

Daniel Golovin, John Karro, Greg Kochanski, Chansoo Lee, Xingyou Song, and Qiuyi Zhang. 2020. Gradientless descent: High-dimensional zeroth-order optimization. In *International Conference on Learning Representations*.

Wentao Guo, Jikai Long, Yimeng Zeng, Zirui Liu, Xinyu Yang, Yide Ran, Jacob R. Gardner, Osbert Bastani, Christopher De Sa, Xiaodong Yu, Beidi Chen, and Zhaozhuo Xu. 2024. Zeroth-order fine-tuning of LLMs with extreme sparsity. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024)*.

Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195.

Geoffrey E. Hinton. 2022. The forward-forward algorithm: Some preliminary investigations. *ArXiv*, abs/2212.13345.

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 *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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 *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 18363–18371. AAAI Press.

Feihu Jin, Yifan Liu, and Ying Tan. 2024. Derivative-free optimization for low-rank adaptation in large language models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:4607–4616.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Taebum Kim, Hyoungjoon Kim, Gyeong-In Yu, and Byung-Gon Chun. 2023. Bpipe: Memory-balanced pipeline parallelism for training large language models. In *International Conference on Machine Learning*.

Hector J. Levesque. 2011. The winograd schema challenge. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*.

Sijia Liu, Pin-Yu Chen, Xiangyi Chen, and Mingyi Hong. 2019. signsgd via zeroth-order oracle. In *International Conference on Learning Representations*.

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

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. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Horia Mania, Aurelia Guy, and Benjamin Recht. 2018. Simple random search of static linear policies is competitive for reinforcement learning. In *Neural Information Processing Systems*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyang Yang, Minjia Zhang, Dong Li, and Yuxiong He. 2021. Zero-offload: Democratizing billion-scale model training. *ArXiv*, abs/2101.06840.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06, Stanford, California, USA, March 21-23, 2011*. AAAI.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323:533–536.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew 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, Seattle, Washington, USA. Association for Computational Linguistics.

J.C. Spall. 1992. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341.

Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuanjing Huang, and Xipeng Qiu. 2022a. BBTv2: Towards a gradient-free future with large language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3930, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022b. Black-box tuning for language-model-as-a-service. In *Proceedings of ICML*.

Yujie Tang and Na Li. 2019. Distributed zero-order algorithms for nonconvex multi-agent optimization. *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 781–786.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *NeurIPS*.

Yifan Yang, Kai Zhen, Ershad Banijamali, Athanasios Mouchtaris, and Zheng Zhang. 2024. 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, Miami, Florida, USA. Association for Computational Linguistics.

Ziming Yu, Pan Zhou, Sike Wang, Jia Li, and Hua Huang. 2024. Subzero: Random subspace zeroth-order optimization for memory-efficient LLM fine-tuning. *CoRR*, abs/2410.08989.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. 2018. Record: Bridging the gap between human and machine commonsense reading comprehension. *CoRR*, abs/1810.12885.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068.

Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D. Lee, Wotao Yin, Mingyi Hong, Zhangyang Wang, Sijia Liu, and Tianlong Chen. 2024. Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark. In *ICML*.

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor Wai-Hung Tsang. 2024. Second-order fine-tuning without pain for llms: A hessian informed zeroth-order optimizer. *ArXiv*, abs/2402.15173.

A Datasets and Setup

As shown in Table 6, the datasets utilized in our experiments encompass three types of tasks: classification tasks, multiple choice tasks, and question-answer tasks. Previous studies (Brown et al., 2020; Gao et al., 2021; Schick and Schütze, 2021) have demonstrated that incorporating appropriate prompts ensures that fine-tuning objectives are closely aligned with the pre-training one. Specifically, simple prompts can streamline the fine-tuning optimization, enabling zeroth-order methods to work efficiently (Malladi et al., 2023). We investigate three fine-tuning schemes to validate the proposed method: full-tuning (FT), which fine-tunes the entire pre-trained model; low-rank adaptation (LoRA), which fine-tunes the model by introducing low-rank weight perturbations (Hu et al., 2022); and prefix-tuning (Prefix), which fine-tunes the model by appending learnable parameters to the attention mechanism of Transformers (Li and Liang, 2021). For further details, please refer to Appendix D.

