Zeroth-Order Fine-Tuning of LLMs in Random Subspaces

Ziming Yu[1] , Pan Zhou[2] , Sike Wang[1] , Jia Li[1,4†], Mi Tian[3] , Hua Huang[1,4]
[1] Beijing Normal University [2] Singapore Management University [3] TAL Education Group
[4] Engineering Research Center of Intelligent Technology and Educational Application (MOE)
{zimingyu, sikewang}@mail.bnu.edu.cn, {jiali, huahuang}@bnu.edu.cn,
panzhou@smu.edu.sg, tianmi@tal.com

Abstract

Fine-tuning Large Language Models (LLMs) has proven effective for a variety of downstream tasks. However, as LLMs grow in size, the memory demands for backpropagation become increasingly prohibitive. Zeroth-order (ZO) optimization methods offer a memory-efficient alternative by using forward passes to estimate gradients, but the variance of gradient estimates typically scales linearly with the model's parameter dimension—a significant issue for LLMs. In this paper, we propose the random Subspace Zeroth-order (SubZero) optimization to address the challenges posed by LLMs' high dimensionality. We introduce a low-rank perturbation tailored for LLMs that significantly reduces memory consumption while improving performance. Additionally, we prove that our gradient estimation closely approximates the backpropagation gradient, exhibits lower variance than traditional ZO methods, and ensures convergence when combined with SGD. Experimental results show that SubZero enhances fine-tuning performance and achieves faster convergence compared to standard ZO approaches like MeZO across various language modeling tasks. Code is available at https://github.com/zimingyy/SubZero.

1. Introduction

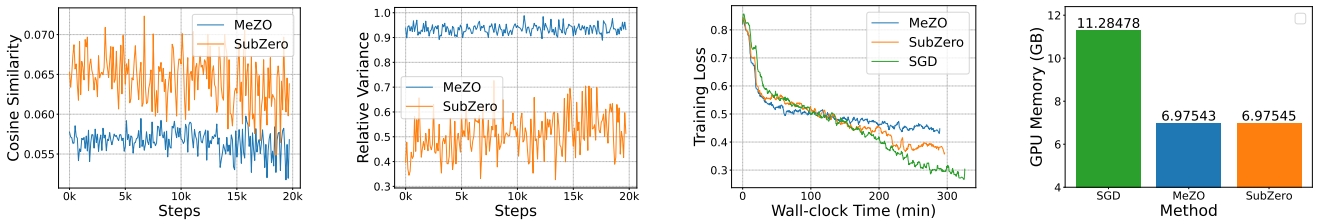
Large Language Models (LLMs), such as the GPT and LLaMA series [55, 64], have recently demonstrated impressive capabilities in natural language processing tasks and beyond [1, 52]. These models utilize deep learning, particularly the transformer architecture [56], to learn complex patterns in language data. However, LLMs can struggle with specialized tasks that require domain-specific knowledge [49]. Fine-tuning presents an effective solution by slightly adjusting pre-trained LLMs with domain data, enabling them to adapt to specific tasks more effectively.

For fine-tuning, first-order (FO) optimizers, such as SGD [3] or Adam [30], are commonly used to achieve promising performance on domain datasets. However, as LLMs grow in size, FO optimizers demand increasingly memory consumption due to the gradient computations required by backpropagation (BP) [67]. Additionally, they are unable to directly handle non-differentiable objectives. To enhance memory efficiency, MeZO [40] first introduces the zeroth-order (ZO) optimizer to LLM fine-tuning without BP. It only requires forward passes and calculates gradient estimates using finite differences of training loss values, enabling it to directly handle non-differentiable objectives. Nevertheless, the variance of ZO gradient estimates scales linearly with the perturbation dimension, which corresponds to the number of model parameters. This can become extremely large in LLMs, leading to significant performance degradation compared to FO optimizers [20, 28, 39].

Reducing the variance of ZO gradient estimates generally results in faster convergence [62]. There are two main attempts to addressing the high variance of ZO gradient estimates. The first approach involves increasing batch size alongside training steps, which reduces gradient noise and variance in ZO gradient estimates [20, 28]. However, this leads to significant runtime and memory costs due to the large batch size in the later training stages. The second approach focuses on perturbing fewer parameters by employing sparse parameter perturbations, such as random and sparse pruning masks [39] and block-coordinate perturbations [65], or by reducing the number of trainable parameters through techniques like parameter-efficient fine-tuning (PEFT) [40, 65] and tensorized adapters [60]. Recent theoretical advancements have proposed using random projections to lessen the dimensionality dependence in ZO optimizers [31, 43, 46] by applying low-dimensional perturbations in random subspaces. Nonetheless, a major drawback of this approach is the need to store a huge projection matrix that scales with model parameter dimensionality, making it impractical for fine-tuning large LLMs.

Contributions. In this work, we propose the first random Subspace Zeroth-order (SubZero) optimization to tackle the challenges of high-dimensional LLM fine-tuning. We intro-

†Corresponding Author



(a) Cosine Similarity (b) Relative Variance (c) Training Loss (d) Memory Cost

Figure 1. Visualization of cosine similarity $\mathbb{E}\left[\texttt{cosine}(\boldsymbol{g}, \hat{\boldsymbol{g}})\right]$, relative variance $\text{Var}\left[\|\hat{\boldsymbol{g}}\|\right] / \|\boldsymbol{g}\|^2$, training loss, and peak total GPU memory cost with OPT-1.3B on SST-2 in the prompt tuning scheme. All three methods utilize a batch size of 16 and run for 20K steps. Here, $\hat{\boldsymbol{g}}$ represents the gradient estimated by MeZO or our SubZero, and $\boldsymbol{g}$ denotes the expected gradient $\mathbb{E}[\hat{\boldsymbol{g}}]$. Theorem 1 (b) ensures that SubZero maintains a small distance between $\boldsymbol{g}$ and the BP gradient in a subspace. (a) and (b) demonstrate that SubZero's estimated gradient $\hat{\boldsymbol{g}}$ has lower angle error and variance than MeZO. (c) and (d) indicate that SubZero enhances convergence speed with minimal extra memory usage.

duce a low-rank perturbation to estimate the gradient, specifically designed for LLM architecture, leading to reduced memory consumption and enhanced training performance. Our main contributions are as follows.

Firstly, we propose a layer-wise low-rank perturbation approach for gradient estimation, specifically designed for fine-tuning LLMs. In each layer, we generate a low-rank perturbation matrix by combining two column-orthogonal matrices with a Gaussian random matrix, which is then used for gradient estimation. Unlike traditional ZO methods like MeZO [40] which apply non-low-rank perturbations to the entire model, our approach significantly reduces the variance of gradient estimates and the angle error between the estimated gradient and its expectation, as respectively shown in Fig. 1 (a) and (b). SubZero also improves upon random subspace ZO methods like S-RGF [43] by using smaller and layer-specific low-rank perturbation matrices instead of a large and model-scale projection matrix, thus cutting memory and computational costs. Additionally, we introduce a lazy update strategy, generating perturbations periodically rather than iteratively, further reducing overhead. Besides, we successfully apply SubZero to four popular LLM fine-tuning schemes, highlighting the compatibility of SubZero.

