Low-Rank Curvature for Zeroth-Order Optimization in LLM Fine-Tuning

Hyunseok Seung[1], **Jaewoo Lee**[2], **Hyunsuk Ko**[3*]
[1]University of Wisconsin – Madison
[2]University of Georgia
[3]Hanyang University
hseung2@wisc.edu, jaewoo.lee@uga.edu, hyunsuk@hanyang.ac.kr

Abstract

We introduce LOREN, a curvature-aware zeroth-order (ZO) optimization method for fine-tuning large language models (LLMs). Existing ZO methods, which estimate gradients via finite differences using random perturbations, often suffer from high variance and suboptimal search directions. Our approach addresses these challenges by: (i) reformulating the problem of gradient preconditioning as that of adaptively estimating an anisotropic perturbation distribution for gradient estimation, (ii) capturing curvature through a low-rank block diagonal preconditioner using the framework of natural evolution strategies, and (iii) applying a REINFORCE leave-one-out (RLOO) gradient estimator to reduce variance. Experiments on standard LLM benchmarks show that our method outperforms state-of-the-art ZO methods by achieving higher accuracy and faster convergence, while cutting peak memory usage by up to 27.3% compared with MeZO-Adam.

Code — https://github.com/hseung88/loren

1 Introduction

Fine-tuning large language models (LLMs) with first-order (FO) methods such as SGD (Robbins and Monro 1951) and AdamW (Kingma and Ba 2015; Loshchilov and Hutter 2019) incurs significant memory overhead primarily due to gradient computations during backpropagation. To address this limitation, there has been renewed interest in developing zeroth-order (ZO) optimization methods for LLM fine-tuning. Recent ZO optimizers, such as MeZO (Malladi et al. 2023), estimate gradient using only forward-pass evaluations of the model, eliminating the need to store intermediate activations or perform backpropagation for gradient computation, thereby significantly reducing memory requirements. The low memory footprint makes ZO optimizers particularly appealing for LLM fine-tuning and recent studies (Malladi et al. 2023; Chen et al. 2025) have shown promising results.

Despite the memory efficiency, existing ZO optimizers exhibit slow convergence rates due to two fundamental limitations. First, the finite-difference gradient estimators employed in ZO methods suffer from high variance, particularly in high-dimensional stochastic settings. This high variance leads to noisy gradient approximations, resulting in unstable parameter updates and degraded optimization performance (Nesterov and Spokoiny 2017; Ghadimi and Lan 2013). In the absence of variance reduction techniques, the sample complexity measured in terms of function evaluations scales poorly with model dimensionality (Duchi et al. 2015; Nesterov and Spokoiny 2017; Gautam et al. 2024). Second, existing ZO optimizers are agnostic to the anisotropic curvature of loss landscapes in LLMs, i.e., they fail to adapt to curvature heterogeneity across different weights and layers. This lack of curvature awareness can lead to optimization inefficiencies (e.g., oscillations in high-curvature directions or stagnation along nearly flat directions) and may even result in convergence to saddle points (Zhao et al. 2025).

In this paper, we propose LOREN (**L**ow-rank c**O**variance, **RE**INFORCE, and **N**atural evolution strategies), a novel ZO optimization method designed to overcome these challenges. LOREN introduces three main innovations:

(i) We reformulate the problem of gradient preconditioning in ZO optimization as that of adaptively estimating a sampling distribution from which random perturbations are drawn for finite-difference gradient estimation. Existing ZO optimizers typically draw random perturbations from either isotropic Gaussian or uniform distribution over unit sphere, which assume uniform curvature in all directions and thus ignore the underlying geometry of loss landscape. In contrast, LOREN dynamically learns a perturbation distribution that captures anisotropic local curvature of loss function.

(ii) LOREN leverages the framework of natural evolution strategy (NES) (Rechenberg 1973; Wierstra et al. 2008) to accelerate the search for optimal parameters of the perturbation distribution and models the inverse of Hessian using the Kronecker factored rank-1 approximation to scale NES to LLM fine-tuning. The Kronecker factorization approach allows LOREN to approximate curvature information with significantly reduced memory overhead, making it suitable for LLM training.

(iii) Unlike traditional ZO optimizers that rely on finite-difference gradient estimators, LOREN employs the REINFORCE leave-one-out (RLOO) (Kool, van Hoof, and Welling 2019) estimator to reduce the variance and make effective use of multiple function evaluations.

*Corresponding author.

