Harmony in Divergence: Towards Fast, Accurate, and Memory-efficient Zeroth-order LLM Fine-tuning

Qitao Tan[1] **Jun Liu**[2] **Zheng Zhan**[2] **Caiwen Ding**[3]
Yanzhi Wang[2] **Xiaolong Ma**[4] **Jaewoo Lee**[1] **Jin Lu**[1] **Geng Yuan**[1]
[1]University of Georgia [2]Northeastern University
[3]University of Minnesota [4]University of Arizona

Abstract

Large language models (LLMs) excel across various tasks, but standard first-order (FO) fine-tuning demands considerable memory, significantly limiting real-world deployment. Recently, zeroth-order (ZO) optimization stood out as a promising memory-efficient training paradigm, avoiding backward passes and relying solely on forward passes for gradient estimation, making it attractive for resource-constrained scenarios. However, ZO method lags far behind FO method in both convergence speed and accuracy. To bridge the gap, we introduce a novel layer-wise divergence analysis that uncovers the distinct update pattern of FO and ZO optimization. Aiming to resemble the learning capacity of FO method from the findings, we propose **Di**vergence-driven **Z**eroth-**O**rder (**DiZO**) optimization. DiZO conducts divergence-driven layer adaptation by incorporating projections to ZO updates, generating diverse-magnitude updates precisely scaled to layer-wise individual optimization needs. Our results demonstrate that DiZO significantly reduces the needed iterations for convergence without sacrificing throughput, cutting training GPU hours by up to 48% on various datasets. Moreover, DiZO consistently outperforms the representative ZO baselines in fine-tuning RoBERTa-large, OPT-series, and Llama-series on downstream tasks and, in some cases, even surpasses memory-intensive FO fine-tuning. Our code is released at `https://github.com/Skilteee/DiZO`.

1 Introduction

Fine-tuning large language models (LLMs) via backpropagation achieves strong performance across many NLP tasks [1, 2, 3, 4, 5], but their large parameter counts create substantial memory burdens, limiting downstream applicability. Following neural scaling laws [6, 7], next-generation LLMs grow rapidly, e.g., model sizes increase 410× every two years, far outpacing DRAM bandwidth (1.4×) and capacity (2×) growth. This imbalance leads to the *memory wall*[8], a growing challenge especially for deployment on memory-limited devices[9, 10, 11].

Zeroth-order (ZO) optimization has recently emerged as a memory-efficient approach for LLM fine-tuning, gaining growing attention [12, 13, 14, 15]. By relying solely on forward passes for gradient estimation, ZO eliminates the need for backpropagation and significantly reduces memory usage for activations, gradients, and optimizer states. As shown in [14], ZO fine-tuning can reduce memory cost by up to $12\times$. However, ZO still shows a notable **gap** in convergence speed and accuracy compared to first-order (FO) methods, as shown in Table 1. Although ZO benefits from higher throughput due to its simpler computation, it requires over 10× more iterations to converge, ultimately increasing GPU time. Prior work often attributes this gap to noisy gradient estimates, without exploring other contributing factors [14, 16, 15].

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

To further uncover the fundamental causes of this gap, we begin by analyzing the distinct update patterns shown by FO and ZO methods during LLM fine-tuning. Interestingly, our analysis reveals a substantial difference in the magnitude of weight updates between layers. Specifically, FO methods benefit from fine-grained gradient estimation and enable diverse-magnitude updates precisely scaled to the layer-wise individual optimization needs. In contrast, ZO method tends to behave with uniform-magnitude updates without considering layer-wise individual characteristics. This is attributed to the nature of ZO that relies on high-dimensional random search and leverages random perturbation for gradient estimation. Based on this, we conjecture that the compromised performance of ZO stems from its limited capability in achieving diverse-magnitude updates. This naturally raises the question: *if we could enable ZO to achieve the desired diverse-magnitude updates, could we effectively achieve training acceleration and accuracy improvement?*

Table 1: Fine-tuning results on SST-2 datasets. Although ZO method shows advantages in memory saving, left behind FO method in terms of both accuracy and GPU hours.

Model	**Type**	**Acc.**	**Memory**	**GPU Hours**
RoBERTa	FO	91.9	9.2 GB	12.3%
	ZO	90.5	4.5 GB	100.0%
OPT-2.7B	FO	94.2	45.4 GB	16.8%
	ZO	90.0	6.8 GB	100.0%

To validate our conjecture and fill the performance gap, we innovatively propose **Di**vergence-driven **Z**eroth-**O**rder optimization (**DiZO**), which performs divergence-driven layer adaptation via anchor-based learnable projections, enabling principled adaptive updates that resemble FO methods. Specifically, DiZO guides updates along geometrically constrained directions by learning target distances from an anchor point (e.g., the pre-trained model). We also design a ZO-based method for projection learning that ensures the entire training process is memory-efficient. We extensively evaluate DiZO on a range of tasks, including classification and generation, using models such as RoBERTa-large, the OPT series, and the Llama series. Results show that DiZO significantly reduces training iterations and cuts GPU hours by up to 48% without sacrificing throughput. DiZO also integrates seamlessly with parameter-efficient tuning methods like LoRA [17], and consistently outperforms ZO baselines, sometimes even surpassing memory-intensive FO fine-tuning. Finally, we comprehensively analyze several potential alternatives and validate the necessity and effectiveness of our approach.

The summary of our contributions is as follows:

- We introduce a novel layer-wise divergence analysis to uncover the fundamental differences in the updating patterns of FO and ZO methods.
- We introduce DiZO, a novel ZO method using divergence-driven layer adaptation, achieving a learning capacity closely resembling FO while maintaining the throughput benefit.
- DiZO consistently exceeds existing baselines in both accuracy and GPU hours, and it can be seamlessly integrated with LoRA for additional benefits. These advantages hold across diverse tasks and LLM architectures.
- We also provide comprehensive analysis and discussions on overheads, convergence guarantee, and potential alternatives, which further strengthen the efficiency, feasibility, and necessity of our proposed approach.

2 Preliminaries and Pattern Analysis

2.1 Revisiting Zeroth-order Optimization

Recently, ZO optimization has gained significant attention in machine learning [18, 19, 20, 21]. Unlike conventional FO optimization, which calculates gradients via backpropagation, ZO optimization estimates gradients using only objective oracles via finite differences [22, 23, 24]. This property can be leveraged for LLM fine-tuning to alleviate the extensive memory costs. Specifically, as ZO only needs two forward passes to obtain the estimated gradients, it avoids computing and storing the most memory-consuming information needed in the conventional FO training, i.e., activations in the forward process, gradients in the backward process, and the optimizer state.

The core idea of ZO optimization is to estimate gradients by applying random perturbations to the weights and computing differences in the objective. For a mini-batch of data $\mathcal{B}$, sampled from a

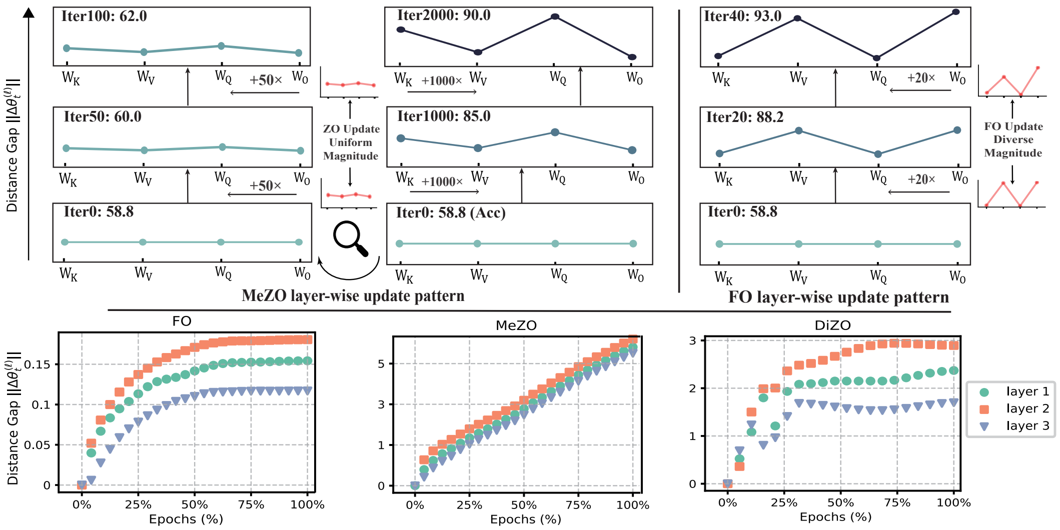

Figure 1: Comparison of the training dynamics of ZO and FO methods. For the upper subfigure, W_K, W_V, W_Q, W_O indicate the corresponding weight matrix in the attention module.

labeled dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{|\mathcal{D}|}$, a model with parameters $\boldsymbol{\theta} \in \mathbb{R}^d$, where d represents the dimension of the parameter space, and the corresponding loss function $\mathcal{L}(\boldsymbol{\theta}; \mathcal{B})$. The gradient is estimated as:

$$\nabla \mathcal{L}(\boldsymbol{\theta}; \mathcal{B}) = \frac{1}{q} \sum_{i=1}^{q} \left[\frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \boldsymbol{u}_i; \mathcal{B}) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \boldsymbol{u}_i; \mathcal{B})}{2\epsilon} \boldsymbol{u}_i \right] \tag{1}$$

where $\boldsymbol{u}_i \sim \mathcal{N}(0, \mathbf{I})$ is a random perturbation typically drawn from standard Gaussian distribution, q is the number of queries, and $\epsilon > 0$ is a small perturbation scalar for smoothing.

Given the learning rate η and the mini-batch data $\mathcal{B}_t$ at t-th iteration, once the estimated gradient $\nabla \mathcal{L}(\theta; \mathcal{B}_t)$ is obtained, then ZO-SGD updates the parameters with the following rule:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_t; \mathcal{B}_t) \tag{2}$$

2.2 Layer-wise Divergence Analysis

ZO optimization applies uniform-magnitude updates across layers, with similar L2-norms per iteration (see Appendix F). This uniformity may underlie its weaker performance. To explore how update divergence impacts convergence and accuracy, we analyze the training dynamics of ZO and FO methods.

Analysis indicator. To quantify the effect of updates, we adapt the layer-wise L2-norm distance gap between the weights of the pre-trained and the weights of fine-tuned model as an indicator. The layer-wise L2-norm distance gap is defined as:

$$\|\Delta \boldsymbol{\theta}_t^{(\ell)}\| = \|\boldsymbol{\theta}_t^{(\ell)} - \boldsymbol{\theta}_0^{(\ell)}\|_2 \tag{3}$$

where t is t-th fine-tuning iteration, ℓ is ℓ-th layer of the model, and $\boldsymbol{\theta}_0^{(\ell)}$ indicates the weights of ℓ-th layer of pre-trained model.

Analysis result. Figure 1 compares the training dynamics of FO and ZO optimization methods, focusing on how each shapes the layer-wise distance gap between model parameters and the pre-trained initialization. Both methods show increasing divergence among layers over time, as illustrated by the twisting lines in the upper sub-figure, indicating that layers benefit from different levels of deviation from the pre-trained model. However, FO and ZO differ in how this divergence accumulates. FO uses fine-grained gradients to produce diverse-magnitude updates, quickly establishing meaningful distance gaps within a few iterations. In contrast, ZO performs random search with uniform-magnitude updates, requiring thousands of iterations to reach similar divergence. Additionally, FO-based methods show fast but converging growth in distance gaps, while ZO-based methods exhibit linear, unconstrained growth. This continued expansion under ZO may reflect a lack of effective constraint, contributing to accuracy drops in later stages and resulting in suboptimal performance.