Secondly, we provide theoretical guarantees for SubZero. We first convert our gradient estimation into an equivalent formulation, highlighting the key differences between our approach and existing traditional ZO methods [40], as well as random subspace ZO methods [43]. Then, we prove that the gradient estimated by SubZero closely approximates the BP gradient, i.e., the ground-truth gradient, and enjoys significantly lower gradient variance than traditional ZO methods like MeZO. Furthermore, we establish the theoretical convergence of SubZero when combined with the SGD optimizer.

Finally, experimental results demonstrate SubZero's superior performance and memory efficiency compared to other ZO approaches in both full-parameter tuning and parameter-efficient fine-tuning (PEFT) schemes, such as LoRA, prefix tuning, and prompt tuning. For instance, SubZero improves upon MeZO by 7.1% on LLaMA-7B and by 3.2% on OPT-1.3B under full-parameter tuning and prompt tuning, while maintaining nearly identical memory costs to MeZO.

2. Related Work

Zeroth-Order Fine-Tuning. ZO optimizers utilize just two forward passes to estimate gradient without BP. Malladi et al. [40] first used ZO optimization to fine-tune LLMs, significantly lowering the GPU hours and memory usage to levels similar to inference, which offers a considerable advantage over FO optimizers. They demonstrated that LLM fine-tuning benefits from a well-structured loss landscape by introducing suitable task-specific prompt templates. Convergence theories for ZO optimization have been elaborated in both convex [19, 25, 41] and non-convex settings [26, 37]. However, these convergence rates typically increase linearly with the number of trainable parameters [19, 25, 26, 37, 41].

Recently, more work in ZO has focused on improving the convergence rates and reducing gradient estimation variance for LLM fine-tuning. Increasing batch size can diminish noise in ZO gradient estimation [20, 28]. Perturbing a subset of model parameters also lowers gradient variance. This approach induces sparse parameter perturbations through random and sparse pruning masks [39] or block-coordinate perturbations [65]. Additionally, some approaches tried to reduce trainable parameters through PEFT [40, 65] and tensorized adapters [60].

Random Subspace Optimization. To lessen dependence on dimensionality, some research utilizes random projections and low-dimensional perturbations in subspaces [31, 43, 46]. However, these methods are hindered by the need to store a large projection matrix that increases with dimensionality, making it impractical for fine-tuning LLMs.

Memory-Efficient Fine-Tuning. Fine-tuning generally employs FO optimizers like SGD [3] or Adam [30]. Various approaches have been developed to reduce the memory cost of BP, such as LoRA [24, 58], gradient sparsification [54], low-rank gradient projection [67], and optimizer state quantization [15, 35]. Additional methods to conserve activation and weight memory during forward and backward passes include gradient checkpointing [8], FlashAttention [12], QLoRA [16], and LLM.int8() [14].

3. Preliminaries

Here we introduce the most popular ZO optimization approach and existing random subspace optimization methods.
Notations. Let non-bold letter like a and A denote a scalar, a boldfaced lower-case letter like $\boldsymbol{w}$ denote a column vector, and a boldfaced upper-case letter like $\boldsymbol{W}$ denote a matrix. $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is a multivariate normal distribution with a zero mean vector and an identity covariance matrix. $\text{vec}(\boldsymbol{W})$ denotes the vectorization of matrix $\boldsymbol{W}$ which reshapes $\boldsymbol{W}$ into a column vector by stacking the columns of $\boldsymbol{W}$ vertically. $\boldsymbol{A} \otimes \boldsymbol{B}$ is the Kronecker product of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. $\mathbb{E}[\boldsymbol{x}]$ is the expected value of a random variable $\boldsymbol{x}$. $\text{Var}[\boldsymbol{x}]$ is the variance of a random variable $\boldsymbol{x}$. The ℓ_2-norm of a vector $\boldsymbol{x}$ is $\|\boldsymbol{x}\| = \sqrt{\sum_{i=1}^{n} \boldsymbol{x}_i^2}$. The spectral norm of a matrix $\boldsymbol{A}$ is $\|\boldsymbol{A}\|$. The Frobenius norm of a matrix $\boldsymbol{A}$ is $\|\boldsymbol{A}\|_F = \sqrt{\langle \boldsymbol{A}, \boldsymbol{A} \rangle}$. $C_L^{s,p}(\mathcal{S})$ denotes the class of s-th smooth and p-th L-smooth functions over the set $\mathcal{S}$. $\text{bdiag}(\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_l)$ is a block diagonal matrix with diagonal blocks $\boldsymbol{A}_1, \boldsymbol{A}_2, \cdots, \boldsymbol{A}_l$.

We are interested in fine-tuning large LLMs [17]. These models typically comprise multiple layers, with trainable parameter vectors represented as $\boldsymbol{w} = \left[\boldsymbol{w}_1^\top, \boldsymbol{w}_2^\top, \ldots, \boldsymbol{w}_l^\top\right]^\top \in \mathbb{R}^d$, where $\boldsymbol{w}_i$ denotes the flattened parameter vector from the i-th layer and d is model parameter dimension. Then training these models involves optimizing the problem:

$$\min_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}), \tag{1}$$

where $\mathcal{L}(\cdot)$ denotes the loss function.
Zeroth-Order Optimization. ZO optimization is BP-free and estimates gradients via random perturbations. A classical gradient estimator is the simultaneous perturbation stochastic approximation (SPSA) [53], which is defined as

$$\widehat{\nabla}\mathcal{L}(\boldsymbol{w}; \mathcal{B}) = \frac{\mathcal{L}(\boldsymbol{w} + \varepsilon \boldsymbol{z}; \mathcal{B}) - \mathcal{L}(\boldsymbol{w} - \varepsilon \boldsymbol{z}; \mathcal{B})}{2\varepsilon} \boldsymbol{z}, \tag{2}$$

where $\mathcal{L}(\boldsymbol{w}; \mathcal{B})$ is the loss on a minibatch $\mathcal{B}$ of size B uniformly sampled from the training dataset $\mathcal{D}$, $\boldsymbol{z} \in \mathbb{R}^d$ represents a random perturbation sampled from $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$, and ε is the perturbation scale.

The SPSA in Eqn. (2) is an unbiased gradient estimator of the desired gradient $\nabla \mathbb{E}_{\boldsymbol{z}}[\mathcal{L}(\boldsymbol{w} + \varepsilon \boldsymbol{z})]$ [41]. It only requires two forward passes to estimate the gradient and eliminates BP computation, greatly reducing computation cost and GPU memory. With this estimated gradient, one can integrate with existing FO optimizers like SGD to develop corresponding ZO optimizers, e.g., ZO-SGD defined as:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \eta^t \widehat{\nabla}\mathcal{L}(\boldsymbol{w}^t; \mathcal{B}^t), \tag{3}$$

where η^t is the learning rate at step t. In practice, MeZO [40] implements ZO-SGD via in-place operations and uses a single random seed to facilitate efficient perturbation regeneration, greatly reducing memory overhead.