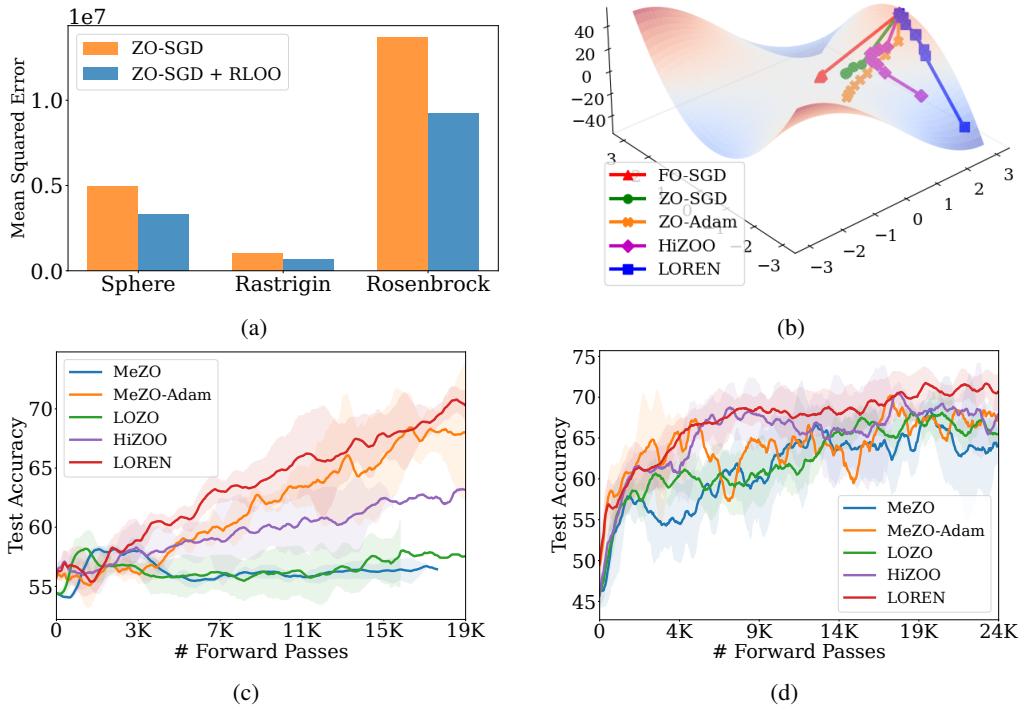

Figure 1: **(a)** Mean squared errors of ZO gradient estimates, with and without RLOO, relative to the true gradient on the 1,000-dimensional Sphere, Rastrigin, and Rosenbrock functions. **(b)** Optimization trajectories of FO-SGD and ZO optimizers on the monkey saddle function, all initialized at (2.9, -0.01). Accuracy curves for **(c)** GPT-2-XL fine-tuned on QNLI and **(d)** OPT-13B fine-tuned on CB, using early stopping.

By combining these approaches, LOREN produces search directions that are both well-conditioned and low-variance, all while preserving the memory-efficient nature of ZO methods. On Figure 1a, we present the mean squared error (MSE) of gradient estimates for ZO-SGD, with and without RLOO, for three 1,000-dimensional test functions: Sphere, Rastrigin, and Rosenbrock. For each method, we generate 5,000 gradient estimates at a fixed point and compute their MSE relative to the true gradient. Both methods use four perturbations per gradient estimate for fairness. As shown, ZO-SGD with RLOO consistently achieves a lower MSE, demonstrating effective variance reduction. On Figure 1b, we visualize the optimization trajectories of FO and ZO optimizers on the monkey saddle function. While ZO-SGD and ZO-Adam struggle due to noisy gradient estimates, HiZOO (Zhao et al. 2025) shows moderate improvement by using a ZO-Hessian estimate. Notably, LOREN follows the most efficient path, escaping the saddle region by leveraging low-rank curvature and low-variance gradient estimates.

To validate the effectiveness of LOREN, we evaluate its performance in fine-tuning both masked language models and autoregressive models on the GLUE (Wang et al. 2018) and SuperGLUE (Wang et al. 2019) tasks. Figure 1c and 1d present the test accuracy curves of state-of-the-art ZO optimizers, fine-tuning GPT-2-XL on QNLI and OPT-13B on CB, respectively. By incorporating curvature-aware updates and variance reduction, LOREN achieves the highest mean accuracy, demonstrating both superior performance and markedly more stable convergence. The key contributions of our work can be summarized as follows:

- We introduce LOREN, the first ZO optimizer that simultaneously adapts to curvature and applies variance reduction, enabling an efficient fine-tuning of LLMs. This combined approach delivers stable and scalable updates, even in high-dimensional and ill-conditioned settings.
- We establish the link between ZO gradients and evolution strategies and directly estimate the preconditioned ZO gradients using the score function estimator without any additional forward passes. We construct a damped rank-1 covariance structure to preserve memory efficiency and ensure that the additional memory overhead to store curvature information remains negligible.
- To the best of our knowledge, LOREN is the first method to apply a block-diagonal approximation of the Hessian matrix in ZO optimization, capturing richer curvature information than a pure diagonal approximation.
- We provide extensive experimental results on standard LLM benchmarks, comparing all leading ZO methods for fine-tuning. LOREN consistently delivers higher test accuracy while maintaining lower memory footprint compared to other state-of-the-art preconditioned or adaptive ZO methods, raising the bar for memory-efficient LLM fine-tuning.

2 Related Work

Our work intersects ZO optimization, memory-efficient LLM fine-tuning, and curvature-aware variance reduction.