Algorithm 1: Divergence-diven ZO Optimization (DiZO)

```
Require: parameter of $t$-th iteration $\boldsymbol{\theta_t}$ and pre-trained model $\boldsymbol{\theta_0}$, loss function $\mathcal{L}$, step budget $T$,
  perturbation scalar $\epsilon$, mini-batch data $\mathcal{B}_t$, learning rate $\eta$, projection at $t$-th iteration $\boldsymbol{\gamma_t} = \{\gamma_t^i\}_{i=1}^L$,
  projection update interval $\kappa$
for $t = 1$ to $T$ do
    $\nabla\mathcal{L} = \texttt{GradEst}(\boldsymbol{\theta_t}, \epsilon, \mathcal{B}_t)$;
    $\boldsymbol{\theta_t} = \boldsymbol{\theta_{t-1}} - \eta\nabla\mathcal{L}$;
    if $t \bmod \kappa = 0$ then
        $\boldsymbol{\gamma}_t^* = \arg\min_{\boldsymbol{\gamma}_t} \mathcal{L}(\boldsymbol{\theta}_0 + \frac{\gamma_t}{\|\Delta\boldsymbol{\theta}_t\|}\Delta\boldsymbol{\theta}_t; \mathcal{B}_t)$;
        $\boldsymbol{\theta}_t = \texttt{ApplyProjection}(\boldsymbol{\theta}_t, \boldsymbol{\theta}_0, \boldsymbol{\gamma}_t^*)$;
    end
end
Subroutine GradEst($\boldsymbol{\theta}$, $\epsilon$, $\mathcal{B}$):
    Sample: $u_1, \ldots, u_q \backsim \mathcal{N}(0, \mathbf{I})$;
    Query: $y_i = \mathcal{L}(\boldsymbol{\theta} + \epsilon u_i; \mathcal{B}) - \mathcal{L}(\boldsymbol{\theta} - \epsilon u_i; \mathcal{B})$;
    Estimator: $\nabla\mathcal{L} = \frac{q}{2\epsilon}\sum_{i=1}^q y_i u_i$;
    return $\nabla\mathcal{L}$;
return
Subroutine ApplyProjection($\boldsymbol{\theta}_t$, $\boldsymbol{\theta}_0$, $\boldsymbol{\gamma}_t$):
    for $\ell = 1, 2, \ldots, L$ do
        // Project $l$-th layer
        $\boldsymbol{\theta}_t^{(\ell)} = \boldsymbol{\theta}_0^{(\ell)} + \frac{\gamma_t^{(\ell)}}{\|\Delta\boldsymbol{\theta}_t^{(\ell)}\|}\Delta\boldsymbol{\theta}_t^{(\ell)}$;
    end
    return $\boldsymbol{\theta}_t$;
return
```

3 Methodology

3.1 Design of the Divergence-driven Layer Adaptation

To provide layer-wise adaptive updates for ZO optimization, we propose applying anchor-based learnable projections to the updates of different layers. The pseudocode for the method is shown in Algorithm 1.

Specifically, we treat training iteration as a two-step process that iteratively updates the weights and the projection factor. Our approach involves two key steps performed in an alternating manner. First, we perform vanilla ZO optimization as defined in Eq. (2). Second, we identify the ideal projections for the weights and apply them, generating the projected weights. Formally, we define the ideal projection learning as solving the following minimization problem:

$$\min_{\boldsymbol{\gamma}_t} \mathcal{L}(\boldsymbol{\theta}_0 + \frac{\boldsymbol{\gamma}_t}{\|\Delta\boldsymbol{\theta}_t\|}\Delta\boldsymbol{\theta}_t; \mathcal{B}_t) \tag{4}$$

where $\boldsymbol{\gamma}_t = \{\gamma_t^{(\ell)}\}_{\ell=1}^L$ is a projection vector at t-th iteration, and L is the number of layers. While searching for the ideal projection, we freeze the model weights and use the same mini-batch data $\mathcal{B}_t$ that is employed for the main ZO weight fine-tuning.

After finding the ideal projection for the t-th ZO step, we project the weights as:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_0 + \frac{\boldsymbol{\gamma}_t}{\|\Delta\boldsymbol{\theta}_t\|}\Delta\boldsymbol{\theta}_t \tag{5}$$

where we get the new $\boldsymbol{\theta}_t$ after projection, and then we use the projected one for the following fine-tuning. When the value of $\boldsymbol{\gamma_t}$ is larger than $\|\Delta\boldsymbol{\theta_t}\|$, it enlarges the distance gap between the fine-tuned model and the pre-trained model, and vice versa.

3.2 How to Learn the Projection?

Although promising, finding the ideal projection (defined in Eq. (4)) remains challenging due to the high complexity of the objective. A straightforward solution is to also perform backpropagation for gradient computation and optimize the projection accordingly (FO-based method). For example, we use Adam optimizer to directly update γ_t. The results are shown in Table 2, which significantly reduces 67.7% of the iterations and 58.5% of GPU hours, and increases by 3.4% in accuracy.

Table 2: Fine-tuning OPT-2.7B on SST-2 dataset. ●: partial gradient-free; DiZO†: learning projection by FO method;

Task Type	Gradient Free	Acc.	#Train Iter.	GPU Hours
MeZO	✓	90.0	100%	100%
DiZO† (w. FO)	●	93.4	33.3%	41.5%
FT	✗	94.2	9.3%	16.8%

However, searching projection with the FO method makes DiZO only partially gradient-free. Specifically, while the model weights are updated via ZO, the per-layer projection parameter $\gamma_t^{(\ell)}$ is updated via FO, which still requires the backward pass and storing memory-intensive activation. The only memory saving, compared to the vanilla FO fine-tuning, is the optimizer state. As a result, relying on FO to find the ideal projection, though it achieves faster convergence speed and better accuracy in ZO optimization, offers limited overall benefit. It is worth noting that the peak memory usage during training of the FO-based DiZO is similar to that of low-rank adaptation (LoRA) [17].

3.3 Projection Learning by Zeroth-order Optimization

Our major goal is to find the ideal projection for adaptive updates while avoiding memory-intensive backpropagation. One potential promising solution is to also utilize the ZO method to update the projection. We estimate the gradient and update the projection as:

$$\nabla\widehat{\mathcal{L}}(\gamma_t;\boldsymbol{\theta}_t)=\left[\frac{\widehat{\mathcal{L}}\left(\gamma_t+\epsilon\boldsymbol{u};\boldsymbol{\theta}_t\right)-\widehat{\mathcal{L}}\left(\gamma_t-\epsilon\boldsymbol{u};\boldsymbol{\theta}_t\right)}{2\epsilon}\boldsymbol{u}\right] \tag{6}$$

$$\gamma_{t,j+1}=\gamma_{t,j}-\eta\nabla\widehat{\mathcal{L}}(\gamma_t;\boldsymbol{\theta}_t) \tag{7}$$

where $\boldsymbol{u}\in\mathbb{R}^L$ is a random vector from $\mathcal{N}(0,\mathbf{I})$ and $\widehat{\mathcal{L}}=\mathcal{L}(\boldsymbol{\theta_0}+\frac{\gamma_t}{\|\Delta\boldsymbol{\theta_t}\|}\Delta\boldsymbol{\theta_t};\mathcal{B}_t)$.

However, directly applying vanilla ZO optimization to the sub-task of projection learning yields limited improvement and can even cause failure, undermining the main fine-tuning process (see Appendix C.2). This failure stems from two key issues. First, projection values depend not only on γ_t but also on the distance gap $\|\Delta\boldsymbol{\theta}_t\|$. Ignoring this gap leads to uninformative updates and suboptimal solutions. Second, due to noisy ZO updates over a few iterations, projection magnitudes can become too small or too large. A small projection pulls the model too close to the pre-trained state, erasing progress, while a large one applies overly aggressive updates that destabilize training.

To address the above issues, two strategies are devised.
Re-initialization. To introduce the distance gap $\|\Delta\boldsymbol{\theta}_t\|$ into the projection learning process, the initial value $\gamma_{t,0}$ is reset to $\|\Delta\boldsymbol{\theta}_t\|$ each time the projection is optimized. This means that, initially, the projection magnitude $\frac{\gamma_t}{\|\Delta\boldsymbol{\theta}_t\|}=1$. If not perform projection updates, DiZO reverts to standard ZO.
Projection clipping. To prevent drastic weight changes and maintain training stability, we introduce projection clipping. Specifically, given a clipping range $\tau>0$, if the projection magnitude $\frac{\gamma_t}{\|\Delta\boldsymbol{\theta}_t\|}\notin[1-\tau,1+\tau]$, it is clipped to remain within this interval. This prevents aggressive model adjustments that could destabilize training.

4 Overhead Analysis

We simply analyze the computational overhead of our method here and will elaborate further later.

Memory overhead. Our method requires additional memory as it involves storing the pre-trained model and calculating the weight distance gap with the fine-tuned model, which can become costly when scaling to large LLMs. However, in DiZO, we find that projecting only the weight updates of the *Query* and *Value* layers in the attention module, instead of updating all layers, not only reduces memory requirements but also delivers better performance. As a result, we only need to store the weights of these two types of layers from the pre-trained model, accounting for approximately 16.7% of the parameters in OPT-2.7B, which is a manageable overhead. Similarly, LoRA [17] also focuses on weight decomposition for *Query* and *Value* layers, which echoes our observation.

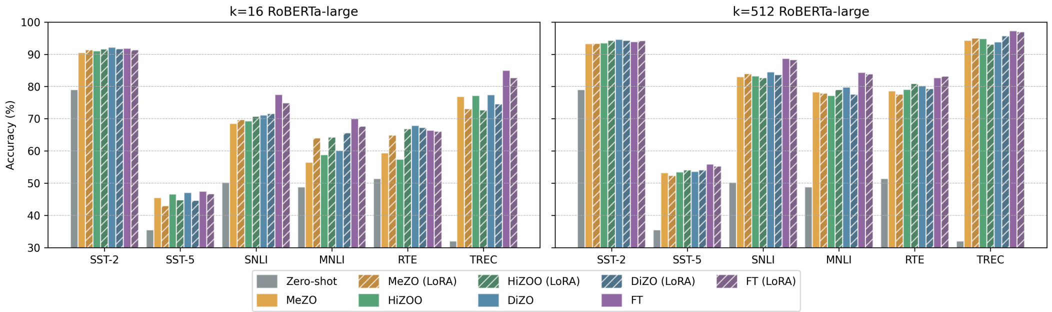

Figure 2: Experiments on RoBERTa-large. DiZO outperforms the baselines with and without LoRA. Detailed numbers are presented in Table E.1, and the loss trajectory is shown in Figure E.1.

Computational overhead. Our method introduces extra computational cost, as the projection is learned alongside the main optimization (fine-tuning). However, we observe that performing projection learning intermittently, only once every few training iterations, does not compromise performance and significantly reduces the added overhead. This strategy reduces the computational burden while maintaining efficiency, allowing DiZO to achieve throughput comparable to vanilla ZO fine-tuning. Additionally, the reduced frequency of projection updates ensures that DiZO remains scalable for larger models and datasets.

5 Convergence Analysis

In this section, we give a nonconvex convergence guarantee in terms of the expected gradient norm. The bound improves over basic ZO-SGD by replacing the factor d (full dimension) with an effective dimension on the order of D where we denote by $D = \max_{1 \le \ell \le L} d^{(\ell)}$.

We assume the following, which are standard in ZO analyses:

Assumption 5.1. $\mathcal{L}$ is L_f-smooth, i.e. there exists $L_f > 0$ such that for all $\boldsymbol{\theta}, \boldsymbol{\theta}'$,

$$\|\nabla\mathcal{L}(\boldsymbol{\theta}) \;-\; \nabla\mathcal{L}(\boldsymbol{\theta}')\| \;\le\; L_f \,\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|.$$

Assumption 5.2. Due to DiZO's layerwise projection step, each $\boldsymbol{\theta}^{(\ell)}$ remains in a ball (or line segment) around $\boldsymbol{\theta}_0^{(\ell)}$ of radius $R^{(\ell)}$. In particular, define

$$\mathcal{S} \;=\; \big\{\boldsymbol{\theta} \;\big|\; \|\boldsymbol{\theta}^{(\ell)} - \boldsymbol{\theta}_0^{(\ell)}\| \;\le\; R^{(\ell)} \;\;\forall\,\ell\big\}.$$

The algorithm ensures $\boldsymbol{\theta}_t \in \mathcal{S}$ for all iterations t.