Random Subspace Optimization. Recent theoretical work [43, 46] has explored using low-dimensional perturbations in random subspaces to reduce gradient variances and hence enhance convergence rates. The key to random subspace methods is the generation of the perturbation vector $\tilde{\boldsymbol{z}}$ within a subspace spanned by $\boldsymbol{P}$:

$$\tilde{\boldsymbol{z}} = \boldsymbol{P}\boldsymbol{z}, \tag{4}$$

where $\boldsymbol{P} \in \mathbb{R}^{d \times q}$ is a random projection matrix with entries drawn from $\mathcal{N}(0, 1)$, $\boldsymbol{z} \in \mathbb{R}^q$ is a low-dimensional random perturbation vector sampled from $\mathcal{N}(\mathbf{0}, \boldsymbol{I}_q)$, and $q < d$ is the dimension of the subspace. Thus, the gradient estimator in the subspace is given as follows:

$$\widehat{\nabla}\mathcal{L}(\boldsymbol{w}, \boldsymbol{P}; \mathcal{B}) = \frac{\mathcal{L}(\boldsymbol{w} + \varepsilon \boldsymbol{P}\boldsymbol{z}; \mathcal{B}) - \mathcal{L}(\boldsymbol{w} - \varepsilon \boldsymbol{P}\boldsymbol{z}; \mathcal{B})}{2\varepsilon} \boldsymbol{P}\boldsymbol{z}. \tag{5}$$

LLMs have a large model size, and thus their training and fine-tuning parameters can be very high-dimensional. This results in an excessively large matrix $\boldsymbol{P}$ which is q times larger than the model size d in full-parameter tuning [2] and is also large in other fine-tuning schemes e.g., LoRA [24]. Consequently, this approach significantly increases memory requirements and computational complexity. Therefore, it is crucial to develop an efficient subspace construction strategy with minimal memory consumption for LLM fine-tuning.

4. Methodology

Here we first elaborate on our SubZero, a powerful ZO framework for LLM fine-tuning. Then we present how to integrate SubZero into four representative fine-tuning schemes.

4.1. Random Subspace Optimization for LLM Fine-Tuning

Our intuition is that exploring update directions in a low-dimensional subspace may result in a reduced variance of the estimated gradient [43, 46] compared to the estimation in the vanilla space as used in MeZO. Moreover, recent work indicates that BP gradients in LLM fine-tuning rapidly converge to a small subspace [23, 40, 66, 67]. Accordingly, we propose the random Subspace Zeroth-order (SubZero) optimization framework tailored for LLM fine-tuning. This framework reduces gradient estimation variance, and minimizes the memory overhead associated with gradient estimation, such as the memory overhead caused by the projection matrix $\boldsymbol{P}$ in Eqn. (5) used in [43, 46].
Layer-wise Random Subspace Perturbation. LLMs primarily consist of dense layers that perform matrix multiplication. We denote the trainable parameters of the i-th layer in matrix form as $\boldsymbol{W}_i \in \mathbb{R}^{m_i \times n_i}$. Then we will explain how to design its low-rank perturbation $\tilde{\boldsymbol{Z}}_i \in \mathbb{R}^{m_i \times n_i}$.

We propose a low-rank perturbation strategy for model parameter matrix of each layer, contrasting with previous random subspace methods that focus on the entire

model's parameters [43, 46]. At each step, we generate a low-dimensional random matrix $\boldsymbol{Z}_i \in \mathbb{R}^{r\times r}$, where $r \ll \min\{m_i, n_i\}$, and perform QR decomposition on two Gaussian random matrices with entries sampled from $\mathcal{N}(0,1)$ to create projection matrices $\boldsymbol{U}_i \in \mathbb{R}^{m_i\times r}$ and $\boldsymbol{V}_i \in \mathbb{R}^{n_i\times r}$ (see Algorithm 1). Both $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ are column-orthogonal matrices. Our experiments in Table 5 indicate that directly using Gaussian random projection matrices yields worse performance than using our designed column-orthogonal matrices. Then we combine these three matrices to yield a low-rank perturbation as follows:

$$\tilde{\boldsymbol{Z}}_i = \boldsymbol{U}_i \boldsymbol{Z}_i \boldsymbol{V}_i^{\mathsf{T}}, \tag{6}$$

where $\tilde{\boldsymbol{Z}}_i$ is the perturbation matrix in a subspace spanned by $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$, and $\boldsymbol{Z}_i$ represents the low-dimensional random perturbation matrix with entries sampled from $\mathcal{N}(0,1)$. The projection matrices for SubZero can only be derived using the QR decomposition of the weights, activations, ZO gradients, or random matrix, without increasing memory overhead. As shown in Table 1, our proposed random matrix achieves the best performance. While the weight and ZO gradient matrices are feasible, their performance drops significantly. The activation matrix is ineffective due to its batch size dependency, requiring more sophisticated handling. Detailed experimental setups are in Appendix 8.4.

Let the model consist of l layers, with the parameter matrix set defined as $\mathcal{W} = \{\boldsymbol{W}_i\}_{i=1}^{l}$ and the perturbation matrix set as $\tilde{\mathcal{Z}} = \{\tilde{\boldsymbol{Z}}_i\}_{i=1}^{l}$. Similar to Eqns. (2) and (5), we compute the loss difference:

$$\rho = \frac{\mathcal{L}(\mathcal{W} + \varepsilon\tilde{\mathcal{Z}}; \mathcal{B}) - \mathcal{L}(\mathcal{W} - \varepsilon\tilde{\mathcal{Z}}; \mathcal{B})}{2\varepsilon}. \tag{7}$$

Note that multiplying a set by a scalar means that the scalar is multiplied by each element in the set. The addition of two sets means that the corresponding elements are added. This is only for mathematical expression, and ρ in Eqn. (7) can be calculated by two forward passes through all the layers in practice. Then we obtain the gradient estimate for the i-th layer as

$$\widehat{\nabla}\mathcal{L}(\boldsymbol{W}_i; \mathcal{B}) = \rho\tilde{\boldsymbol{Z}}_i = \rho\boldsymbol{U}_i\boldsymbol{Z}_i\boldsymbol{V}_i^{\mathsf{T}}. \tag{8}$$

In Sec. 5, we analyze the effectiveness of this new gradient estimation (8). Specifically, Theorem 1 proves the close distance between our gradient estimate (8) and the vanilla gradient computed by BP in FO methods, while Theorem 2 shows smaller variance and angle error of our gradient estimate in Eqn. (8) compared to the gradient estimate (2) in MeZO [40]. See more theoretical details in Sec. 5.

Then, one can use estimated gradient in (8) to replace the gradient in any FO optimizer such as SGD:

$$\boldsymbol{W}_i^{t+1} = \boldsymbol{W}_i^t - \eta^t \widehat{\nabla}\mathcal{L}(\boldsymbol{W}_i^t; \mathcal{B}^t) = \boldsymbol{W}_i^t - \eta^t\rho^t\boldsymbol{U}_i^t\boldsymbol{Z}_i^t{\boldsymbol{V}_i^t}^{\mathsf{T}}. \tag{9}$$

Table 1. Projection matrix generation for SubZero in full-parameter tuning with OPT-1.3B and LLaMA2-7B on SST-2 and OPT-13B on RTE.

Matrix	1.3B	7B	13B
Weight	91.5	91.7	65.3
Activation	51.5	52.9	53.1
ZO Gradient	89.6	92.0	67.5
Random	**93.4**	**94.5**	**74.0**

Table 2. Memory cost in full-parameter tuning with RoBERTa-large on SST-2.

Method	Mem.(GB)
SGD	6.063
MeZO [40]	2.683
S-RGF [43]	23.845
SubZero	2.690

Here we choose SGD as the default optimizer of SubZero. Theorem 3 in Sec. 5 guarantees the convergence of SubZero with SGD as basic optimizer and gives its convergence rate. The choice of FO optimizers is orthogonal to ZO optimization. Also, some empirical work indicates that adaptive optimizers like Adam [30] do not necessarily enhance convergence of ZO approaches during LLM fine-tuning [22, 65]. So the combination of SubZero and Adam is included in Appendix 8.1 due to the limited space. We apply the primitive ZO approach. There are other ZO optimizers that utilize stochastic momentum [28] and second-order information [68] to facilitate faster convergence. While SubZero can be adapted to these ZO optimizers, we leave a comprehensive evaluation of these approaches for future work.