Setup. We compare our methods with zero-shot, in-context learning (ICL), and fine-tuning with Adam (FT). Additionally, we validate the effectiveness of our methods by applying them to MeZO (Malladi et al., 2023) and SubZero (Yu et al., 2024). Following the MeZO, we randomly sample 1,000 examples for training, 500 examples for validation, and 1,000 examples for testing. Unless otherwise specified, we set the query budget per gradient estimation to $q = 1$ and the hyperparameter α to 0.5. The number of prior-estimated times M is set to either 2 or 4. We execute MeZO and SubZero for 20,000 steps, while our proposed method is trained for 10,000 steps. All models are validated every 1,000 steps. To reduce memory consumption, we employ half-precision training (FP16) for zeroth-order optimization (ZO) methods. All experiments are conducted on Nvidia A100 GPUs with 80GB of memory or Nvidia 3090 GPUs with 24GB of memory. Detailed learning rates, batch sizes, and other hyperparameter configurations for the different models are provided in Table 7 and Table 8. Our code is available in https://github.com/stan-anony/MeZO-GV

Table 6: The prompts of the datasets used in our OPT experiments.

Dataset Type	Task Type	Prompt
SST-2	cls.	<text> It was terrible/great
RTE	cls.	<premise> Does this mean that "<hypothesis>" is true? Yes or No? Yes/No
CB	cls.	Suppose <premise> Can we infer that "<hypothesis>"? Yes, No, or Maybe? Yes/No/Maybe?
BoolQ	cls.	<passage> <question>? Yes/No
WSC	cls.	<text> In the previous sentence, does the pronoun "<span2>" refer to "<span1>"? Yes or No? Yes/No
WIC	cls.	Does the word "<word>" have the same meaning in these two sentences? Yes or No? <sent1> <sent2> Yes/No
MultiRC	cls.	<paragraph> Question: <question> I found this answer "<answer>". Is that correct? Yes or No? Yes/No
COPA	mch.	<premise> so/because <candidate>
ReCoRD	mch.	<passage> <query>.replace("@placeholder", <candidate>)
SQuAD	QA	Title: <title> Context: <context> Question: <question> Answer:
DROP	QA	Passage: <context> Question: <question> Answer:

Table 7: Experiment Hyperparameters on OPT Models

Experiment	Hyperparameter	Value
MeZO (FT)	Batch Size	16
	Learning Rate	{1e-7, 2e-7, 5e-7}
	ϵ	1e-3
MeZO (LoRA)	Batch Size	16
	Learning Rate	{3e-5, 5e-5, 1e-4}
	ϵ	1e-2
MeZO (Prefix)	Batch Size	16
	Learning Rate	{1e-3, 5e-3, 1e-2}
	ϵ	1e-1
SubZero (FT)	Batch Size	16
	Learning Rate	{1e-7, 2e-7, 5e-7}
	ϵ	1e-3
	Rank	{32, 64}
	Subspace Change Frequency	{500, 1000, 2000}
SubZero (LoRA)	Batch Size	16
	Learning Rate	{3e-5, 5e-5, 1e-4}
	ϵ	1e-2
	Rank	{32, 64}
	Subspace Change Frequency	{500, 1000, 2000}
SubZero (Prefix)	Batch Size	16
	Learning Rate	{1e-3, 5e-3, 1e-2}
	ϵ	1e-1
	Rank	{8, 16}
	Subspace Change Frequency	{500, 1000, 2000}
MeZO-GV (FT)	Batch Size	16
	Learning Rate	{1e-7, 2e-7, 3e-7, 5e-7}
	ϵ	1e-3
	k	{4, 8, 12}
MeZO-GV (LoRA)	Batch Size	16
	Learning Rate	{3e-5, 5e-5,1e-4}
	ϵ	1e-2
	k	{4, 8, 12}
MeZO-GV (Prefix)	Batch Size	16
	Learning Rate	{1e-3, 5e-3, 1e-2}
	ϵ	1e-1
	k	{4, 8, 12}
SubZero-GV (FT)	Batch Size	16
	Learning Rate	{1e-7, 2e-7, 3e-7, 5e-7}
	k	{4, 8, 12}
	ϵ	1e-3
	Rank	{32, 64}
	Subspace Change Frequency	{500, 1000, 2000}
SubZero-GV (LoRA)	Batch Size	16
	Learning Rate	{3e-5, 5e-5,1e-4}
	ϵ	1e-2
	k	{4, 8, 12}
	Rank	{32, 64}
	Subspace Change Frequency	{500, 1000, 2000}
SubZero-GV (Prefix)	Batch Size	16
	Learning Rate	{1e-3, 5e-3, 1e-2}
	ϵ	1e-1
	k	{4, 8, 12}
	Rank	{8, 16}
	Subspace Change Frequency	{500, 1000, 2000}
SGD (FT)	Batch Size	16
	Learning Rate	{1e-4, 1e-3, 5e-3}