Theorem 5.3. *Under Assumptions 5.1–5.2, suppose DiZO runs for T iterations with step size $\eta = c/\sqrt{T}$ for a sufficiently small constant $c > 0$. Then there exist constants such that*

$$\min_{0 \le t < T} \mathbb{E}\big[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2\big] \;\;=\;\; O\Big(\frac{\sqrt{D}}{\sqrt{T}}\Big).$$

Note that D can often be much smaller than the total parameter count $\sum_{\ell=1}^{L} d^{(\ell)}$. This drastically improves the variance bounds in the zeroth-order gradient estimation, leading to a faster convergence rate in practice.

Standard ZO-SGD in $\mathbb{R}^d$ often incurs a factor of $\sqrt{d}$ in its nonconvex stationarity bound, due to estimator variance. By restricting each layer $\boldsymbol{\theta}^{(\ell)}$ to remain near the pre-trained $\boldsymbol{\theta}_0^{(\ell)}$, *DiZO* effectively reduces the dimension to $D \ll d$, improving the rate to $O(\sqrt{D}/\sqrt{T})$. The full proof and detailed analysis are provided in Appendix F.2.

6 Experiments

6.1 Experimental Settings

Models and datasets. We evaluate DiZO with various models, including medium-sized masked models [2] (RoBERTa-large) and large-sized autoregressive models [25, 26] with different size, including OPT-2.7B, OPT-6.7B, OPT-13B, Llama2-7B, Llama3-3B, and Llama3-8B. The total parameter size is ranging from 355M to 13B. We evaluate on the SuperGLUE [27] benchmark, MMLU [28] and MT-Bench [29] benchmarks. More details on datasets are shown in Appendix B.1.

Baseline. We mainly compare with two ZO works, memory-efficient ZO optimization (MeZO) [14] and Hessian-informed ZO optimization (HiZOO) [15]. MeZO is a fundamental and representative work in ZO LLM fine-tuning but suffers from slow convergence speed. HiZOO is a recently proposed ZO acceleration for LLM fine-tuning, which leverages the estimated second-order information to speed up. In addition, we also incorporate the parameter-efficient fine-tuning (PEFT) technique LoRA [17].

Evaluation. For SuperGLUE, we follow prior work [30, 14], evaluating few-shot and many-shot settings on RoBERTa-large using $k = 16$ and $k = 512$ samples per class. For each setting, we randomly sample data for training, validation, and testing. For OPT and LLaMA, we use 1000, 500, and 1000 samples respectively. For MMLU and MT-Bench, we adopt the setup in [31, 32], fine-tuning on the Alpaca GPT-4 dataset [33]. All experiments are conducted on NVIDIA A100 and A6000 GPUs, with results averaged over three trials. Full results are provided in Appendix E.

6.2 Medium-sized Masked Language Models

We conduct experiments on RoBERTa-large across three types of datasets and compare DiZO with two ZO baselines. We also explore PEFT by integrating LoRA. Figure 2 presents the results, while Figure E.1 shows the trajectory of training loss curves, indicating the convergence speed of DiZO and MeZO. Our key findings are as follows:

DiZO greatly increases the convergence speed over MeZO. By using divergence-driven layer adaptation, the loss curve of DiZO decreases much faster, cutting the required iterations by over 50% on SST-2, MNLI, and RTE. Moreover, DiZO improves accuracy by 1.7%, 3.6%, and 8.5%.

DiZO outperforms MeZO and achieves results on par with full fine-tuning. From Figure 2, DiZO consistently surpasses MeZO on all six datasets. Notably, on SST-2 and RTE datasets, DiZO even shows better performance than FO full-parameter fine-tuning, increasing by 0.3% and 1.5%.

DiZO is effective for both full-parameter fine-tuning and PEFT. Although DiZO applies projections based on the distance with the pre-trained model, while such prior knowledge does not exist for the decomposed weights of LoRA, it still delivers some gains.

6.3 Large Autoregressive Language Models

Table 3: Experiments results of fine-tuning OPT-2.7B (with 1000 training samples). Better results between MeZO, HiZOO, and DiZO are highlighted in bold.

Dataset	**SST-2**	**RTE**	**CB**	**BoolQ**	**WSC**	**WIC**	**MultiRC**	**SQuAD**	**DROP**
Task Type	classification							generation	
Zero-shot	56.3	54.2	50.0	47.6	36.5	52.7	44.4	29.8	10.0
FT	94.2	81.2	82.1	72.2	63.8	65.8	71.6	78.4	30.3
LoRA	94.6	80.8	82.7	77.7	59.8	64.0	72.8	77.9	31.1
MeZO	90.0	63.5	69.6	67.4	61.5	57.6	**58.7**	68.7	22.9
HiZOO	90.8	60.6	70.4	**68.0**	60.2	56.6	54.8	68.3	23.4
DiZO	**92.5**	**68.2**	**71.4**	67.0	**63.4**	**57.9**	56.4	**69.0**	**24.3**
MeZO LoRA	91.4	66.6	71.1	67.6	59.6	57.0	57.0	70.8	22.5
HiZOO LoRA	90.6	65.2	71.4	67.4	52.6	**58.8**	**59.0**	71.8	22.7
DiZO LoRA	**91.5**	**68.4**	**71.8**	**70.0**	**61.6**	58.4	56.2	**74.4**	**23.3**

To assess the generalizability of DiZO, we run experiments on the OPT and Llama. The overall results are summarized in Table 3, Table 4, and Figure 3 for OPT-2.7B, OPT-6.7B, and Llama series, respectively. We also compare the convergence speeds of DiZO and MeZO on OPT-2.7B across datasets in Figure 4. We highlight key observations from experiments as follows.

DiZO significantly reduces training GPU hours over MeZO. As shown in Table 4, DiZO achieves faster convergence with up to 48% less GPU time by quickly establishing effective layer-wise

Table 4: Experiment results on OPT-6.7B (with 1000 training samples).

Dataset	**SST-2**	**RTE**	**CB**	**WSC**	**SQuAD**
Task Type	———	classification	———		–generation–
MeZO	90.2	73.2	71.4	**62.2**	76.0
HiZOO	90.7	74.2	71.8	62.1	77.3
DiZO	**91.1**	**74.8**	**73.2**	61.8	**78.6**
MeZO LoRA	91.6	71.2	71.4	61.8	76.3
HiZOO LoRA	91.3	**71.3**	71.4	62.1	76.1
DiZO LoRA	**92.4**	70.2	**71.8**	**62.6**	**77.9**

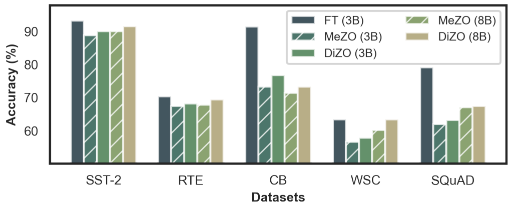

Figure 3: Experiment result on Llama3-3B and Llama3-8B. More results and detailed numbers are shown in Appendix E.4.

divergence. Unlike HiZOO, which reduces iterations but suffers from slow throughput due to costly Hessian estimates, DiZO maintains MeZO-level efficiency with a lightweight projection update using only two forward passes.

DiZO outperforms baselines in both standard and parameter-efficient settings. As shown in Table 3, DiZO consistently outperforms MeZO and HiZOO, with or without LoRA, achieving performance close to FO methods. It ranks first on five of seven classification tasks and leads both text generation tasks. These gains extend to OPT-6.7B (Table 4) and Llama models (Figure 3), highlighting the benefit of layer-wise adaptive updates.

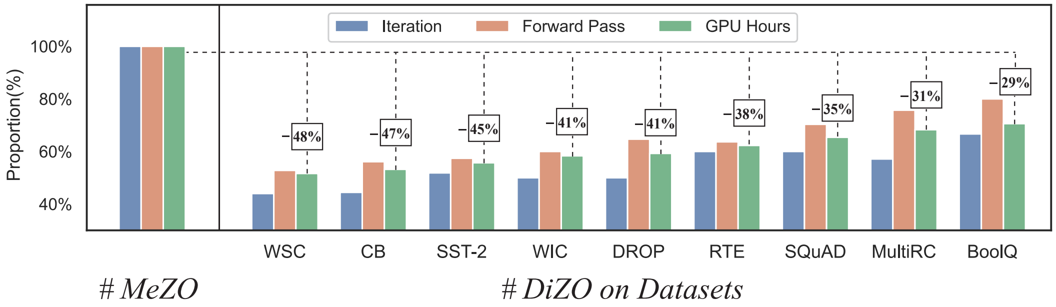

Figure 4: Comparison between MeZO and DiZO on speed.

6.4 Memory and Speed Analysis

Table 5: Memory utilization and speed test on OPT-2.7B on RTE dataset (180 tokens per example on average). ●: partial gradient-free; DiZO†: learning projection with Adam. For a fair comparison, the speed and memory are measured on the same machine with the same settings.

Task Type	**Gradient Free**	**LoRA Added**	**Peak Memory**	**Averaged Memory**	**Throughput**	**#Train Iter.**	**GPU Hours**
FT	✗	✗	62.2 GB	62.2 GB	1.05 it/s	10.0%	16.2%
LoRA	✗	✓	42.5 GB	42.5 GB	2.12 it/s	8.3%	6.6%
DiZO†	●	✗	44.7 GB	10.1 GB	1.43 it/s	33.3%	39.6%
DiZO LoRA†	●	✓	40.1 GB	9.8 GB	2.40 it/s	26.6%	18.8%
MeZO	✓	✗	7.8 GB	7.8 GB	1.70 it/s	100.0%	100.0%
HiZOO	✓	✗	13.2 GB	13.2 GB	1.21 it/s	63.3%	88.9%
DiZO	✓	✗	9.5 GB	9.5 GB	1.54 it/s	60.0%	62.3%
MeZO LoRA	✓	✓	7.7 GB	7.7 GB	3.10 it/s	94.2%	51.6%
HiZOO LoRA	✓	✓	13.0 GB	13.0 GB	2.07 it/s	80.0%	65.7%
DiZO LoRA	✓	✓	9.4 GB	9.4 GB	2.87 it/s	66.7%	39.5%

In this section, we examine the memory utilization and convergence speed of DiZO in comparison with both ZO baselines and FO fine-tuning approaches (with and without LoRA). Table 5 presents the results of fine-tuning OPT-2.7B on the RTE dataset, more results are shown in Appendix E.2.

From a memory perspective, DiZO avoids backpropagation and memory-heavy activations, cutting memory use by 90% compared to FO. Its overhead stems only from storing *Query* and *Value* weights (16.7% of total). In contrast, HiZOO stores full-layer Hessians, scaling poorly with model size. In terms of convergence speed, DiZO significantly reduces iteration count while maintaining throughput comparable to MeZO, leading to much lower training GPU hours. By comparison, HiZOO achieves less iteration reduction and slows throughput of MeZO by about 1.5× due to Hessian estimation, resulting in only modest savings, or even higher training cost in some cases, such as HiZOO+LoRA on RTE

A notable byproduct of our method is using FO (e.g., with the Adam optimizer) to learn the projections. While this version has memory consumption comparable to LoRA and requires additional training GPU hours, it offers distinct advantages. Since DiZO does not update projections at every iteration, FO-based DiZO exhibits significantly lower average memory usage than FO-based LoRA, with an average memory overhead close to that of the ZO-based DiZO. Although average memory usage may seem less critical in single-process, single-GPU setup, many real-world on-device training scenarios involve multi-process environments [34, 35]. In such cases, the FO-based DiZO can stagger memory usage phases across processes, enabling parallel operations that purely FO methods cannot achieve. Furthermore, compared with ZO-based DiZO, the FO version reduces extra training GPU hours and delivers better performance. These qualities make it particularly appealing for specific on-device training cases.