We compare the memory overhead of SubZero with the existing random subspace method S-RGF [43] using identical experimental settings, including layer-wise perturbation and matching subspace dimension, with all methods utilizing the SGD optimizer. As shown in Table 2, S-RGF's memory usage is roughly four times greater than SGD and 8.8 times that of MeZO [40], while our SubZero's memory usage is comparable to MeZO. See more experimental comparisons on OPT-13B in Table 10 of Appendix 8.1.

Lazy Low-rank Subspace Update. According to Eqn. (9), at the t-th step, the gradient estimate of the parameter matrix in the i-th layer, $\widehat{\nabla}\mathcal{L}(\boldsymbol{W}_i^t; \mathcal{B}^t)$, lies within a subspace defined by the projection matrices $\boldsymbol{U}_i^t$ and $\boldsymbol{V}_i^t$. Specifically, $\boldsymbol{U}_i^t$ spans the column subspace, while $\boldsymbol{V}_i^t$ determines the row subspace, with both matrices generated iteratively, leading to extra computational overhead to LLM fine-tuning.

However, for LLM fine-tuning, enhancing the computational efficiency and the accuracy of gradient subspace approximation is crucial. An excessively short update interval for $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$, such as generating them iteratively, can incur high computational costs and limit exploration of the gradient subspace they established. Conversely, a long interval may result in inaccuracies in subspace approximation and fail to capture the evolving nature of the gradient subspace. Accordingly, we propose a lazy subspace update strategy that periodically regenerates the projection matrices $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$. Specifically, these matrices are generated at the first step of every $F > 1$ training steps and remain unchanged for the subsequent $F - 1$ steps (see lines 4-7 in Algorithm 3). We utilize QR decomposition on two different random matrices for generating the column-orthogonal matrices $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$,

as summarized in Algorithm 1. This lazy subspace update strategy is both efficient and effective in all our experiments.

Algorithm 1 GenerateProjMatrix(m, n, r)

Input: size of parameter matrix $m \times n$, rank r.

Generate random matrices $\boldsymbol{R}_1 \in \mathbb{R}^{m\times r}$ and $\boldsymbol{R}_2 \in \mathbb{R}^{n\times r}$ whose entries are sampled from $\mathcal{N}(0,1)$
$\boldsymbol{U}, \_ \leftarrow$ QR_Decomposition$(\boldsymbol{R}_1)$
$\boldsymbol{V}, \_ \leftarrow$ QR_Decomposition$(\boldsymbol{R}_2)$
return $\boldsymbol{U}, \boldsymbol{V}$

Algorithm 2 PerturbParams$(\mathcal{W}, \mathcal{U}, \mathcal{V}, r, \varepsilon, s)$

Input: model parameter set $\mathcal{W}$, projection matrix sets $\mathcal{U}$ and $\mathcal{V}$, rank r, perturbation scale ε, seed s.

Reset random number generator with seed s
for $i = 1, 2, \ldots, l$ **do**
  Generate the perturbation matrix $\boldsymbol{Z}_i \in \mathbb{R}^{r\times r}$ whose entries are sampled from $\mathcal{N}(0,1)$
  $\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i + \varepsilon \boldsymbol{U}_i \boldsymbol{Z}_i \boldsymbol{V}_i^\mathsf{T}$
return $\mathcal{W}$

Algorithm 3 SubZero

Input: parameter matrix in the i-th layer $\boldsymbol{W}_i \in \mathbb{R}^{m_i\times n_i}, i = 1, 2, \ldots, l$, loss $\mathcal{L}$, step budget T, perturbation scale ε, learning rate schedule $\{\eta^t\}$, subspace change frequency F, rank r.

for $t = 0, 1, \ldots, T-1$ **do**
  Sample a minbatch $\mathcal{B}^t \subset \mathcal{D}$ and a random seed s^t
  for $i = 1, 2, \ldots, l$ **do**
    if $t \mod F \equiv 0$ **then**
      $\boldsymbol{U}_i^t, \boldsymbol{V}_i^t \leftarrow$ GenerateProjMatrix(m_i, n_i, r)
    else
      $\boldsymbol{U}_i^t \leftarrow \boldsymbol{U}_i^{t-1}, \boldsymbol{V}_i^t \leftarrow \boldsymbol{V}_i^{t-1}$
  // Note that $\mathcal{W}^t = \{\boldsymbol{W}_i^t\}_{i=1}^l, \mathcal{U}^t = \{\boldsymbol{U}_i^t\}_{i=1}^l$, $\mathcal{V}^t = \{\boldsymbol{V}_i^t\}_{i=1}^l$
  $\mathcal{W}^t \leftarrow$ PerturbParams $(\mathcal{W}^t, \mathcal{U}^t, \mathcal{V}^t, r, \varepsilon, s^t)$, $\ell_+^t \leftarrow \mathcal{L}(\mathcal{W}^t; \mathcal{B}^t)$
  $\mathcal{W}^t \leftarrow$ PerturbParams $(\mathcal{W}^t, \mathcal{U}^t, \mathcal{V}^t, r, -2\varepsilon, s^t)$, $\ell_-^t \leftarrow \mathcal{L}(\mathcal{W}^t; \mathcal{B}^t)$
  $\mathcal{W}^t \leftarrow$ PerturbParams $(\mathcal{W}^t, \mathcal{U}^t, \mathcal{V}^t, r, \varepsilon, s^t)$
  $\rho^t \leftarrow \left(\ell_+^t - \ell_-^t\right)/(2\varepsilon)$
  Reset random number generator with seed s^t
  for $i = 1, 2, \ldots, l$ **do**
    Regenerate the perturbation matrix $\boldsymbol{Z}_i^t \in \mathbb{R}^{r\times r}$ whose entries are sampled from $\mathcal{N}(0,1)$
    $\boldsymbol{W}_i^{t+1} \leftarrow \boldsymbol{W}_i^t - \eta^t \rho^t \left(\boldsymbol{U}_i^t \boldsymbol{Z}_i^t \boldsymbol{V}_i^{t\mathsf{T}}\right)$
return $\mathcal{W}^{t+1}$

SubZero maintains just three small matrices per layer: a perturbation matrix $\boldsymbol{Z}_i \in \mathbb{R}^{r\times r}$, and two column-orthogonal matrices $\boldsymbol{U}_i \in \mathbb{R}^{m_i\times r}$ and $\boldsymbol{V}_i \in \mathbb{R}^{n_i\times r}$. This design enhances memory efficiency, as r is generally much smaller than the size of the corresponding parameter matrix $\boldsymbol{W}_i \in \mathbb{R}^{m_i\times n_i}$ (i.e., $r \ll \min\{m_i, n_i\}$). Moreover, we employ in-place operations and per-layer parameter updates to estimate gradients and update parameters in parallel (see Appendix 8.2). Consequently, SubZero uses significantly less GPU memory than previous methods while achieving similar or better performance. For example, fine-tuning OPT-1.3B [64] on SST-2 [51] using SGD (without momentum) in full-parameter scheme as shown in Table 4, SubZero requires only 6.8GB GPU memory, compared to 11.5GB for SGD, yielding a 1.6× improvement in memory efficiency, similar as illustrated in Fig. 1 (d).