Table 8: Experiment Hyperparameters on Llama2 Models

Experiment	Hyperparameter	Value
MeZO (FT)	Batch Size	16
	Learning Rate	{1e-7, 2e-7, 5e-7}
	ϵ	1e-3
MeZO (LoRA)	Batch Size	16
	Learning Rate	{3e-5, 5e-5, 1e-4}
	ϵ	1e-2
MeZO-GV (FT)	Batch Size	16
	Learning Rate	{1e-7, 2e-7, 3e-7, 5e-7}
	ϵ	1e-3
	k	{4, 8, 12}
MeZO-GV (LoRA)	Batch Size	16
	Learning Rate	{3e-5, 5e-5,1e-4}
	ϵ	1e-2
	k	{4, 8, 12}

B Related Work

Gradient-free Optimization of LLMs Recent advancements in gradient-free optimization have utilized evolutionary algorithms, particularly the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier, 2001), to optimize continuous prompt vectors in black-box tuning methods. This approach has demonstrated significant advantages for applying large language models by reducing complexity. However, training these prompt vectors has exhibited instability and slow convergence rates (Sun et al., 2022b,a). To address these issues, Jin et al. (2024) proposed a gradient-free optimization framework for low-rank adaptation to stabilize training and improve convergence speed.

Zeroth-Order Optimization of LLMs Zeroth-order optimization (ZO) has emerged as a pivotal gradient-free method in machine learning, particularly in scenarios where gradient computation is infeasible or prohibitively expensive (Ren et al., 2021; Kim et al., 2023; Chen et al., 2024). ZO has also inspired the development of distributed optimization techniques (Tang and Li, 2019) and has been effectively applied to black-box adversarial example generation in deep learning (Cai et al., 2021; Liu et al., 2019). In addition, several ZO methods have been proposed that achieve optimization without explicitly estimating gradients (Golovin et al., 2020; Mania et al., 2018; Hinton, 2022). Recently, the application of ZO optimization to fine-tuning LLMs has demonstrated significant reductions in GPU utilization and memory footprint (Malladi et al., 2023; Gautam et al., 2024; Zhang et al., 2024). These advancements have catalyzed a growing body of research on zeroth-order optimization techniques tailored for LLMs. Recent advancements in ZO optimization have primarily focused on enhancing convergence rates and minimizing gradient estimation variance to optimize fine-tuning of LLMs. Increasing the batch size has effectively reduced noise in ZO gradient estimation (Gautam et al., 2024; Jiang et al., 2024). Sparse perturbation strategies improve efficiency by selectively perturbing a subset of parameters, thereby reducing computational overhead and gradient variance (Liu et al., 2024; Guo et al., 2024). These strategies achieve sparse parameter perturbations through techniques such as random and sparse pruning masks (Liu et al., 2024) or block-coordinate perturbations (Zhang et al., 2024). Notably, Guo et al. (2024) extended zero-order optimization to the Adam algorithm, while Zhao et al. (2024) enhanced model inference performance by incorporating Hessian matrix-based gradient estimation in ZO optimization, albeit at the expense of increased memory consumption. Additionally, innovative approaches have been proposed to reduce the number of trainable parameters, such as mapping models to subspaces and employing PEFT methods (Hu et al., 2022; Li and Liang, 2021) alongside tensorized adapters (Yang et al., 2024).