6.5 Discussion on Potential Alternatives and Limitations

Adaptive learning rate methods may appear analogous to DiZO at first glance, as both introduce per-layer adaptive control over parameter updates, but their principles differ. Methods like Adam and RMSProp adjust update magnitude based on gradient history, controlling how fast parameters move. DiZO, by contrast, uses geometric constraints to guide parameters toward a learnable target distance from a fixed anchor (the pre-trained model), determining where the parameters move. This projection-based approach enables principled, divergence-aware updates that step-size modulation alone cannot replicate. Additionally, adaptive methods maintain gradient moment estimates, adding memory and computational overhead, particularly in LLMs. We empirically compare DiZO to these methods in Appendix D.

Line search could potentially be a simpler method to replace ZO for optimizing projection scalars. However, line search approaches, e.g., backtracking, generally require tuning each layer's scalar independently, leading to inefficient and unscalable coordinate-wise search. These methods also rely on directional derivatives and assume smooth interactions, which do not generalize well to joint tuning across layers. DiZO avoids these issues by using a ZO-based strategy that updates all scalars simultaneously, stabilized by projection clipping and re-initialization. As shown in Appendix D, replacing ZO with line search under the same forward-pass budget leads to worse performance, confirming its inefficiency in this context.

Limitations. While DiZO demonstrates notable improvements in both accuracy and training efficiency, the theoretical foundations behind its design choices remain incomplete. For example, the choice of using a pre-trained model as the anchor point and the selection of which layer to be projected is primarily supported by empirical observations rather than formal justification(detailed ablation study in Appendix C). The absence of a solid theoretical framework to explain why such design yields consistent performance gains leaves open questions about the optimality of the approach. Nevertheless, our findings offer valuable insights and point to promising directions for future research, particularly in developing anchor-guided, adaptive ZO optimization frameworks with stronger theoretical grounding.

7 Conclusion

In this paper, we propose a novel layer-wise divergence analysis to reveal the distinct update pattern between FO and ZO methods. Building on these insights, we present DiZO, an enhanced ZO method using divergence-driven layer adaptation to resemble the learning capacity of the FO method. DiZO achieves significant training acceleration and superior performance across diverse tasks and architectures. Moreover, our method can be seamlessly integrated with PEFT techniques like LoRA for additional speedup. For future work, we plan to explore DiZO in other fields, particularly for fine-tuning large pre-trained vision models.

8 Acknowledgment

This work was supported by the U.S. National Science Foundation, under Grant No. CCF-2553684, CNS-2312158, No. 1943046 and IIS-2223768.

References

[1] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*, 2019.

[2] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[3] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[5] Minghang Zheng, Peng Gao, Renrui Zhang, Kunchang Li, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *arXiv preprint arXiv:2011.09315*, 2020.

[6] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35:30016–30030, 2022.

[7] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[8] Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W Mahoney, and Kurt Keutzer. Ai and memory wall. *IEEE Micro*, 2024.

[9] Shulin Zeng, Jun Liu, Guohao Dai, Xinhao Yang, Tianyu Fu, Hongyi Wang, Wenheng Ma, Hanbo Sun, Shiyao Li, Zixiao Huang, et al. Flightllm: Efficient large language model inference with a complete mapping flow on fpgas. In *Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 223–234, 2024.

[10] Hongzheng Chen, Jiahao Zhang, Yixiao Du, Shaojie Xiang, Zichao Yue, Niansong Zhang, Yaohui Cai, and Zhiru Zhang. Understanding the potential of fpga-based spatial acceleration for large language model inference. *ACM Transactions on Reconfigurable Technology and Systems*, 2024.

[11] Qitao Tan, Sung-En Chang, Rui Xia, Huidong Ji, Chence Yang, Ci Zhang, Jun Liu, Zheng Zhan, Zhenman Fang, Zhou Zou, et al. Perturbation-efficient zeroth-order optimization for hardware-friendly on-device training. *arXiv preprint arXiv:2504.20314*, 2025.

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

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

[14] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

[15] Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor W Tsang. Second-order fine-tuning without pain for llms: A hessian informed zeroth-order optimizer. *arXiv preprint arXiv:2402.15173*, 2024.

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

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

[18] Astha Verma, Siddhesh Bangar, A Venkata Subramanyam, Naman Lal, Rajiv Ratn Shah, and Shin'ichi Satoh. Certified zeroth-order black-box defense with robust unet denoiser. *arXiv preprint arXiv:2304.06430*, 2023.

[19] Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. Model agnostic contrastive explanations for structured data. *arXiv preprint arXiv:1906.00117*, 2019.

[20] Xiaoxing Wang, Wenxuan Guo, Jianlin Su, Xiaokang Yang, and Junchi Yan. Zarts: On zero-order optimization for neural architecture search. *Advances in Neural Information Processing Systems*, 35:12868–12880, 2022.

[21] Jiaqi Gu, Chenghao Feng, Zheng Zhao, Zhoufeng Ying, Ray T Chen, and David Z Pan. Efficient on-chip learning for optical neural networks through power-aware sparse zeroth-order optimization. In *Proceedings of the AAAI conference on artificial intelligence*, pages 7583–7591, 2021.

[22] Aochuan Chen, Yimeng Zhang, Jinghan Jia, James Diffenderfer, Jiancheng Liu, Konstantinos Parasyris, Yihua Zhang, Zheng Zhang, Bhavya Kailkhura, and Sijia Liu. Deepzero: Scaling up zeroth-order optimization for deep model training. *arXiv preprint arXiv:2310.02025*, 2023.

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

[24] Haishan Ye, Zhichao Huang, Cong Fang, Chris Junchi Li, and Tong Zhang. Hessian-aware zeroth-order optimization for black-box adversarial attack. *arXiv preprint arXiv:1812.11377*, 2018.

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

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

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

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

[29] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.

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

[31] Rui Pan, Xiang Liu, Shizhe Diao, Renjie Pi, Jipeng Zhang, Chi Han, and Tong Zhang. Lisa: layerwise importance sampling for memory-efficient large language model fine-tuning. *Advances in Neural Information Processing Systems*, 37:57018–57049, 2024.

[32] Qijun Luo, Hengxu Yu, and Xiao Li. Badam: A memory efficient full parameter optimization method for large language models. *Advances in Neural Information Processing Systems*, 37:24926–24958, 2024.

[33] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

[34] Xiangyu Li, Yuanchun Li, Yuanzhe Li, Ting Cao, and Yunxin Liu. Flexnn: Efficient and adaptive dnn inference on memory-constrained edge devices. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 709–723, 2024.

[35] Shengyuan Ye, Liekang Zeng, Xiaowen Chu, Guoliang Xing, and Xu Chen. Asteroid: Resource-efficient hybrid pipeline parallelism for collaborative dnn training on heterogeneous edge devices. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pages 312–326, 2024.

[36] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[37] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[38] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[39] Jun Chen, Han Guo, Kai Yi, Boyang Li, and Mohamed Elhoseiny. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18030–18040, 2022.

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

[41] Mannat Singh, Laura Gustafson, Aaron Adcock, Vinicius de Freitas Reis, Bugra Gedik, Raj Prateek Kosaraju, Dhruv Mahajan, Ross Girshick, Piotr Dollár, and Laurens Van Der Maaten. Revisiting weakly supervised pre-training of visual perception models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 804–814, 2022.

[42] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[43] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024.

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

[45] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35:30318–30332, 2022.

[46] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

[47] Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*, 2023.

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

[49] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

[50] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Advances in neural information processing systems*, 31, 2018.

[51] Anirudh Vemula, Wen Sun, and J Bagnell. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2926–2935. PMLR, 2019.

[52] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

[53] Yao Shu, Zhongxiang Dai, Weicong Sng, Arun Verma, Patrick Jaillet, and Bryan Kian Hsiang Low. Zeroth-order optimization with trajectory-informed derivative estimation. In *The Eleventh International Conference on Learning Representations*, 2023.

[54] Ozan Sener and Vladlen Koltun. Learning to guide random search. *arXiv preprint arXiv:2004.12214*, 2020.

[55] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 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, 2013.

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

[57] Ellen M Voorhees and Dawn M Tice. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207, 2000.

[58] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pages 581–590. IEEE, 2020.

[59] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer, 2005.

[60] Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the second PASCAL challenges workshop on recognising textual entailment*, volume 1. Citeseer, 2006.

[61] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1, 2009.

[62] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9, 2007.

[63] Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, pages 107–124, 2019.

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

[65] Mohammad Taher Pilehvar and Jose Camacho-Collados. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*, 2018.

[66] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

[67] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 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, 2018.

[68] P Rajpurkar. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

[70] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

A Related Work

A.1 Memory-efficient Fine-tuning

Fine-tuning a pre-trained model offers a powerful way to reuse learned representations and reduce training costs compared to building models from scratch, often achieving superior performance [36, 37]. Initially successful in NLP with models like BERT, RoBERTa, and GPT [38, 2, 39], fine-tuning has also shown promise in vision tasks such as CLIP and SWAG [40, 41].

Despite the success of fine-tuning, its high cost makes it not feasible. Therefore memory efficient fine-tuning method come up. Recent parameter-efficient fine-tuning (PEFT), including LoRA [17], and prefix tuning [42], minimize resource needs by updating only a small subset of parameters, preserving most of the pre-trained weights and ensuring valuable knowledge is retained. Low-rank decomposition-based methods, led by LoRA, have achieved remarkable success. The main idea is to minimize resource needs by updating only a small subset of parameters, preserving most of the pre-trained weights and ensuring valuable knowledge is retained. DoRA [43] decomposes the pre-trained weight into two components, magnitude and direction, to enhance both the learning capacity and training stability of LoRA. GaLore [44] proposed gradient low-rank projection, allows full-parameter learning while retaining the memory advantages of low-rank training. Beside low-rank method, quantization stood out as a promising method to reduce resources utilization. GPT3.int8() [45] identified the outlier in activation, and include a new mixed-precision decomposition scheme, which isolates the outlier feature dimensions into a 16-bit matrix multiplication while still more than 99.9% of values are multiplied in 8-bit. Despite training with mix-precision, SmoothQuant [46] smooths the activation outliers by offline by scaling, migrating the quantization difficulty from activations to weights. Moreover, Outlier Suppression+ [47] and OmniQuant [48] apply channel-wise shifting for asymmetry and channel-wise scaling for concentration for int4-level quantization.

A.2 Zeroth-order Optimization and Acceleration

ZO optimization emerges as an attractive technique that optimizes the model without backpropagation [22, 49, 24, 18, 50, 19]. Unlike most frequently used FO optimization, which directly obtains and leverages the gradient for optimization, the zeroth-order method utilizes the objective function value oracle only, estimating the gradient by finite differences. ZO method has a wide range of applications in machine learning fields, including adversarial attack and defense [49, 24, 18], machine learning explainability [50, 19], reinforcement learning [51], and on-chip training [21]. Recently, the ZO method has been proposed to be leveraged on LLM fine-tuning to address the significant memory usage. [14] proposed MeZO, first scaling ZO optimization to fine-tuning parameter-intensive LLMs, greatly reducing memory utilization. On top of MeZO, [15] proposed HiZOO, leveraging the estimated Hessian information for better learning capacity, but reducing the throughput of MeZO to some extent.

ZO optimization, although it significantly saves memory, converges more slowly than FO methods due to higher variance from random search. [23] introduced ZO-SVRG by incorporating variance reduction techniques [52]. [53] proposed using a Gaussian process to model objective function queries, thereby reducing query complexity and allowing more frequent queries to lower gradient variance. [54] performed random search on a learned low-dimensional manifold, reducing the number of needed objective queries. However, existing ZO accelerators face two main challenges when adapting to ZO fine-tuning for LLMs. First, these approaches were typically designed for smaller-scale tasks involving fewer parameters and less data, and cannot be directly extended to large-scale LLMs. Second, many prior methods focus on improving query efficiency, whereas recent work has shown that a single query can suffice for LLM fine-tuning [14]. How to effectively accelerate ZO optimization on large model fine-tuning remains a problem.