Now we are ready to summarize the overall algorithm of SubZero in Algorithm 3. Each training step consists of three sequential phases. First, it obtains the projection matrices $\boldsymbol{U}_i^t$ and $\boldsymbol{V}_i^t$ using Algorithm 1 or directly adopts previous ones. Next, it computes the loss value difference ρ with Eqn. (7) by applying Algorithm 2 to perturb all parameter matrices. Finally, SubZero updates all parameter matrices layer by layer, following Eqn. (9).

4.2. Integration into Fine-Tuning Schemes

We describe the integration of SubZero into full-parameter tuning [2] and three prominent PEFT schemes: LoRA [24], prefix tuning [36], and prompt tuning [33]. Typically, SubZero can be easily incorporated into these fine-tuning schemes. However, it encounters a challenge with extremely non-square parameter matrices, which have far more rows than columns or vice versa. This issue is particularly prevalent in LoRA, which employs two low-rank matrices $\boldsymbol{A}_i \in \mathbb{R}^{m_i\times k}$ and $\boldsymbol{B}_i \in \mathbb{R}^{k\times n_i}$ to approximate a full matrix $\boldsymbol{W}_i' \in \mathbb{R}^{m_i\times n_i}$, with $k \ll \min\{m_i, n_i\}$, e.g., $k = 8$ while $\min\{m_i, n_i\} = 2048$ used in [65]. Consequently, it is impossible to find a smaller rank $r \ll k$ to compute the gradient estimates of $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ using Eqn. (6), imposing a challenge when applying SubZero to this scenario.

To overcome this limitation, we propose a reshaping strategy that transforms the original non-square matrix into an approximate square matrix. For instance, we reshape $\boldsymbol{A}_i \in \mathbb{R}^{m_i\times k}$ into $\boldsymbol{A}_i' \in \mathbb{R}^{m_i'\times k'}$ such that $m_i k = m_i' k'$ and m_i' is close to k'. This reshaping allows us to apply Eqn. (6) to find a low-rank perturbation with rank r significantly smaller than $\min\{m_i', k'\}$, demonstrating the applicability of SubZero in the scenario. Table 8 in Sec. 6.4 shows the effectiveness of this reshaping strategy.

5. Theoretical Analysis

In this section, we theoretically analyze why SubZero can reduce the variance of gradient estimates and hence accelerate convergence. Before the analysis, we first define some necessary notations:

$$\boldsymbol{P} = \mathrm{bdiag}(\boldsymbol{V}_1 \otimes \boldsymbol{U}_1, \cdots, \boldsymbol{V}_l \otimes \boldsymbol{U}_l), \tag{10}$$

$$\boldsymbol{z} = [\mathrm{vec}(\boldsymbol{Z}_1)^\mathsf{T}, \ldots, \mathrm{vec}(\boldsymbol{Z}_l)^\mathsf{T}]^\mathsf{T}, \tag{11}$$

$$\tilde{\boldsymbol{z}} = [\mathrm{vec}(\tilde{\boldsymbol{Z}}_1)^\top, \ldots, \mathrm{vec}(\tilde{\boldsymbol{Z}}_l)^\top]^\top. \quad (12)$$

Then we first state the main theoretical results on our gradient estimation in Eqn. (8).

Theorem 1. *For the gradient estimation in Eqn. (8), the following two properties hold.*
a) *By using gradient estimation in (8), our estimated gradient $\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})$ is equivalent to*

$$\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z}) = \frac{f(\boldsymbol{x} + \varepsilon \boldsymbol{P}\boldsymbol{z}) - f(\boldsymbol{x} - \varepsilon \boldsymbol{P}\boldsymbol{z})}{2\varepsilon} \boldsymbol{P}\boldsymbol{z}, \quad (13)$$

where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$, $\varepsilon > 0$, $\boldsymbol{P} \in \mathbb{R}^{d \times q}$ satisfies $\boldsymbol{P}^\top \boldsymbol{P} = \boldsymbol{I}_q$ with $d = \sum_{i=1}^{l} m_i n_i$ and $q = lr^2$.
b) *Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$, and $f \in C_{L_2}^{2,2}(\mathbb{R}^d)$. Then we have*

$$\Phi(\boldsymbol{x}) = \|\mathbb{E}_{\boldsymbol{z}}[\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] - \boldsymbol{P}\boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|_2 \leq \frac{\varepsilon^2}{6} L_2 (q+4)^2.$$

See its proof in Appendix 8.6. Theorem 1 (a) provides the equivalent form (13) of our gradient estimation (8). By comparing this with the gradient estimation (5) in random subspace optimization [43, 46], we observe significant differences. First, our gradient estimation (13) accounts for the layer-wise structure of the network, requiring the projection matrix $\boldsymbol{P}$ to be block-diagonal, whereas in random subspace optimization, $\boldsymbol{P}$ is not. Additionally, our method introduces a layer-wise low-rank perturbation matrix, reflected by the block-diagonal structure of $\boldsymbol{P}$, with lazy updates to the column and row spaces defined by $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$. In contrast, random subspace optimization simply requires $\boldsymbol{P}$ to be random. These distinctions highlight the key differences between our gradient estimation and existing methods in random subspace optimization.

Theorem 1 (b) guarantees that the distance $\Phi(\boldsymbol{x})$ between the expected gradient estimate and the BP gradient in the subspace spanned by $\boldsymbol{P}$ is small. Moreover, by setting $\varepsilon = \frac{1}{q+4}$, the distance $\Phi(\boldsymbol{x})$ is bounded by a constant $L_2/6$, independent of the parameter dimension d. This implies that the error in our gradient estimation does not scale with the extremely high parameter dimensions of LLMs, providing accurate gradient estimation—crucial for optimizing LLMs.

Next, we utilize a strictly convex quadratic loss to further analyze our gradient estimation in Eqn. (13). This choice is motivated by the fact that, after pretraining, the LLM parameters tend to converge toward a local minimum within a local basin, which can be well-approximated by a quadratic loss [42].

Theorem 2. *Let $f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{H} \boldsymbol{x}$ and $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$, where $\boldsymbol{H} \in \mathbb{R}^{d \times d}$ is positive definite. We have*

$$\mathbb{E}_{\boldsymbol{z}}[\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})] = \boldsymbol{P}\boldsymbol{P}^\top \nabla f(\boldsymbol{x}), \quad (14)$$

$$\mathbb{E}_{\boldsymbol{z}}[\|\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2] = (q+2)\|\boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|^2, \quad (15)$$

$$\mathbb{E}_{\boldsymbol{z}}\left[\frac{\langle \nabla f(\boldsymbol{x}), \hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\rangle^2}{\|\boldsymbol{P}^\top \nabla f(\boldsymbol{x})\|^2 \|\hat{g}_\varepsilon(\boldsymbol{x}, \boldsymbol{P}, \boldsymbol{z})\|^2}\right] = \frac{1}{q}. \quad (16)$$

See its proof in Appendix 8.6. Theorem 2 demonstrates several advantageous properties of our gradient estimation on the quadratic function. First, Eqn. (14) establishes the equivalence between the expected gradient estimation and the BP gradient within the subspace spanned by our projection matrix $\boldsymbol{P}$. Second, Eqn. (15) shows that, in this subspace, the variance of the gradient estimation scales linearly with the subspace dimension q. In contrast, the variance of gradient estimation (2) in MeZO depends linearly on the model's parameter dimension d, which is significantly larger than q. Finally, Eqn. (16) reveals that the expected cosine similarity between our estimated gradient and the BP gradient within the subspace depends only on the subspace dimension $q \ll d$, indicating that our gradient estimation provides a highly accurate parameter update direction.