C More Results and Analysis

C.1 Training Loss Curves on Different Models

We present the training loss curves of the GV-based method across various models, including datasets such as SST-2, WSC, BoolQ, and CB across OPT and Llama2 models, further demonstrating the effectiveness of our approach. The GV-based method achieves a faster gradient descent at each step, reaching convergence in significantly fewer iterations compared to baselines.

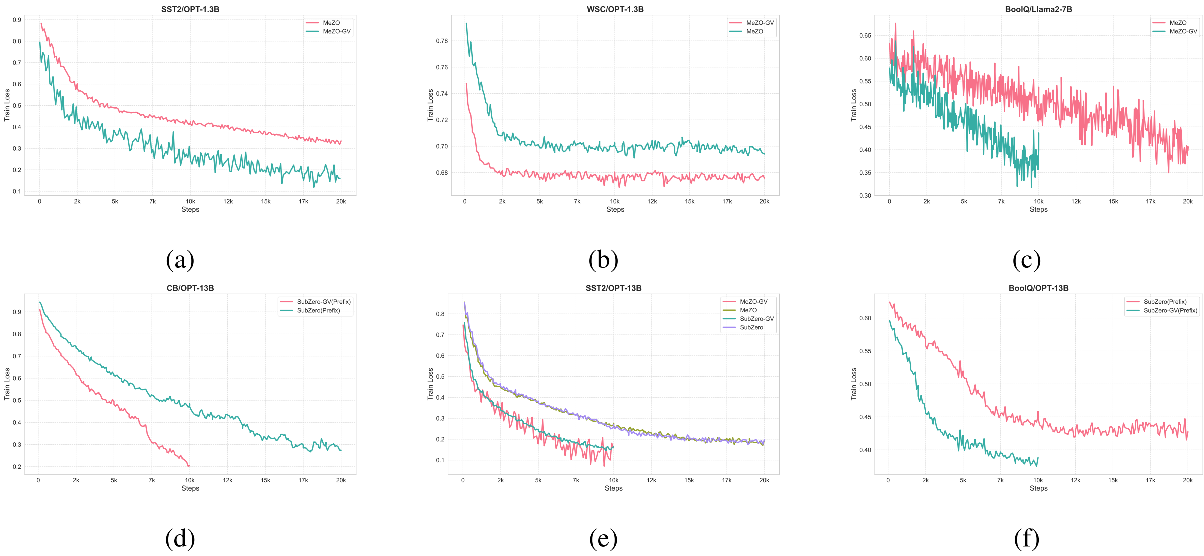


Figure 5: Training loss on SST2, WSC, BoolQ, and CB Tasks for OPT-1.3B/13B and Llama2-7B Models. For OPT-1.3B/13B, we employ a learning rate of 2e-7, while for Llama2-7B, a learning rate of 5e-7 is used. All experiments are conducted with a consistent batch size of 16.

C.2 Training Loss Curves of Greedy Stategy on Different Models

The training loss curves of the greedy-based method across various models, including datasets such as BoolQ and RTE.

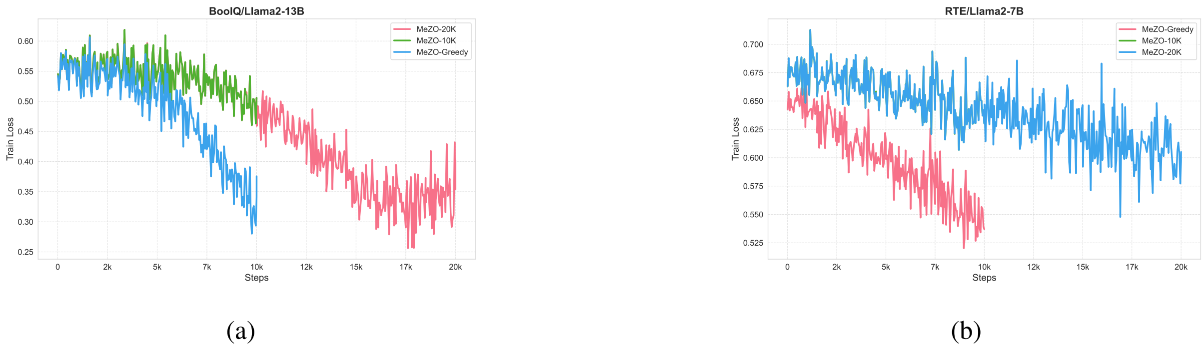


Figure 6: Training loss on BoolQ and RTE Tasks with Llama2-7B Model. We employ a learning rate of 5e-7. All experiments are conducted with a consistent batch size of 16.