Moreover, ZO has several properties that make it well-suited for on-device or edge training scenarios. 1) Memory efficiency: Edge devices such as mobile phones and FPGAs typically offer limited memory resources. ZO significantly reduces memory usage by avoiding activation and gradient storage, making it more deployable in such constrained setting. 2) Forward-only optimization: As ZO only relies on forward passes, it is compatible with existing inference accelerators (e.g., NNAPI on Android, edge TPUs, etc.), which typically lack support for backpropagation. This makes ZO a strong candidate for adapting inference-only hardware for training.

B Experiment Settings and Analysis

B.1 Datasets and Evaluation

Table B.1: The hyperparameter for experiments. For DiZO and DiZO LoRA, we only show the setting of extra hyperparameters, and have the same setting in other common hyperparameters with MeZO and MeZO LoRA respectively.

Experiment	Hyperparameters	Values
FT	Batch size	8
	Learning rate	{1e-5, 5e-5}
	Lr schedule	Constant for RoBERTa Linear for OPT and Llama
MeZO	Batch size	{64, 16}
	Learning rate η (Lr)	{1e-6, 5e-7}
	ϵ	1e-3
	Lr schedule	Constant for RoBERTa Linear for OPT and Llama
MeZO LoRA	Batch size	{64, 16}
	Learning rate η (Lr)	{1e-4, 5e-5}
	ϵ	1e-2
	Lr schedule	Constant for RoBERTa Linear for OPT and Llama
DiZO (LoRA)	Projection update cycle	{50, 100, 200, 400}
	Smoothing scalar $\epsilon^{'}$	{1e-1, 5e-2}
	Clip range τ	{0.1, 0.2, 0.3}

For the RoBERTa-large model, we use the following classification datasets: SST-2 [55], SST-5 [55], SNLI [56], TREC [57], MNLI [58], and RTE [59, 60, 61, 62]. Following previous studies, we cap the test set size at 1000 samples. Two training settings are considered: $k = 16$ and $k = 512$, where we randomly select 16 or 512 samples per class for both training and validation.

For the OPT and Llama series models, we use the SuperGLUE benchmark [27], which includes RTE [59, 60, 61, 62], CB [63], BoolQ [64], WIC [65], WSC [66], and MultiRC [67]. We also include SST-2 [55] and two question answering datasets, SQuAD [68] and DROP [69]. For each of these datasets, we randomly sample 1000 instances for training, 500 for validation, and 1000 for testing.

B.2 Hyperparameter Setting

We use the hyperparameters in Table B.1 for experiments on RoBERTa-large, OPT-series, and Llama-series models. Specifically, the choice of clip range did not significantly impact the performance. The selection of the projection update cycle and scalar for projection affects the performance somewhat. Generally, for datasets that need larger iterations for convergence, or for these harder datasets, DiZO prefers a larger update cycle, while for those less complicated datasets, DiZO benefits from a smaller update cycle.

C Ablation study on DiZO

C.1 Ablation for Projection Layers Selection

Instead of applying projections to all layers, which would require storing the entire pre-trained model, we focus only on projecting the weights of the *Query* and *Value* in the attention modules. As shown in Table C.1, this strategy achieves the best trade-off between the overall performance and extra storage requirements, does not reduce the performance and only 16.7% of the parameters of the pre-trained model are needed to store. A Similar strategy has also been adopted in LoRA [17].

Table C.1: Ablation study for selecting which layers to project. The highlighted line with a blue rectangle is the setting used in DiZO. Extra memory indicates the extra memory needed due to pre-trained model storing. Attn_Q: attention Query layer; Attn_V: attention Value layer; Attn_K: attention Key layer; Attn_O: attention output projection; Dense: dense fully connected layer.

Attn_Q	Attn_V	Attn_K	Attn_O	Dense	Extra memory	SST-2	RTE	SQuAD
✓	✓	✓	✓	✓	100%	91.7	68.4	67.3
✓	✓	✓	✓	✗	33.3%	92.2	67.9	69.2
✓	✓	✓	✗	✗	25.0%	91.9	67.1	68.1
✓	✓	✗	✗	✗	16.7%	92.5	68.2	69.0
✓	✗	✗	✗	✗	8.4%	90.5	64.9	66.5

C.2 Ablation for Strategies in ZO Projection Learning

As discussed in Section 3.3, we introduce two strategies, *re-initialization* (Re-init) and *projection clipping* (Clipping), to enhance projection learning and improve the stability of fine-tuning. The ablation results for these strategies, along with the corresponding loss curves, are shown in Figure C.1.

Overall (left in Figure C.1), omitting either Re-init or Clipping significantly diminishes the benefits of DiZO, with MeZO outperforming DiZO in these cases. Specifically, without Re-init, accuracy drops sharply, falling below MeZO. Similarly, without Clipping, while DiZO slightly outperforms MeZO on simpler datasets like SST-2, it suffers from severe model collapse on more challenging datasets, leading to a significant decline in accuracy.

From the loss curve trajectory (right in Figure C.1), without Re-init, DiZO loses its advantage in training acceleration, as the loss curve becomes noticeably slower to decrease. Without Clipping, the loss curve exhibits significant oscillations during certain training steps. This instability arises when projections are optimized to unsuitable values, such as extremely large or small magnitudes. These inappropriate projections cause substantial changes in model weights, leading to pronounced oscillations in the loss.

Type	Re-init	Clipping	SST-2	SNLI	TREC
MeZO	-	-	90.5	66.0	76.9
DiZO	✗	✓	88.6	64.2	73.8
	✓	✗	90.9	56.2	61.2
	✓	✓	92.2	71.6	77.4

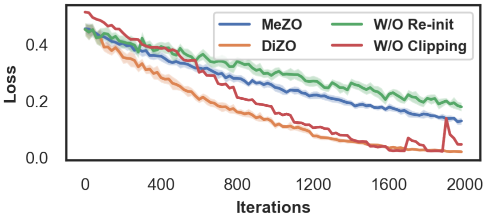

Figure C.1: Ablation study for the two strategies: re-initialization and projection clipping, which is conducted on RoBERTa-large ($k = 16$). Left: overall results when ablating the strategies. Right: loss curve when ablating the strategies.

C.3 Ablation for the Selection of the Anchor

We conduct an experiment to illustrate the effect of other anchor points beyond the pre-trained weights, the results are shown in Table C.2. In conclusion, using $\mathbf{0}$ or $\boldsymbol{\theta_{t-1}}$ significantly reducing the benefit of our method. Specifically, using $\mathbf{0}$ as an anchor yields similar results to using θ_{t-1} in terms of GPU hours, but causes unstable training, the accuracy decreases at the later training stage, and causes the results to be even worse than MeZO. Therefore, the effectiveness of our method is inseparable from the choice of anchors. Selecting a more robust anchor could not only improve accuracy but also the convergence speed.

C.4 Ablation on Hyperparameter Setting

We conduct experiments to investigate the effect of different hyperparameter settings, including different clip ranges, smoothing scalars, and projection update frequencies. Results are obtained by fine-tuning RoBERTa-large on SST-2 and SST-5 with 3 different settings for the 3 hyperparameters,

Table C.2: Comparison on conducting projection on learning rate (LR) or use weight at $(t-1)$-th iteration $\boldsymbol{\theta}_{t-1}$ instead of the weight of the pre-trained model $\boldsymbol{\theta_0}$ as the base of projection. Results are obtained by fine-tuning OPT-2.7B.

Anchor	**SST-2**		**RTE**		**SQuAD**	
	Acc.	GPU Hours	Acc.	GPU Hours	F1.	GPU Hours
NA (MeZO)	90.0	100.0%	63.5	100.0%	68.7	100.0%
$\mathbf{0}$	86.9	85.7%	58.4	91.0%	62.2	85.8%
$\boldsymbol{\theta}_{t-1}$ projection	90.7	87.8%	64.5	90.3%	67.2	88.4%
DiZO	92.5	55.7%	68.2	62.3%	69.0	65.4%

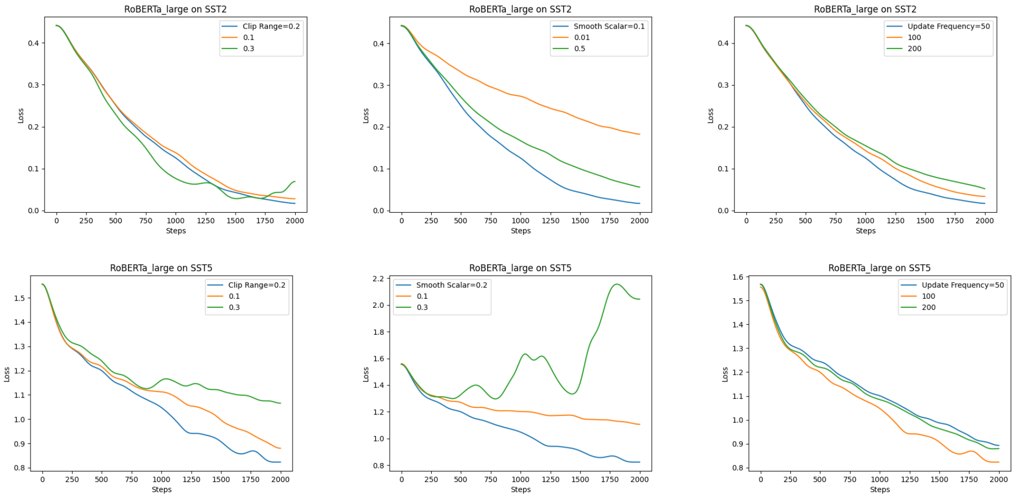

Figure C.2: Hyperparameter sensitive testing, including clip range, smooth scalar, and update frequency. Evaluated by training Roberta-large on SST2 and SST5 datasets.

as shown in Figure C.2. In conclusion, for easier classification tasks like SST-2, the weight of the model changes fast, and so as the convergence speed, therefore, we can apply more aggressive projection strategies, apply a larger clip range, a larger smoothing scalar, and update the projection more frequently. In contrast, for the rather more difficult tasks, a more conservative projection strategy is preferred.

C.5 Quantization of Anchor (Pre-trained) Model

Our method introduce a anchor model (i.e., the pre-trained model) for projection, roughly 16% of the pretrained parameters must still reside in memory. To further reduce extra memory cost, we explored anchor compression via quantization of the pretrained Query and Value matrices to 8-bit and 4-bit precision [48]. As shown in Table C.3, DiZO can effectively incorporate with the quantization technique, still preserving advantages in both accuracy and GPU hours. Exploring more advanced quantization or compression methods to further improve anchor efficiency while maintaining performance is an important avenue for future work.

D Do Other Alternative Strategies Work?

As discussed in Section 6.5, we compare with two representative straightforward alternative strategies, Adam and Backtracking linesearch, to highlight the effectiveness of our method.

Adam as a representative of learnable learning rate methods, Adam leverages first- and second-moment estimates of historical gradients to adaptively modulate the update magnitude. However, incorporating Adam into ZO optimization poses significant practical challenges. Storing gradient mo-

Table C.3: DiZO with quantized anchor.

Method	**SST-2**		**RTE**	
	Acc	**GPU hours**	**Acc**	**GPU hours**
MeZO	90.0	100%	63.5	100%
DiZO (8-bits)	92.2	63%	67.2	71%
DiZO (4-bits)	91.7	67%	65.2	68%
DiZO	92.5	56%	68.4	62%

Table D.1: Results on fine-tuning OPT-2.7B on SST2 when using Adam as the optimizer. Adam is either memory-intensive or introduces a lot of extra computational overhead.