Building upon the above results, we can prove the convergence of our SubZero.

Theorem 3. *Let $f \in C_{L_1}^{1,1}(\mathbb{R}^d)$ be a non-convex function bounded below by f^*. Suppose $\mathcal{E}_k = (\boldsymbol{z}_0, \boldsymbol{z}_1, \cdots, \boldsymbol{z}_k)$ with $\boldsymbol{z}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_q)$, and let $\mathcal{P}_j = (\boldsymbol{P}_0, \boldsymbol{P}_1, \cdots, \boldsymbol{P}_j)$, where $\boldsymbol{P}_j$ is defined in (10) with a fixed update frequency F. Then, the sequence $\{\boldsymbol{x}_k\}_{k>0}$ generated by Algorithm 3 satisfies:*

$$\frac{1}{T}\sum_{k=0}^{T-1} \mathbb{E}_{\mathcal{E}_k, \mathcal{P}_{\lfloor k/F \rfloor}}\left[\|\nabla f(\boldsymbol{x}_k)\|^2\right] \leq \epsilon$$

with $T = \mathcal{O}\left(\frac{d}{\epsilon}\right)$ if the perturbation scale satisfies $\varepsilon \leq \mathcal{O}\left(\frac{\epsilon^{1/2}}{q^{3/2} d^{1/2} L_1^{3/2}}\right)$. Here $T = KF$, where K denotes the total number of subspace updates.

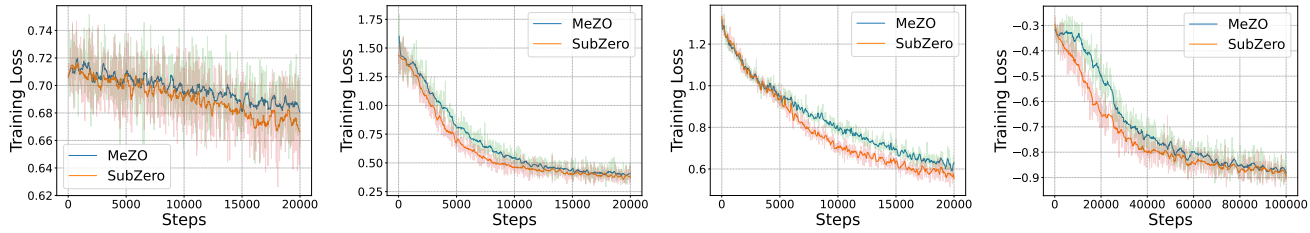
See its proof in Appendix 8.6. Theorem 3 guarantees the convergence of our SubZero when the projection matrix $\boldsymbol{P}$ is updated at a fixed frequency F.

6. Experiments

In this section, we present comprehensive experiments to evaluate the effectiveness of SubZero. We conduct our experiments using medium-sized masked LLMs (RoBERTa-large [38]) and large-scale autoregressive LLMs (OPT-1.3B and 13B [64], LLaMA2-7B [55], and Mistral-7B [27]). Our exploration covers full-parameter tuning (FT) [2] and three PEFT schemes: LoRA [24], prefix tuning [36], and prompt tuning [33]. For comparison, we include leading ZO methods, such as MeZO [40], ZO-AdaMU [28], S-MeZO [39], HiZOO [68], and LOZO [9] alongside inference-only memory-efficient baselines like zero-shot, in-context learning (ICL) [7], and linear probing (LP) [32]. As the first and most popular ZO optimizer for LLM fine-tuning, MeZO is considered our primary competitor. We also use the FO optimizer SGD as a benchmark. Since appropriate prompts are critical for ZO optimization [40, 65], all experiments

Table 3. Performance of fine-tuning OPT-13B on SuperGLUE with various experimental settings (with 1000 examples). AVG: average relative percentage difference with MeZO of all tasks.

Task type	classification							– multiple choice –		— generation —		
Task	**SST-2**	**RTE**	**CB**	**BoolQ**	**WSC**	**WIC**	**MultiRC**	**COPA**	**ReCoRD**	**SQuAD**	**DROP**	**AVG.**
SGD(FT)	94.9	82.3	85.7	78.4	65.3	65.8	74.2	90.0	82.4	88.0	35.5	-
Zero-shot	58.8	59.6	46.4	59.0	38.5	55.0	46.9	80.0	81.2	46.2	14.6	-
ICL	87.0	62.1	57.1	66.9	39.4	50.5	53.1	87.0	82.5	75.9	29.6	-
LP	93.4	68.6	67.9	59.3	63.5	60.2	63.5	55.0	27.1	3.7	11.1	-
MeZO(FT)	92.1	71.5	71.4	74.4	61.5	60.0	60.1	87.0	82.0	84.2	31.2	0%
ZO-AdaMU(FT)	92.1	72.9	67.9	73.0	61.5	60.7	63.0	**89.0**	**83.0**	82.4	**32.0**	0.46%
S-MeZO(FT)	92.3	**76.9**	**75.0**	**76.5**	61.1	58.2	**63.3**	87.0	71.2	77.9	31.9	-0.10%
HiZOO(FT)	91.3	69.3	69.4	67.3	63.5	59.4	55.5	88.0	81.4	81.9	31.3	-2.15%
LOZO(FT)	**92.9**	73.6	71.4	70.7	63.5	60.2	60.3	87.0	81.7	**84.5**	30.7	0.10%
SubZero(FT)	92.1	74.0	73.2	75.3	**65.4**	**60.8**	61.0	88.0	82.3	**84.5**	**32.0**	**1.89%**
MeZO(LoRA)	92.2	74.4	69.6	75.2	64.4	59.7	58.2	87.0	82.0	82.9	31.0	0%
ZO-AdaMU(LoRA)	88.0	72.0	71.6	72.6	60.1	56.4	58.9	88.0	**83.2**	76.8	**32.4**	-1.78%
S-MeZO(LoRA)	90.8	62.2	**75.0**	72.9	51.9	55.8	56.4	86.0	69.9	76.4	31.7	-5.79%
HiZOO(LoRA)	90.6	67.5	69.6	70.5	63.5	60.2	60.2	87.0	81.9	**83.8**	31.2	-1.16%
SubZero(LoRA)	**93.8**	**75.5**	71.4	**76.1**	**65.4**	**60.3**	**60.3**	**89.0**	81.9	83.7	31.3	**1.57%**



(a) WIC: OPT-13B, FT (b) SQuAD: OPT-13B, LoRA (c) CB: LLaMA2-7B, Prompt (d) MNLI: RoBERTa, FT, Non-diff

Figure 2. Training loss curves of MeZO and SubZero.

incorporate prompt templates. Since a larger batch size reduces the variance of the estimated gradients, all compared methods use a batch size of 16 unless otherwise specified. All experimental settings are detailed in Appendix 8.2-8.5.

6.1. Results under Different Experimental Settings

Following the settings in MeZO [40], we evaluated SubZero using OPT-13B on the SuperGLUE benchmark [57], which covers a diverse range of tasks, including classification, multiple-choice, and generation, as outlined in Table 3. The ZO methods were applied to both full-parameter tuning (FT) and LoRA fine-tuning schemes. The comparisons with vanilla LoRA and SGD with gradient accumulation are provided in Appendix 8.1.