C.3 Additional Results on LLMs

The experimental results on Llama2-13B and OPT-30B models further validate the effectiveness and scalability of guiding vector (GV)-based methods across diverse model sizes and tasks. On Llama2-13B, GV-based methods consistently outperform non-GV variants, demonstrating significant performance improvements with reduced training steps. For instance, MeZO-GV-10k(LoRA) achieves 93.7% accuracy on SST2, surpassing MeZO-10k(LoRA) (89.7%) and closely matching the performance of MeZO-20k(LoRA) (94.3%) with only half the training steps. Similarly, on RTE, MeZO-GV-10k(LoRA) attains 72.2% accuracy, outperforming MeZO-10k(LoRA) (66.8%) and approaching the results of MeZO-20k(LoRA) (70.4%). For BoolQ, GV methods exhibit notable improvements: MeZO-GV-10k(LoRA) achieves 83.3% accuracy, surpassing MeZO-10k(LoRA) (76.3%) and MeZO-20k(LoRA) (82.1%). In more challenging tasks such as WSC and WIC, GV methods also demonstrate consistent gains: MeZO-GV-10k(LoRA) achieves 65.4% on WSC and 65.8% on WIC, exceeding both MeZO-10k(LoRA) (59.6%

Table 9: Task Performance Comparison for Different Methods on Llama2-13B

Task	SST2	RTE	BoolQ	WSC	WIC
MeZO-10k(LoRA)	89.7	66.8	76.3	59.6	59.9
MeZO-20k(LoRA)	94.3	70.4	82.1	61.5	62.7
MeZO-GV-10k(LoRA)	93.7	72.2	83.3	65.4	65.8

Table 10: Task Performance Comparison on OPT-30B

Task	SST2	RTE	BoolQ	WSC	WIC
Zero-shot	56.7	52.0	39.1	38.5	50.2
ICL	81.9	66.8	66.2	56.7	51.3
MeZO (prefix)	87.5	72.6	73.5	55.7	59.1
MeZO-GV(prefix)	91.4	75.8	77.4	61.5	62.7
SubZero (prefix)	89.3	74.0	76.8	59.6	58.3
SubZero-GV(prefix)	91.6	75.1	79.4	61.5	62.9

and 59.9%) and MeZO-20k(LoRA) (61.5% and 62.7%). These findings underscore the efficiency of GV methods in achieving competitive performance with fewer training iterations.

On the OPT-30B model, GV-based methods also demonstrate superior performance compared to non-GV variants and baseline approaches. For example, MeZO-GV(prefix) achieves 91.4% accuracy on SST2, outperforming MeZO(prefix) (87.5%) and SubZero(prefix) (89.3%). On RTE, MeZO-GV(prefix) attains 75.8% accuracy, surpassing MeZO(prefix) (72.6%) and SubZero(prefix) (74.0%). For BoolQ, GV methods show significant improvements: MeZO-GV(prefix) achieves 77.4% accuracy, a notable gain over MeZO(prefix) (73.5%) and SubZero(prefix) (76.8%). In more complex tasks such as WSC and WIC, GV methods consistently outperform non-GV approaches: MeZO-GV(prefix) achieves 61.5% on WSC and 62.7% on WIC, demonstrating robust performance gains, highlighting the adaptability and effectiveness of GV methods across different model architectures and task types. These findings position GV-based fine-tuning as a promising approach for efficient adaptation of large-scale language models to downstream applications.