Method	**Optimizer**	**Acc.**	**Training FLOPs**	**Memory**	**Iter.**
MeZO	SGD	85.2	100%	6.8 GB	2K
MeZO	Adam (Recompute)	86.1	431%	6.8 GB	2K
MeZO	Adam (Store)	86.1	100.2%	17.6GB	2K
DiZO	**SGD**	**86.3**	**61%**	7.5GB	1K
DiZO	SGD	89.8	122%	7.5GB	2K

ments leads to over 3× memory overhead compared to ZO-SGD, as shown in Table D.1. MeZO [14] attempts to address this by recomputing moment statistics on the fly rather than storing them. However, this strategy is also impractical as the total computational overhead increases quadratically with the training iterations, e.g., 4.3× more FLOPs for only 2K iterations. Given that zeroth-order methods often require tens of thousands of iterations to converge, both strategies render Adam either memory-intensive or computationally impractical at scale. In contrast, DiZO with SGD achieves higher accuracy with significantly lower FLOPs and comparable memory usage.

Line Search appears to be a simpler alternative for tuning the projection scalar. However, it requires freezing all other layers when optimizing the scalar for a particular layer, resulting in significant computational overhead. To empirically validate this limitation, we replace the ZO-based projection search in DiZO with Armijo-style backtracking line search. For fairness, the line search is initialized at $(1+\tau)$ × original learning rate, matching the scale used in DiZO. The results in Table D.2 show that, under the same forward pass budget, backtracking performs notably worse than the original MeZO. Furthermore, even when increasing the number of forward passes to 7800, it only matches MeZO's performance and still falls short of DiZO. These results confirm that traditional line search is not only less effective but also less efficient in the context of high-dimensional, layer-wise projection tuning.

E More Experimental Results

In this section, we provide a comprehensive presentation of our results across various datasets and models to complement the main paper. Specifically, the results include:

- Detailed accuracy number and trajectory of training loss on RoBERTa-large (Table E.1 and Figure E.1).

Table D.2: Results on fine-tuning OPT-2.7B on SST2. For fair comparison, we use Armijo-style backtracking line search to replace ZO in our method.

Searching Strategy	Forward Pass	Accuracy
NA (MeZO)	4000	85.2
Backtracking	4000	81.6
Backtracking	7800	85.1
ZO (DiZO)	4000	89.3

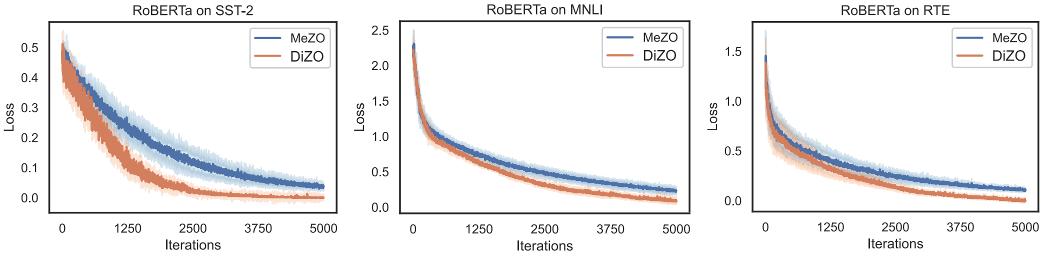

Figure E.1: Trajectory of training loss curves when using MeZO and DiZO to fine-tune Roberta-large.

- More memory and speed results of fine-tuning OPT-2.7B on SST-2 and SQuAD datasets (Table E.2 and Table E.3).
- Results on larger model, OPT-13B (Table E.4).
- Results on Llama3-3B and Llama-8B (Table E.5 and Table E.6).
- Results on more challenging benchmark MMLU and MT-Bench (Table E.7 and Table E.8).

E.1 RoBERTa-large Experiments

Table E.1 reports the corresponding detailed numbers from Figure 2, and Figure E.1 shows the trajectory of training loss.

Table E.1: Experiment results on RoBERTa-large on six datasets.

Dataset	**SST-2**	**SST-5**	**SNLI**	**MNLI**	**RTE**	**TREC**
Task Type	——-sentiment——-		————-language inference————-			–topic–
Zero-shot	79.0	35.5	50.2	48.8	51.4	32.0
Gradient-free methods: $k = 16$						
MeZO	90.5 (1.2)	45.5 (2.0)	68.5 (3.9)	56.5 (2.5)	59.4 (5.3)	76.9 (2.7)
MeZO LoRA	91.4 (0.9)	43.0 (1.6)	69.7 (6.0)	64.0 (2.5)	64.9 (3.6)	73.1 (6.5)
HiZOO	91.1 (1.6)	46.1 (1.3)	69.3 (3.1)	58.8 (3.1)	57.4 (6.2)	77.2 (2.2)
HiZOO LoRA	91.4 (0.9)	44.8 (1.5)	70.7 (5.2)	64.3 (2.8)	66.9 (3.2)	72.7 (7.3)
DiZO	**92.2** (0.9)	**47.1** (1.3)	71.0 (3.1)	60.1 (3.5)	**67.9** (4.7)	**77.4** (2.4)
DiZO LoRA	91.7 (0.8)	44.6 (1.7)	**71.6** (3.8)	**65.6** (2.8)	67.3 (3.9)	74.6 (4.3)
Gradient-based methods: $k = 16$						
FT	91.9 (1.8)	47.5 (1.9)	77.5 (2.6)	70.0 (2.3)	66.4 (7.2)	85.0 (2.5)
FT LoRA	91.4 (1.7)	46.7 (1.1)	74.9 (4.3)	67.7 (1.4)	66.1 (3.5)	82.7 (4.1)
Gradient-free methods: $k = 512$						
MeZO	93.3 (0.7)	53.2 (1.4)	83.0 (1.0)	78.3 (0.5)	78.6 (2.0)	94.3 (1.3)
MeZO LoRA	93.4 (0.4)	52.4 (0.8)	84.0 (0.8)	77.9 (0.6)	77.6 (1.3)	95.0 (0.7)
HiZOO	93.5 (0.4)	53.5 (1.2)	83.3 (1.4)	77.2 (1.5)	79.1 (1.2)	94.9 (1.7)
HiZOO LoRA	94.3 (0.5)	54.1 (0.6)	82.7 (1.8)	79.0 (0.8)	**80.9** (1.6)	93.1 (0.5)
DiZO	**94.6** (0.1)	53.6 (1.7)	**84.5** (0.6)	**79.8** (0.9)	80.3 (1.8)	93.8 (1.5)
DiZO LoRA	94.3 (0.3)	**54.1** (1.4)	83.7 (1.1)	77.6 (0.4)	79.3 (1.4)	**95.7** (0.9)
Gradient-based methods: $k = 512$						
FT	93.9 (0.7)	55.9 (0.9)	88.7 (0.8)	84.4 (0.8)	82.7 (1.4)	97.3 (0.2)
FT LoRA	94.2 (0.2)	55.3 (0.7)	88.3 (0.5)	83.9 (0.6)	83.2 (1.3)	97.0 (0.3)

E.2 More Memory and Speed Analysis

We present the memory and speed results for OPT-2.7B on the SST-2 and SQuAD datasets in Table E.2 and Table E.3, respectively. DiZO significantly reduces the number of required iterations while maintaining throughput comparable to MeZO, leading to substantially fewer training GPU hours. In contrast, HiZOO achieves only modest iteration savings and further reduces the throughput of MeZO by approximately 1.5× due to its reliance on second-order information estimation. As a result, HiZOO offers only a slight improvement over MeZO in terms of training GPU hours. In some

Table E.2: Memory utilization and speed test on OPT-2.7B on SST-2 dataset. ●: partial gradient-free; ✓: gradient-free; ✗: gradient-based. DiZO†: searching projection with Adam.

Task Type	**Gradient Free**	**LoRA Added**	**Peak Memory**	**Averaged Memory**	**Throughput**	**#Train Iter.**	**GPU Hours**
FT	✗	✗	45.4 GB	45.4 GB	1.81 it/s	9.3%	16.8%
LoRA	✗	✓	18.4 GB	18.4 GB	4.50 it/s	5.6%	4.3%
DiZO† (w. FO)	●	✗	17.8 GB	15.7 GB	2.63 it/s	33.3%	41.5%
DiZO LoRA†	●	✓	16.1 GB	14.7 GB	4.16 it/s	22.2%	17.5%
MeZO	✓	✗	6.8 GB	6.8 GB	3.28 it/s	100.0%	100.0%
HiZOO	✓	✗	11.8 GB	11.8 GB	2.22 it/s	59.2%	87.4%
DiZO	✓	✗	7.5 GB	7.5 GB	3.05 it/s	51.8%	55.7%
MeZO LoRA	✓	✓	6.5 GB	6.5 GB	5.56 it/s	74.1%	43.7%
HiZOO LoRA	✓	✓	11.5 GB	11.5 GB	3.70 it/s	46.3%	41.0%
DiZO LoRA	✓	✓	7.2 GB	7.2 GB	4.92 it/s	38.9%	25.9%

Table E.3: Memory utilization and speed test on OPT-2.7B on SQuAD dataset (300 tokens per example on average). ●: partial gradient-free; ✓: gradient-free; ✗: gradient-based. DiZO†: searching projection with Adam.

Task Type	**Gradient Free**	**LoRA Added**	**Peak Memory**	**Averaged Memory**	**Throughput**	**#Train Iter.**	**GPU Hours**
FT	✗	✗	73.5 GB	73.5 GB	0.36 it/s	7.5%	27.7%
LoRA	✗	✓	58.5 GB	58.5 GB	0.73 it/s	6.3%	11.5%
DiZO†	●	✗	57.8 GB	20.3 GB	1.22 it/s	41.7%	45.5%
DiZO LoRA†	●	✓	49.4 GB	19.9 GB	2.44 it/s	31.7%	17.3%
MeZO	✓	✗	8.4 GB	8.4 GB	1.33 it/s	100.0%	100.0%
HiZOO	✓	✗	12.3 GB	13.3 GB	0.97 it/s	66.7%	91.5%
DiZO	✓	✗	9.7 GB	9.7 GB	1.22 it/s	60.0%	65.4%
MeZO LoRA	✓	✓	8.4 GB	8.4 GB	2.80 it/s	73.3%	34.8%
HiZOO LoRA	✓	✓	11.6 GB	12.6 GB	2.10 it/s	56.7%	35.9%
DiZO LoRA	✓	✓	9.6 GB	9.6 GB	2.49 it/s	45.0%	24.0%

cases, such as HiZOO combined with LoRA on SQuAD, it even consumes more training GPU hours than MeZO with LoRA.

E.3 Larger OPT Models Fine-tuning

To further illustrate the generalizability of our method, we conduct experiments on OPT-13B. The results are shown in Table E.4, DiZO consistently outperforms the baselines both in terms of accuracy and speed.

E.4 Llama Fine-tuning

To demonstrate the generalizability of DiZO, we conducted experiments on the Llama-series models. The results for Llama3-3B and Llama3-8B are presented in Table E.5 and Table E.6, respectively. DiZO consistently outperforms MeZO across both the 3B and 8B Llama models.

Table E.4: Experiment results on OPT-13B (with 1000 training samples). Better results are highlighted in bold.

Dataset	SST-2		RTE		SQuAD	
	Acc	GPU hours	Acc	GPU hours	Acc	GPU hours
MeZO	91.4	100%	66.1	100%	84.7	100%
HiZOO	92.1	86%	69.3	82%	82.9	91%
DiZO	**92.4**	**69%**	**72.6**	**76%**	**85.2**	**73%**

However, we observed that ZO LoRA performs poorly with Llama models (including DiZO, MeZO and HiZOO). The loss value remains stagnant, and the resulting accuracy is comparable to or even worse than zero-shot results. We leave it to future work to investigate why ZO LoRA fails with Llama models. We suspect that this limitation may be related to the Group Query Attention (GQA) [70] mechanism employed in Llama3.

Table E.5: Experimental results on Llama3-3B for seven classification datasets and two text generation datasets (with 1000 training samples). Better results between MeZO and DiZO are highlighted in bold.