Table 3 presents the key findings, highlighting the best-performing ZO method in bold. The results show that ZO techniques significantly outperform baseline approaches like zero-shot, in-context learning, and linear probing, underscoring their ability to enhance a pre-trained model's performance on downstream tasks. From Table 3, one can also observer that MeZO, the first ZO optimizer for LLM fine-tuning, is highly competitive after carefully tuning its hyperparameters. Only ZO-AdaMU and LOZO, apart from SubZero, outperform MeZO in FT scheme. SubZero consistently surpasses MeZO across all tasks and fine-tuning schemes. In FT scheme, S-MeZO showed competitive performance on several classification tasks. However, its performance on ReCoRD remained unsatisfactory even after hyperparameter tuning. Excluding ReCoRD, SubZero still outperforms S-MeZO with 2.05% vs. 1.20% in FT scheme and 1.74% vs. -4.90% in LoRA scheme.

We further extended our evaluation of SubZero using OPT-1.3B, LLaMA2-7B, and Mistral-7B in FT and three PEFT schemes: LoRA, prefix tuning, and prompt tuning. As shown in Table 4, SubZero outperformed MeZO across all models and fine-tuning schemes. Notably, while MeZO struggled in the prompt tuning scheme, SubZero excelled, achieving performance levels that closely matched those of the SGD optimizer.

We also present several training loss curves in Fig. 2 (a)-(c), demonstrating that SubZero generally converges faster and achieves lower losses compared to MeZO.

6.2. Results on Non-Differentiable Objectives

Following MeZO [40], we apply SubZero to fine-tune RoBERTa-large and OPT-13B using full-parameter fine-tuning with two non-differentiable objectives—optimizing accuracy in classification tasks and F1 in the SQuAD task.

Table 4. Performance of fine-tuning LLaMA2-7B and Mistral-7B on CB, and OPT-1.3B on SST-2.

	LLaMA2-7B				Mistral-7B				OPT-1.3B			
	FT	**LoRA**	**Prefix**	**Prompt**	**FT**	**LoRA**	**Prefix**	**Prompt**	**FT**	**LoRA**	**Prefix**	**Prompt**
SGD	69.6	75.0	69.6	69.6	73.2	75.0	69.6	62.5	93.2	93.0	93.1	90.7
MeZO	64.3	73.2	69.6	60.7	62.5	69.6	58.3	57.1	92.3	92.8	91.6	85.9
SubZero	**71.4**	**75.0**	**76.8**	**66.1**	**64.3**	**73.2**	**64.3**	**62.5**	**93.4**	**92.9**	**92.2**	**89.1**

Table 5. Orthogonal projection matrix.

Dataset	Ortho.	Accuracy
RTE	✗	67.5
	✓	**74.0**
WSC	✗	59.6
	✓	**65.1**

Table 6. Fine-tuning performance comparison between SubZero and MeZO on RoBERTa-large and OPT-13B with non-differentiable objectives.

Model	RoBERTa-large				OPT-13B
Task	**SST-2**	**SST-5**	**SNLI**	**MNLI**	**SQuAD**
Zero-shot	79.0	35.5	50.2	48.8	46.2
Cross entropy (Adam)	93.9	55.9	88.7	83.8	84.2
Cross entropy (MeZO)	92.9	53.2	83.0	77.0	84.2
Cross entropy (SubZero)	93.4	54.0	84.7	77.4	84.5
Accuracy/F1 (MeZO)	92.4	46.5	81.9	73.9	80.2
Accuracy/F1 (SubZero)	**92.7**	**47.1**	**83.0**	**74.8**	**81.1**

As baselines, we also report results using the differentiable cross-entropy objective with Adam, MeZO, and SubZero. As shown in Table 6, SubZero consistently outperforms MeZO with the two non-differentiable objectives. The training loss curves in Fig. 2 (d) also demonstrate the advantage of SubZero over MeZO.

6.3. Memory Usage and Wall-Clock Time Analysis

We compare the memory consumption and wall-clock time of ZO methods (MeZO and SubZero), SGD, and inference-only approaches (zero-shot and in-context learning (ICL)) using OPT-13B (see Table 10 in Appendix 8.1). Since inference-only methods do not involve fine-tuning, they have zero wall-clock time and their memory usage reflects only the inference load. For fine-tuning, all methods were run for 20K steps. The ZO methods, including SubZero, achieved over a 1.8× reduction in memory usage compared to SGD. Notably, SubZero's memory footprint closely aligns with MeZO's, while offering improved performance. We use per-layer weight updates for MeZO and SubZero (see Appendix 8.2), resulting in nearly identical memory usage for FT and LoRA schemes when one decimal place is reserved.

Although SubZero introduces additional computational overhead for generating projection matrices via QR decomposition, the observed extra time overhead remains below 9% across all OPT model sizes, with a mean value of 4.79% of the total wall-clock time (see Table 11 in Appendix 8.1).

Note that due to differences in how steps are defined between ZO methods and SGD, direct wall-clock time comparisons between the two are not entirely meaningful.

6.4. Ablation Study

We conducted a thorough investigation of the effectiveness of our techniques. Table 5 shows that using a column-orthogonal projection matrix significantly outperforms a Gaussian random projection matrix, primarily due to the low-rank structure of the perturbation matrices (see experimental settings in Appendix 8.4). This low-rank perturbation is key to improving the quality of gradient estimation.

Next, Table 7 explores the effects of subspace rank r and update frequency F in Algorithm 3. The results demonstrate that SubZero is robust to variations in the subspace rank. However, performance drops sharply when the update frequency is too low, as the optimization becomes constrained to a single subspace for too long, limiting its adaptability.

Table 7. Subspace change frequency F and rank r.

$F \setminus r$	32	64	128
500	**72.6**	70.0	72.2
1000	73.6	71.8	**74.0**
2000	72.2	**73.3**	72.2
20000	70.4	**71.1**	68.6

Table 8. Reshaping strategy for non-square matrices on SST-2 with OPT-1.3B in PEFT schemes.

Method	LoRA	Prefix	Prompt
MeZO	92.8	91.6	85.9
SubZero(w/o)	92.1	89.4	74.2
SubZero(w/)	**92.9**	**92.2**	**89.1**

Finally, Table 8 underscores the critical role of the reshaping strategy for handling highly non-square perturbation matrices, essential for ensuring effective perturbations in different layers of the model. Together, these results highlight the improvements brought by our design choices, particularly in terms of projection and reshaping strategies, and their impact on SubZero's robustness and performance.

Due to limited space, the ablation studies of SubZero on random seed, batch size, and combination with Adam are given in Appendix 8.1.

7. Conclusion

We have demonstrated that SubZero effectively fine-tunes large LLMs across various tasks and schemes with a memory cost comparable to that of inference. Extra experiments indicate that SubZero can optimize non-differentiable objectives. Our theory explains how SubZero reduces the variance of gradient estimates and hence accelerates convergence.

Limitation. In addition to the representative first-order and primitive zero-order optimizers, we have yet to investigate the combinations of SubZero with other first-order and zero-order optimizers to evaluate the implications on convergence speed. While SubZero is compatible with memory-efficient techniques like parameter quantization [35], we have not thoroughly explored the practical effects of these combinations. A theoretical analysis of the reshaping strategy is valuable but left for future work.