C.4 Memory Usage of Different Methods

Table 11 compares memory usage (in GB) for fine-tuning the OPT-13B model across SST-2, WIC, and BoolQ tasks using zero-shot, in-context learning (ICL), full fine-tuning (FT), and MeZO variants. Zero-shot and ICL exhibit the lowest memory usage, ranging from 26.0 to 29.3 GB, as they do not require parameter updates. In contrast, FT is highly memory-intensive, consuming between 242.3 and 315.3 GB due to the need for full parameter updates. MeZO variants—MeZO-FT, MeZO-LoRA, and MeZO-Prefix significantly reduce memory usage by avoiding full gradient computations, making them efficient alternatives to FT. Notably, MeZO-GV variants, which incorporate guiding vector (GV) techniques, achieve comparable memory efficiency while further enhancing model convergence speed and performance, demonstrating that GV not only maintains low memory usage but also improves optimization effectiveness, making it a powerful tool for resource-constrained fine-tuning of large language models.

D Parameter-Efficient Fine-Tuning (PEFT)

We consider two PEFT methods, including {LoRA, prefix tuning}.

1) Low-Rank Adaptation (LoRA)

LoRA modifies a pre-trained model by introducing trainable low-rank matrices, enabling fine-tuning with a limited parameters. Given a weight matrix $W \in \mathbb{R}^{m \times n}$ in a transformer model, LoRA decomposes it as:

$$W' = W + BA$$

where W is the original weight matrix, $B \in \mathbb{R}^{m \times r}$ and $A \in \mathbb{R}^{r \times n}$ are the low-rank matrices, and $r \ll \min(m, n)$ represents the rank. During fine-tuning, only B and A are updated, keeping W frozen.

Table 11: Memory usage (GB) of fine-tuning OPT-13B, with FT's batch size being 8 and 16 for other tasks.

Method	Task		
	SST-2	WIC	BoolQ
Zero-shot	26.0	26.0	26.3
ICL	27.2	28.5	29.3
FT	242.3	244.7	315.3
MeZO (FT)	28.9	29.1	45.6
MeZO (LoRA)	28.6	29.3	46.5
MeZO (Prefix)	29.5	29.7	46.9
MeZO-GV (FT)	28.9	29.1	45.6
MeZO-GV (LoRA)	28.6	29.3	46.5
MeZO-GV (Prefix)	29.5	29.7	46.9

2) Prefix Tuning

Prefix tuning adds context vectors to the attention mechanism of transformer models. Given an input sequence x, the model processes it with additional context vectors C_k and C_v serving as keys and values in the attention mechanism:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q(K + C_k)^T}{\sqrt{d_k}}\right)(V + C_v)$$

where Q, K, and V represent the query, key, and value matrices in the attention mechanism, $C_k \in \mathbb{R}^{l \times d_k}$, $C_v \in \mathbb{R}^{l \times d_v}$, and l is the length of the prefix. During training, only C_k and C_v are updated, and the original model parameters are frozen.

E Algorithms

E.1 MeZO with Guiding Vector

Algorithm 1 MeZO with Guiding Vector

Require: Parameters $\theta \in \mathbb{R}^d$, loss function $\mathcal{L}(\theta; \mathcal{B})$, step budget T, perturbation scale ϵ, batch size $\mathcal{B}$, learning rate η, weight decay λ, fireworks size M, split ratio $\alpha \in (0, 1)$

```
for iteration t = 1 to T do
    Sample minibatch B_t ~ D and random seed s
    Compute guidance vector: v ← ComputeGuidingVector(θ, M, α, s, B)
    GuidingPerturbation(θ, +ε, v)
    Evaluate L+ ← L(θ; B_t)
    GuidingPerturbation(θ, −2ε, v)
    Evaluate L− ← L(θ; B_t)
    GuidingPerturbation(θ, +ε, v)
    Estimate directional gradient: g ← (L+ − L−)/(2ε)
    Update parameters: θ ← θ − η · (g · v)
end for
```