Task	**SST-2**	**RTE**	**CB**	**BoolQ**	**WSC**	**WIC**	**MultiRC**	**SQuAD**	**DROP**
Task Type		————————classification————————						——generation——	
FT	94.2 (0.4)	81.2 (2.1)	91.4 (4.7)	72.2 (4.2)	63.8 (1.8)	65.8 (2.3)	78.2 (3.2)	79.6 (2.9)	40.3 (1.2)
MeZO	88.8 (1.1)	67.4 (1.7)	73.2 (2.4)	78.0 (4.4)	56.6 (3.8)	63.4 (2.3)	**64.8** (3.1)	61.9 (2.7)	27.8 (2.0)
HiZOO	89.5 (1.4)	67.1 (1.3)	74.4 (1.9)	**78.8** (4.7)	56.3 (3.1)	**64.4** (2.4)	64.3 (2.9)	61.7 (2.8)	28.6 (3.0)
DiZO	**90.0** (0.9)	**68.2** (1.6)	**76.7** (3.3)	76.8 (3.8)	**57.8** (4.2)	63.8 (1.7)	64.2 (2.9)	**63.2** (2.7)	**29.7** (1.3)

Table E.6: Experiments results on Llama3-8B for seven classification datasets and two text generation datasets (with 1000 training samples). Better results between MeZO and DiZO are highlighted in bold.

Task	**SST-2**	**RTE**	**CB**	**WSC**	**SQuAD**
Task Type		————-classification————-			–generation–
MeZO	90.0 (0.7)	67.8 (1.4)	71.4 (2.2)	60.2 (1.6)	67.0 (2.6)
HiZOO	91.1 (0.9)	68.2 (1.3)	71.4 (2.9)	62.2 (1.9)	**68.3** (3.1)
DiZO	**91.5** (0.8)	**69.4** (1.8)	**73.2** (3.1)	**63.4** (2.9)	67.4 (2.1)

E.5 Fine-tuning on MMLU and MT-Bench

To demonstrate the generalizability of DiZO in more realistic and challenging scenarios, we evaluate our method on MMLU and MT-Bench benchmarks. we follow the setting in [31, 32], fine-tune on the Alpaca GPT-4 dataset [33], which consists of 52k conversations, and then evaluate. We conduct experiments based on Llama2-7B and Llama3-8B, the results are shown in Table E.7 and Table E.8, respectively.

E.6 Compare with Sparse Technique

We conducted a direct comparison between our DiZO and Sparse MeZO [13] in terms of both accuracy and training efficiency across two datasets and two model sizes. The results are presented in Table R.6 and Table R.7. From the accuracy perspective, DiZO consistently outperforms Sparse MeZO under all evaluated settings, demonstrating the effectiveness of our projection-based approach.

From the efficiency perspective, Sparse MeZO requires generating the sparsity mask dynamically during training. Compared to DiZO, Sparse MeZO requires longer GPU hours for about 20%, and slows the throughput for more than 30%. Moreover, with the grows of model size, the throughput of Sparse MeZO will further decrease, due to the growing cost of maintaining and updating the sparse

Table E.7: Results of fine-tuning Llama2-7B on more challenging benchmarks, better results are highlighted in bold.

	MT-Bench	MMLU (5 shot)	GPU hours
Zero-shot	3.93	45.87	-
MeZO	4.59	45.22	100%
HiZOO	4.62	45.42	92%
DiZO	**4.79**	**45.91**	**78%**

Table E.8: Results of fine-tuning Llama3-8B on more challenging benchmarks, better results are highlighted in bold.

	MT-Bench	MMLU (5 shot)	GPU hours
Zero-shot	5.46	65.20	-
MeZO	5.89	65.08	100%
HiZOO	5.93	65.20	95%
DiZO	**6.15**	**65.42**	**83%**

mask. While Sparse MeZO reduces the number of training iterations, its lower throughput results in longer total GPU hours compared to DiZO. These findings demonstrate that DiZO not only delivers better accuracy but also achieves more practical training efficiency compared to Sparse MeZO.

Table E.9: Acc and Speed Comparison on OPT-2.7B.

Method	**Dataset**	**Accuracy**	**Throughput**	**#Train Iter.**	**GPU Hours**
MeZO	SST2	90.0	3.3it/s	100%	100%
Sparse MeZO	SST2	91.4	2.3it/s	55%	79%
DiZO	SST2	92.3	1.9it/s	52%	56%
MeZO	RTE	63.5	1.7it/s	100%	100%
Sparse MeZO	RTE	67.1	1.1it/s	50%	73%
DiZO	RTE	68.4	1.5it/s	60%	62%

Table E.10: Acc and Speed Comparison on OPT-6.7B.

Method	**Dataset**	**Accuracy**	**Throughput**	**#Train Iter.**	**GPU Hours**
MeZO	SST2	90.2	1.8it/s	100%	100%
Sparse MeZO	SST2	91.9	1.0it/s	47%	84%
DiZO	SST2	92.4	1.7it/s	62%	69%
MeZO	RTE	73.2	0.6it/s	100%	100%
Sparse MeZO	RTE	74.3	0.3it/s	39%	88%
DiZO	RTE	74.8	0.5it/s	65%	81%

F Theoretical Analysis

F.1 Variance Symmetry under Isotropic Perturbations

We consider a neural network with L layers (or parameter blocks) and analyze the zeroth-order gradient estimator constructed via two-point finite differences. Let $\mathcal{L}(\boldsymbol{\theta};\mathcal{B})$ denote the loss evaluated on mini-batch $\mathcal{B}$, with $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(L)})$ the full parameter vector. To estimate the gradient $\nabla\mathcal{L}$ without access to derivatives, we apply a two-sided estimator along randomly sampled directions $\boldsymbol{u}_i \in \mathbb{R}^d$:

$$\widehat{\nabla_{\boldsymbol{\theta}^{(\ell)}}\mathcal{L}} \;=\; \frac{1}{q}\sum_{i=1}^{q} \underbrace{\frac{\mathcal{L}(\boldsymbol{\theta}+\epsilon\boldsymbol{u}_i) - \mathcal{L}(\boldsymbol{\theta}-\epsilon\boldsymbol{u}_i)}{2\epsilon}}_{\Delta_i} \,\boldsymbol{u}_i^{(\ell)},$$

where $\boldsymbol{u}_i^{(\ell)}$ is the sub-vector of direction $\boldsymbol{u}_i$ corresponding to layer ℓ.

We aim to characterize the variance of this estimator, in particular:

$$\mathbb{E}\left[\left\|\widehat{\nabla_{\boldsymbol{\theta}^{(\ell)}}\mathcal{L}}\right\|^2\right].$$

Key assumptions:

1. Each $\boldsymbol{u}_i$ is drawn independently from an *isotropic* distribution in $\mathbb{R}^d$, i.e., $\mathbb{E}[\boldsymbol{u}_i\boldsymbol{u}_i^\top] = I$.
2. The scalar Δ_i is the same across all parameter blocks for a given i, as it depends only on the global perturbation.
3. Each block $\boldsymbol{u}_i^{(\ell)}$ has zero mean, unit covariance in its own subspace $\mathbb{R}^{d_\ell}$, and is uncorrelated with other blocks $\boldsymbol{u}_i^{(m)}$ for $\ell \neq m$.

Variance Expansion: We examine the norm-squared of the estimator:

$$\left\|\widehat{\nabla_{\boldsymbol{\theta}^{(\ell)}}\mathcal{L}}\right\|^2 = \left\|\frac{1}{q}\sum_{i=1}^{q}\Delta_i\boldsymbol{u}_i^{(\ell)}\right\|^2.$$

Taking expectation over $\{\boldsymbol{u}_i\}$:

$$\mathbb{E}\left[\left\|\widehat{\nabla_{\boldsymbol{\theta}^{(\ell)}}\mathcal{L}}\right\|^2\right] = \frac{1}{q^2}\sum_{i=1}^{q}\mathbb{E}\left[\Delta_i^2\|\boldsymbol{u}_i^{(\ell)}\|^2\right] + \frac{1}{q^2}\sum_{i\neq j}\mathbb{E}\left[\Delta_i\Delta_j\langle\boldsymbol{u}_i^{(\ell)}, \boldsymbol{u}_j^{(\ell)}\rangle\right].$$

Due to independence and zero-mean isotropy, cross terms vanish, and we obtain:

$$\mathbb{E}\left[\left\|\widehat{\nabla_{\boldsymbol{\theta}^{(\ell)}}\mathcal{L}}\right\|^2\right] = \frac{1}{q}\mathbb{E}\left[\Delta^2\cdot\|\boldsymbol{u}^{(\ell)}\|^2\right],$$

where Δ and $\boldsymbol{u}^{(\ell)}$ are representative samples from the same distribution.

Conclusion. Since Δ is shared across layers and $\boldsymbol{u}^{(\ell)}$ has expected squared norm proportional to the layer dimension d_ℓ, we conclude:

$$\mathbb{E}\left[\left\|\widehat{\nabla_{\boldsymbol{\theta}^{(\ell)}}\mathcal{L}}\right\|^2\right] \propto d_\ell.$$

In other words, the second-moment of the gradient estimator depends on the layer only through its dimensionality d_ℓ, and not through any asymmetry in the distribution of direction vectors. If d_ℓ are the same for all ℓ, each block exhibits identical expected variance.

This property justifies using uniform per-layer treatment in analysis and initialization when random direction sampling is isotropic.

F.2 The Proof of Convergence Analysis

We now prove Theorem 1.

Proof. By L_f-smoothness (Assumption 5.1), for any $\boldsymbol{\theta}, \boldsymbol{\theta}'$:

$$\mathcal{L}(\boldsymbol{\theta}') \;\le\; \mathcal{L}(\boldsymbol{\theta}) \;+\; \langle \nabla\mathcal{L}(\boldsymbol{\theta}),\, \boldsymbol{\theta}' - \boldsymbol{\theta} \rangle \;+\; \frac{L_f}{2}\,\|\boldsymbol{\theta}' - \boldsymbol{\theta}\|^2.$$

Set $\boldsymbol{\theta} = \boldsymbol{\theta}_t$ and $\boldsymbol{\theta}' = \boldsymbol{\theta}_{t+1}$. We get:

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) \;\le\; \mathcal{L}(\boldsymbol{\theta}_t) \;+\; \big\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \big\rangle \;+\; \frac{L_f}{2}\,\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|^2.$$

Taking conditional expectation $\mathbb{E}_t[\cdot] := \mathbb{E}[\cdot \mid \boldsymbol{\theta}_t]$ yields

$$\mathbb{E}_t\big[\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\boldsymbol{\theta}_t)\big] \;\le\; \mathbb{E}_t\big[\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \rangle\big] \;+\; \frac{L_f}{2}\,\mathbb{E}_t\big[\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|^2\big]. \tag{8}$$

Denote $\widetilde{\boldsymbol{\theta}}_{t+1} = \boldsymbol{\theta}_t - \eta\, g_t$. Then

$$\boldsymbol{\theta}_{t+1} \;=\; \mathrm{Proj}_{\mathcal{S}}(\widetilde{\boldsymbol{\theta}}_{t+1}), \qquad \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \;=\; (\boldsymbol{\theta}_{t+1} - \widetilde{\boldsymbol{\theta}}_{t+1}) \;-\; \eta\, g_t.$$

Hence

$$\big\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \big\rangle \;=\; \big\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, \boldsymbol{\theta}_{t+1} - \widetilde{\boldsymbol{\theta}}_{t+1} \big\rangle \;-\; \eta\,\big\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, g_t \big\rangle.$$

Because $\boldsymbol{\theta}_{t+1}$ is the *nearest point* in $\mathcal{S}$ to $\widetilde{\boldsymbol{\theta}}_{t+1}$ (by definition of projection), we have

$$\|\boldsymbol{\theta}_{t+1} - \widetilde{\boldsymbol{\theta}}_{t+1}\| \;\le\; \|\boldsymbol{\theta}_t - \widetilde{\boldsymbol{\theta}}_{t+1}\| \;=\; \eta\,\|g_t\|.$$