Acknowledgments

This work is supported by the NSF of China (grant nos. 62131003 and 62102034), Beijing Municipal Science and Technology Project (Z241100001324011), the Ministry of Education, Singapore, under its AcRF Tier 2 Funding (Proposal ID: T2EP20224-0048). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv:2303.08774*, 2023. 1

[2] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 7319–7328, 2021. 3, 5, 6

[3] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993. 1, 2

[4] 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*, 2006. 16

[5] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth PASCAL recognizing textual entailment challenge. In *Proceedings of the Second Text Analysis Conference*, 2009. 16

[6] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 632–642, 2015. 16

[7] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, pages 1877–1901, 2020. 6

[8] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv:1604.06174*, 2016. 2

[9] Yiming Chen, Yuan Zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. Enhancing zeroth-order fine-tuning for language models with low-rank structures. In *Proceedings of the International Conference on Learning Representations*, 2025. 6

[10] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019. 16

[11] Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In *Proceedings of the International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, 2005. 16

[12] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022. 2

[13] Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *Proceedings of Sinn und Bedeutung 23*, 2019. 16

[14] 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. 2

[15] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *Proceedings of the International Conference on Learning Representations*, 2022. 2

[16] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems*, 36, 2024. 2

[17] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023. 3

[18] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2368–2378, 2019. 16

[19] John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806, 2015. 2

[20] Tanmay Gautam, Youngsuk Park, Hao Zhou, Parameswaran Raman, and Wooseok Ha. Variance-reduced zeroth-order methods for fine-tuning language models. In *Proceedings of the International Conference on Machine Learning*, 2024. 1, 2, 16

[21] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, 2007. 16

[22] Wentao Guo, Jikai Long, Yimeng Zeng, Zirui Liu, Xinyu Yang, Yide Ran, Jacob R Gardner, Osbert Bastani, Christopher De Sa, Xiaodong Yu, Beidi Chen, and Zhaozhuo Xu. Zeroth-order fine-tuning of LLMs with extreme sparsity. *arXiv:2406.02913*, 2024. 4, 15

[23] Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. In *Proceedings of the International Conference on Machine Learning*, 2024. 3

[24] 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. In *Proceedings of the International Conference on Learning Representations*, 2022. 2, 3, 5, 6

[25] Kevin G Jamieson, Robert Nowak, and Ben Recht. Query complexity of derivative-free optimization. *Advances in Neural Information Processing Systems*, 25, 2012. 2

[26] Kaiyi Ji, Zhe Wang, Yi Zhou, and Yingbin Liang. Improved zeroth-order variance reduced algorithms and analysis for nonconvex optimization. In *Proceedings of the International Conference on Machine Learning*, pages 3100–3109. PMLR, 2019. 2

[27] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv:2310.06825*, 2023. 6

[28] Shuoran Jiang, Qingcai Chen, Youcheng Pan, Yang Xiang, Yukang Lin, Xiangping Wu, Chuanyi Liu, and Xiaobao Song. ZO-AdaMU optimizer: Adapting perturbation by the momentum and uncertainty in zeroth-order optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 18363–18371, 2024. 1, 2, 4, 6, 13, 14

[29] 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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 252–262, 2018. 16

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2015. 1, 2, 4

[31] David Kozak, Stephen Becker, Alireza Doostan, and Luis Tenorio. A stochastic subspace approach to gradient-free optimization in high dimensions. *Computational Optimization and Applications*, 79(2):339–368, 2021. 1, 2

[32] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *Proceedings of the International Conference on Learning Representations*, 2022. 6

[33] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021. 5, 6

[34] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning*, 2012. 16

[35] Bingrui Li, Jianfei Chen, and Jun Zhu. Memory efficient optimizers with 4-bit states. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 8

[36] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*, pages 4582–4597, 2021. 5, 6

[37] 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. 2

[38] 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. *arXiv:1907.11692*, 2019. 6

[39] 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:2402.15751*, 2024. 1, 2, 6, 13, 16, 17

[40] 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. 1, 2, 3, 4, 6, 7, 15, 16, 17, 18

[41] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527–566, 2017. 2, 3, 19, 20, 22, 23

[42] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in Neural Information Processing Systems*, 33:512–523, 2020. 6

[43] Ryota Nozawa, Pierre-Louis Poirion, and Akiko Takeda. Zeroth-order random subspace algorithm for non-smooth convex optimization. *Journal of Optimization Theory and Applications*, 204(3):53, 2025. 1, 2, 3, 4, 6

[44] Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1267–1273, 2019. 16

[45] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016. 16

[46] Lindon Roberts and Clément W Royer. Direct search based on probabilistic descent in reduced spaces. *SIAM Journal on Optimization*, 33(4):3057–3082, 2023. 1, 2, 3, 4, 6

[47] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Proceedings of the AAAI Spring Symposium Series*, 2011. 16

[48] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021. 15

[49] Junhong Shen, Neil Tenenholtz, James Brian Hall, David Alvarez-Melis, and Nicolo Fusi. Tag-LLM: Repurposing general-purpose LLMs for specialized domains. In *Proceed-*

ings of the International Conference on Machine Learning, pages 44759–44773, 2024. 1

[50] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013. 16

[51] 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 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013. 5

[52] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, et al. Release strategies and the social impacts of language models. *arXiv:1908.09203*, 2019. 1

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

[54] Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meProp: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *Proceedings of the International Conference on Machine Learning*, pages 3299–3308, 2017. 2

[55] 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. 1, 6

[56] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 1

[57] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv: 1905.00537*, 2019. 7, 16

[58] Junjie Wang, Guangjing Yang, Wentao Chen, Huahui Yi, Xiaohu Wu, Zhouchen Lin, and Qicheng Lao. MLAE: Masked LoRA experts for visual parameter-efficient fine-tuning. *arXiv:2405.18897*, 2024. 2

[59] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018. 16

[60] Yifan Yang, Kai Zhen, Ershad Banijamali, Athanasios Mouchtaris, and Zheng Zhang. AdaZeta: Adaptive zeroth-order tensor-train adaption for memory-efficient large language models fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 977–995, 2024. 1, 2, 16, 17

[61] Hongwei Yong, Ying Sun, and Lei Zhang. A general regret bound of preconditioned gradient method for dnn training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7866–7875, 2023. 17

[62] Pengyun Yue, Long Yang, Cong Fang, and Zhouchen Lin. Zeroth-order optimization with weak dimension dependency. In *Proceedings of the Annual Conference on Learning Theory*, pages 4429–4472. PMLR, 2023. 1

[63] Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint 1810.12885*, 2018. 16

[64] 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:2205.01068*, 2022. 1, 5, 6

[65] Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D. Lee, Wotao Yin, Mingyi Hong, Zhangyang Wang, Sijia Liu, and Tianlong Chen. Revisiting zeroth-order optimization for memory-efficient LLM fine-tuning: A benchmark. In *Proceedings of the International Conference on Machine Learning*, pages 59173–59190, 2024. 1, 2, 4, 5, 6, 15, 17

[66] Zhong Zhang, Bang Liu, and Junming Shao. Fine-tuning happens in tiny subspaces: Exploring intrinsic task-specific subspaces of pre-trained language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1701–1713, 2023. 3

[67] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the International Conference on Machine Learning*, 2024. 1, 2, 3

[68] Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor W Tsang. Second-order fine-tuning without pain for LMMs: A Hessian informed zeroth-order optimizer. In *Proceedings of the International Conference on Learning Representations*, 2025. 4, 6, 13