Algorithm 2 Subroutines for MeZO with Guiding Vector

Subroutine: ComputeGuidingVector(θ, M, α, s, $\mathcal{B}$)
Initialize perturbation set $\mathcal{O} \leftarrow \emptyset$
for particle $i = 1$ to M **do**
  Generate unique seed $s_i \leftarrow \text{Hash}(s \oplus i)$
  RandomPerturbation(θ, ϵ, s_i)
  Evaluate fitness $l_i \leftarrow \mathcal{L}(\theta; \mathcal{B})$
  RandomPerturbation(θ, $-\epsilon$, s_i)
  Store perturbation seed s_i
  $\mathcal{O} \leftarrow \mathcal{O} \cup \{(l_i, s_i)\}$
end for
Sort $\mathcal{O}$ by ascending l_i values
Split into elite/non-elite groups:
  $\mathcal{O}_{\text{top}} \leftarrow \text{First}(\lfloor \alpha M \rfloor, \mathcal{O})$
  $\mathcal{O}_{\text{bottom}} \leftarrow \text{Last}(M - \lfloor \alpha M \rfloor, \mathcal{O})$
Compute guide vector through the z_i corresponding to the seed s_i :
  $v_{\text{top}} \leftarrow \frac{1}{|\mathcal{O}_{\text{top}}|} \sum_{(l_i, s_i) \in \mathcal{O}_{\text{top}}} z_i$
  $v_{\text{bottom}} \leftarrow \frac{1}{|\mathcal{O}_{\text{bottom}}|} \sum_{(l_i, s_i) \in \mathcal{O}_{\text{bottom}}} z_i$
$v \leftarrow v_{\text{top}} - v_{\text{bottom}}$
Return v

Subroutine: GuidingPerturbation(θ, ϵ, v)
for each parameter $\theta_j \in \theta$ **do**
  $\theta \leftarrow \theta + \epsilon \cdot v$
end for

Subroutine: RandomPerturbation(θ, ϵ, s)
Reset random number generator with seed s
for each parameter $\theta_j \in \theta$ **do**
  $z_j \sim \mathcal{N}(0, 1)$
  $\theta_j \leftarrow \theta_j + \epsilon \cdot z_j$
end for

E.2 MeZO with Greedy Strategy

Algorithm 3 MeZO with Greedy Strategy

Require: Parameters $\theta \in \mathbb{R}^d$, loss function $\mathcal{L}(\theta; \mathcal{B})$, step budget T, perturbation scale ϵ, batch size $\mathcal{B}$, learning rate η, weight decay λ, candidate perturbations M
for iteration $t = 1$ to T **do**
  Sample minibatch $\mathcal{B}_t \sim \mathcal{D}$ and random seed s
  Compute optimal perturbation: $z^* \leftarrow$ ComputeGreedyPerturbation$(\theta, M, \epsilon, s, \mathcal{B}_t)$
  GreedyPerturbation(θ, $+\epsilon$, z^*)
  Evaluate $\mathcal{L}^+ \leftarrow \mathcal{L}(\theta; \mathcal{B}_t)$
  GreedyPerturbation(θ, -2ϵ, z^*)
  Evaluate $\mathcal{L}^- \leftarrow \mathcal{L}(\theta; \mathcal{B}_t)$
  GreedyPerturbation(θ, $+\epsilon$, z^*)
  Estimate directional gradient: $g \leftarrow (\mathcal{L}^+ - \mathcal{L}^-)/(2\epsilon)$
  Update parameters: $\theta \leftarrow \theta - \eta \cdot (g \cdot z^*)$
end for

Algorithm 4 Subroutines for MeZO with Greedy Strategy

Subroutine: ComputeGreedyPerturbation(θ, M, ϵ, s, $\mathcal{B}$)
Initialize perturbation set $\mathcal{O} \leftarrow \emptyset$
for particle $i = 1$ to M **do**
    Generate unique seed $s_i \leftarrow \text{Hash}(s \oplus i)$
    RandomPerturbation(θ, ϵ, s_i)
    Evaluate fitness $l_i \leftarrow \mathcal{L}(\theta; \mathcal{B})$
    RandomPerturbation(θ, $-\epsilon$, s_i)
    Store perturbation z_i and loss l_i
    $\mathcal{O} \leftarrow \mathcal{O} \cup \{(l_i, z_i)\}$
end for
Find the optimal perturbation:
    $z^* \leftarrow \text{arg min}_{(l_i,z_i)\in\mathcal{O}} l_i$
Return z^*

Subroutine: GreedyPerturbation(θ, ϵ, z^*)
for each parameter $\theta_j \in \theta$ **do**
    $\theta_j \leftarrow \theta_j + \epsilon \cdot z_j^*$
end for