Thus

$$\|\boldsymbol{\theta}_{t+1} - \widetilde{\boldsymbol{\theta}}_{t+1}\| \;\le\; \eta\,\|g_t\|. \tag{9}$$

Taking conditional expectation:

$$\mathbb{E}_t\big[\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \rangle\big] \;=\; \mathbb{E}_t\big[\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, \boldsymbol{\theta}_{t+1} - \widetilde{\boldsymbol{\theta}}_{t+1} \rangle\big] \;-\; \eta\,\mathbb{E}_t\big[\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, g_t \rangle\big].$$

Using (9) with Cauchy–Schwarz:

$$\big|\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, \boldsymbol{\theta}_{t+1} - \widetilde{\boldsymbol{\theta}}_{t+1} \rangle\big| \;\le\; \|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|\; \|\boldsymbol{\theta}_{t+1} - \widetilde{\boldsymbol{\theta}}_{t+1}\| \;\le\; \eta\,\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|\; \|g_t\|.$$

Moreover, the two-point estimator is *unbiased*, so

$$\mathbb{E}_t\big[\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, g_t \rangle\big] \;=\; \langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, \mathbb{E}_t[g_t] \rangle \;=\; \|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2.$$

Hence

$$\mathbb{E}_t\big[\langle \nabla\mathcal{L}(\boldsymbol{\theta}_t),\, \boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t \rangle\big] \;\le\; \eta\,\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|\; \mathbb{E}_t\big[\|g_t\|\big] \;-\; \eta\,\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2. \tag{10}$$

Again from the projection property:

$$\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\| \;=\; \|\mathrm{Proj}_{\mathcal{S}}(\widetilde{\boldsymbol{\theta}}_{t+1}) - \boldsymbol{\theta}_t\| \;\le\; \|\widetilde{\boldsymbol{\theta}}_{t+1} - \boldsymbol{\theta}_t\| \;=\; \eta\,\|g_t\|.$$

Thus

$$\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|^2 \;\le\; \eta^2\,\|g_t\|^2.$$

Taking expectation completes:

$$\mathbb{E}_t\big[\|\boldsymbol{\theta}_{t+1} - \boldsymbol{\theta}_t\|^2\big] \;\le\; \eta^2\,\mathbb{E}_t\big[\|g_t\|^2\big].$$

Substitute (10) and the above bound into (8):

$$\mathbb{E}_t\big[\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\boldsymbol{\theta}_t)\big] \;\le\; \eta\,\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|\,\mathbb{E}_t[\|g_t\|] \;-\; \eta\,\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2 \;+\; \frac{L_f}{2}\,\eta^2\,\mathbb{E}_t[\|g_t\|^2].$$

Taking total expectation and summing over $t = 0$ to $T - 1$,

$$\mathbb{E}\big[\mathcal{L}(\boldsymbol{\theta}_T)\big] \;-\; \mathcal{L}(\boldsymbol{\theta}_0) \;\le\; \sum_{t=0}^{T-1} \Big\{ \eta\,\mathbb{E}\big[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|\; \|g_t\|\big] \;-\; \eta\,\mathbb{E}\big[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2\big] \;+\; \tfrac{L_f}{2}\,\eta^2\,\mathbb{E}\big[\|g_t\|^2\big] \Big\}.$$

Assume that the two-point finite-difference estimator satisfies

$$\mathbb{E}\big[\|g_t - \nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2\big] \;\le\; \sigma^2\big(D, q\big),$$

where $\sigma^2(D, q)$ grows primarily with D (the maximum layer dimension) and the number of queries q, rather than the full sum of dimensions.

Therefore, we have

$$\mathbb{E}\big[\|g_t\|^2\big] \;\le\; c_1\,\big(\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2 + \sigma^2(D, q)\big)$$

for some constant $c_1 > 0$. In addition, $\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|\;\|g_t\|] \;\le\; \sqrt{\mathbb{E}[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2]\,\mathbb{E}[\|g_t\|^2]} \;\le\; c_2\,(\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2 + \sigma^2(D, q))$ (for a constant c_2). Collecting terms and choosing $\eta = c/\sqrt{T}$ with $c > 0$ sufficiently small ensures that the $-\eta\,\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2$ term dominates the positive terms for large T. A standard telescoping argument shows

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2\big] \;=\; O\Big(\frac{\sqrt{D}}{\sqrt{T}}\Big),$$

which implies the stationarity measure

$$\min_{0\le t<T}\mathbb{E}\Big[\|\nabla\mathcal{L}(\boldsymbol{\theta}_t)\|^2\Big] \;=\; O\Big(\frac{\sqrt{D}}{\sqrt{T}}\Big).$$

This completes the proof. □

F.3 τ-Stability of the Clipping Step

Assumptions.

A1. $L : \mathbb{R}^d \to \mathbb{R}$ is L_f-smooth.

A2. Zeroth-order updates use step-size $\eta_t = \eta$ and produce an intermediate point $\theta_t^{\ell} = \theta_t - \eta g_t$ with an unbiased estimator g_t.

A3. After every k iterations we apply projection clipping:

$$\theta_{t+1}^{(\ell)} = \theta_t^{(0)} + \rho_t^{(\ell)}\Delta\theta_t^{(\ell)}, \quad \rho_t^{(\ell)} := \mathrm{clip}\big[\rho_t^{(\ell)},\, 1-\tau,\, 1+\tau\big],$$

where $\rho_t^{(\ell)} = \gamma_t^{(\ell)}/\|\Delta\theta_t^{(\ell)}\|_2$, and $\tau \in (0, 1)$ is the scalar clipping width.

Let

$$R_{\max} := \max_{t,\ell}\|\Delta\theta_t^{(\ell)}\|_2, \qquad G_{\max} := \max_t\|g_t\|_2.$$

Then for any τ, we have the stability bound:

$$\|\theta_{t+1} - \theta_t^{\ell}\|_2 \le \eta G_{\max} + \tau R_{\max}. \tag{11}$$

Moreover, if τ satisfies

$$\tau \le c_\tau\frac{\eta G_{\max}}{R_{\max}}, \quad 0 < c_\tau \le 1, \tag{12}$$

then under the same conditions as Theorem 5.3, the projected DiZO iterates satisfy

$$\min_{0\le t<T}\mathbb{E}\big[\|\nabla L(\theta_t)\|_2^2\big] = \mathcal{O}\left(\frac{\sqrt{D}}{\sqrt{T}}\right). \tag{13}$$

That is, the original non-convex convergence rate is preserved. If τ violates (12), the bound inflates linearly with τ, i.e.,

$$\widetilde{\mathcal{O}}\big(\sqrt{D/T} + \tau/\eta\big).$$

Proof. For any L_f-smooth loss, we have

$$L(\theta_{t+1}) \leq L(\theta_t) + \langle \nabla L(\theta_t), \theta_{t+1} - \theta_t \rangle + \frac{L_f}{2} \|\theta_{t+1} - \theta_t\|_2^2. \tag{14}$$

Because θ_{t+1} is the nearest feasible point to θ_t^ℓ (Euclidean projection), the proof in the paper shows

$$\|\theta_t - \theta_t^\ell\|_2 \leq \eta \|g_t\|_2. \tag{15}$$

The additional clipping in $\rho_t^{(\ell)}$ imposes another bound:

$$\|\theta_{t+1}^{(\ell)} - \theta_t^{(\ell)}\|_2 = |\rho_t^{(\ell)} - 1| \|\Delta\theta_t^{(\ell)}\|_2 \leq \tau R_{\max}, \tag{16}$$

hence combining gives

$$\|\theta_{t+1} - \theta_t\|_2 \leq \eta G_{\max} + \tau R_{\max}. \tag{17}$$

Plugging (17) into (14) and taking conditional expectation yields

$$\mathbb{E}_t[L(\theta_{t+1}) - L(\theta_t)] \leq -\eta \mathbb{E}_t[\langle \nabla L(\theta_t), g_t \rangle] + L_f(\eta G_{\max} + \tau R_{\max}) \eta G_{\max}.$$

Using standard ZO analysis, the inner product term satisfies

$$-\frac{\eta}{2} \|\nabla L(\theta_t)\|_2^2 + \frac{\eta}{2} \sigma^2,$$

and after telescoping the left-hand side for $t = 0, \ldots, T-1$, dividing by ηT, we obtain

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[\|\nabla L(\theta_t)\|_2^2\big] \leq \mathcal{O}\left(\frac{\sqrt{D}}{\sqrt{T}}\right) + \mathcal{O}\left(\frac{\tau R_{\max}}{\eta}\right).$$

If τ satisfies condition (12), the second term is dominated, yielding (13).

Remark. Equation (12) implies that the projection impulse matches a single ZO step:

$$\tau \approx \frac{\eta \|g_t\|_2}{\|\Delta\theta_t\|_2}.$$

During fine-tuning, we typically observe $\|g_t\|_2 / \|\Delta\theta_t\|_2 \in [0.5, 1.5]$ after warm-up, and learning rates $\eta \sim 10^{-2}$–10^{-1}. This leads to τ in the range 0.05–0.30, with $\tau \approx 0.2$ being the empirically stable choice across tasks.

G Implementation

The following is an implementation of our “ZO projection learning” in PyTorch.

```python
def ZO_Projection_Learning(theta_t, theta_0, Gammas, delta, eta, tau, x):
    """
    Perform Zeroth-order Projection Learning.

    Args:
        theta_t: Current model parameters to be fine-tuned.
        theta_0: Pre-trained model parameters (anchor).
        Gammas: Projection parameters need to be optimized.
        delta: Smoothing parameter.
        eta: Learning rate for projection gradient descent.
        tau: Clipping factor for projection bounds.
        x: Input data for the forward pass.
    """

    # Calculate the L2 norm of the distance gap
    norms = {
        name: torch.norm(param.data - anchor.data)
        for (name, param), anchor in zip(theta_t.named_parameters(),
            theta_0.parameters())
    }

    # Initialize the projection values
    for name, gamma in Gammas.named_parameters():
        gamma.data = norms[name]

    for i in range(max_iters):
        # Step 1: Perturb and apply projection, then compute loss
        Gammas = PerturbGamma(Gammas, delta)
        ApplyProjection(theta_t, pre_trained, Gammas)
        loss1 = Forward(theta_t, x)
        ReverseProjection(theta_t)  # Reset the parameter before
            projection

        # Step 2: Reverse and apply projection, then compute loss
        Gammas = PerturbGamma(Gammas, -2 * delta)
        ApplyProjection(theta_t, pre_trained, Gammas)
        loss2 = Forward(theta_t, x)
        ReverseProjection(theta_t)  # Reset the parameter before
            projection

        # Step 3: Reset projection and compute gradient
        Gammas = PerturbGamma(Gammas, delta)  # Reset projection
        grad = (loss1 - loss2) / (2 * delta)

        # Step 4: Gradient descent with clipping
        for name, gamma in Gammas.named_parameters():
            torch.manual_seed(seed)  # For resampling perturbation
            z = torch.normal(mean=0, std=1, size=gamma.data.size())
            gamma.data = torch.clip(
                gamma.data - eta * grad * z,
                (1 - tau) * norms[name],
                (1 + tau) * norms[name],
            )  # Conduct descent and apply clipping

    return Gammas
```

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the abstract and introduction, we claimed our contribution is to devise a method to improve the performance of ZO on LLM.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations and future work are discussed in Section 6.5 and conclusion.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We have our proof both in Section 5 and Appendix F.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We present our pseudocode and PyTorch-style implementation in the paper and release our code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release our code with the anonymized url.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We present our experiment setting in both the experimental section and the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the standard deviation of the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the computing resources we use, and carefully analyze the computational overhead of our method.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes, we do.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, we have cited all the necessary works.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes, we have released all the assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No human-related parts are included in this paper.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Not related.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Not related.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.