Subroutine: RandomPerturbation(θ, ϵ, s)
Reset random number generator with seed s
for each parameter $\theta_j \in \theta$ **do**
    $z_j \sim \mathcal{N}(0, 1)$
    $\theta_j \leftarrow \theta_j + \epsilon \cdot z_j$
end for

F Proofs

Proof Lemma 1:

Proof. Using Matrix Bernstein theorem, we have $P(||S_k - I_d|| \geq t) \leq d \cdot exp(\frac{-kt^2}{\sigma^2 + \frac{Lt}{3}})$, where $\sigma^2 = \frac{1}{k}$ and $L = ||z_i||^2 \leq d + O(\sqrt{d})$. Let $t = \epsilon$,$d \cdot exp(\frac{-kt^2}{\sigma^2 + \frac{Lt}{3}}) = \delta$ and the proof is completed. □

Proof Lemma 2:

Proof. It is not difficult to see that $V = (V^T g)g + V_\perp$, $V_{||} = (V^T g)g$ and $V_\perp = V - V_{||} = \frac{1}{k}\sum_{i=1}^{k} z_i^T g z_{i,\perp}$. $E[V_{||}] = g, E[V_\perp] = 0, E[||V_\perp||^2] = Trace(Cov(V_\perp)) = Trace(\frac{1}{k}(I_d - gg^T)) = \frac{d-1}{k}$. □

Proof Lemma 3:

Proof. Suppose we sample k points, and we have

$$Y_1 < Y_2 < Y_3 < ... < Y_k, Y_i = z_i^T g$$

Beacuse selecting the i-th smallest value from k samples implies that this value should fall below approximately $\frac{i}{k+1}$ of all possible samples in the overall distribution. Therefore Y_i aligns with the $\frac{i}{k+1}$-th quantile of the normal distribution:

$$\mathbb{E}[Y_i] \approx \Phi^{-1}\left(\frac{i}{k+1}\right) \approx$$

$$\begin{cases} \Phi^{-1}(p) \approx \sqrt{-2\log(1-p)} & p \to 1 \\ \Phi^{-1}(p) \approx -\sqrt{-2\log(p)} & p \to 0 \end{cases}$$

We can get $|Y_1| \approx \sqrt{2log(k)}$ $\square$

Proof Lemma 4:

Proof.

$$\hat{z} = \frac{1}{s}(\sum_{i\in\wedge_1} z_i - \sum_{j\in\wedge_2} z_j) = \frac{1}{s}(\sum_{i\in\wedge_1}(Y_i g + Z_{i,\perp}) - \sum_{j\in\wedge_2}(Y_j g + Z_{j,\perp}))$$

$$\hat{z} = \frac{1}{s}\left(\underbrace{\left(\sum_{i\in\wedge_1} Y_i - \sum_{j\in\wedge_2} Y_j\right)}_{m\in R} g + \underbrace{\left(\sum_{i\in\wedge_1} Z_{i,\perp} - \sum_{j\in\wedge_2} Z_{j,\perp}\right)}_{N\in R^d}\right)$$

$$V = \hat{z}\hat{z}^T g = \frac{1}{s}(m^2 g + mN) = V_{||} + V_{\perp}$$

$$m = \left(\sum_{i\in\wedge_1} Y_i - \sum_{j\in\wedge_2} Y_j\right) = 2 * \sum_{i\in\wedge_1} Y_i \approx -2\sum_{i=1}^{s}\sqrt{2log\frac{k}{i}}$$

$$m \approx -2s\sqrt{2logk}(1 - \frac{logs}{2logk})$$

$$N = \left(\sum_{i\in\wedge_1} Z_{i,\perp} - \sum_{j\in\wedge_2} Z_{j,\perp}\right), Z_{p,\perp} \sim N(0, I_d - gg^T)$$

$$N \sim N(0, 2s(I_d - gg^T))$$

so we can get that,

$$||N|| \approx \sqrt{2s(d-1)}$$

$ratio_1 = \frac{||V_{||}||}{||V_{\perp}||} = \frac{|m|}{||N||} \approx \frac{2\sqrt{slogk}}{\sqrt{d-1}}$

$ratio_2 = \frac{||V_{||}||}{||g||} = \frac{m^2}{s} \approx 8slogk$ $\square$