ZO Optimization for LLMs ZO methods replace explicit gradients with function evaluations via finite-difference approximations such as SPSA (Spall 1992). This paradigm gained traction for LLM fine-tuning due to its potential for extreme memory efficiency compared to backpropagation. MeZO (Malladi et al. 2023) pioneered this application, adapting ZO-SGD (Spall 1992) with an in-place implementation to match inference memory costs. While demonstrating feasibility and achieving strong results, MeZO can be sensitive to prompts and exhibits higher variance than FO methods. LOZO (Chen et al. 2025) focused on aligning the ZO gradient estimator with the observed low-rank structure of LLM gradients, proposing a low-rank gradient estimator. LOREN also uses a low-rank structure but applies it to the preconditioner (i.e., covariance matrix), rather than directly estimating a low-rank gradient as in LOZO.

Preconditioned ZO Methods To address slow convergence on ill-conditioned loss landscapes, curvature information has been incorporated. HiZOO (Zhao et al. 2025) incorporates second-order information by explicitly estimating the diagonal entries of the Hessian with an additional forward pass and uses it for preconditioning. Other works have explored Hessian-aware ZO methods in different contexts rather than fine-tuning LLMs (Ye et al. 2018; Balasubramanian and Ghadimi 2018). LOREN draws inspiration from the NES (Rechenberg 1973; Wierstra et al. 2008), adapting a low-rank Kronecker-factored approximation of the perturbation covariance matrix. This allows capturing curvature information to guide the search direction efficiently without storing or estimating second-order elements directly.

Variance Reduction in ZO Methods The high variance of ZO gradient estimators is a key obstacle. MeZO-SVRG (Gautam et al. 2024) adapts SVRG (Johnson and Zhang 2013), using periodic full-batch estimates to correct minibatch gradients, to improve stability and convergence over MeZO but at the cost of increased memory, requiring storage for reference gradients and parameters. LOREN utilizes the RLOO (Kool, van Hoof, and Welling 2019) method, a score function gradient estimator combined with a leave-one-out baseline, for gradient estimation. RLOO computes a baseline for each sample within a batch using the rewards (function values) of the other samples in the same batch. This provides effective variance reduction without needing full-batch computations like SVRG, thus preserving the minimal memory footprint of LOREN.

3 Preliminaries

3.1 Notations

Vectors are denoted by lowercase bold (e.g., $\mathbf{x}$), and matrices by uppercase bold (e.g., $\mathbf{X}$). We write x_i to denote the i^{th} entry of vector $\mathbf{x}$. $\|\mathbf{x}\|$ represents the Euclidean norm unless otherwise stated. $\otimes$ represents the Kronecker product. For a matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$, its vectorization is $\text{vec}(\mathbf{X}) = \begin{bmatrix}\mathbf{X}_{*,1}^\intercal & \mathbf{X}_{*,2}^\intercal & \cdots & \mathbf{X}_{*,n}^\intercal\end{bmatrix}^\intercal$, where $\mathbf{X}_{*,j}$ denotes the j^{th} column of matrix $\mathbf{X}$. For two matrices $\mathbf{A}$ and $\mathbf{B}$, the symbol : denotes their trace product, i.e., $\mathbf{A} : \mathbf{B} = \text{tr}(\mathbf{A}^\intercal\mathbf{B})$.

3.2 Zeroth-Order Gradient Estimates

We consider the following stochastic optimization problem using the ZO oracle:

$$\arg\min_{\mathbf{x}\in\mathbb{R}^{\mathbf{d}}} f(\mathbf{x}) := \mathbb{E}_{\xi\sim\mathbb{P}}[\ell(\mathbf{x};\xi)]\,,$$

where $\mathbf{x}$ denotes the model parameters, ξ denotes a random data sample, and f is the expected loss over the data distribution. When FO gradients are inaccessible, a common strategy is to estimate gradients using the finite-difference method (Ghadimi and Lan 2013; Nesterov and Spokoiny 2017). A widely used technique is the Simultaneous Perturbation Stochastic Approximation (SPSA), which estimates gradients using random perturbations in all coordinates simultaneously.

Definition 3.1 (SPSA (Spall 1992))**.** Let $f : \mathbb{R}^d \to \mathbb{R}$. For $\epsilon > 0$, the SPSA gradient estimator is given by

$$\hat{\nabla} f(\mathbf{x}) = \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)}\left[\frac{f(\mathbf{x}+\epsilon\mathbf{u}) - f(\mathbf{x}-\epsilon\mathbf{u})}{2\epsilon}\mathbf{u}\right].$$

$\hat{\nabla} f(\mathbf{x})$ is closely related to the gradient of Gaussian smoothed objective.

Definition 3.2 (Generalized Gaussian smoothing)**.** Let $f : \mathbb{R}^d \to \mathbb{R}$. The Gaussian smoothing of $f(\mathbf{x})$ is defined by

$$f_{\epsilon,\mathbf{\Sigma}}(\mathbf{x}) \triangleq \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{\Sigma})}[f(\mathbf{x}+\epsilon\mathbf{u})]\,,$$

where $\epsilon > 0$ controls the smoothness.

Proposition 3.3. *For $f : \mathbb{R}^d \to \mathbb{R}$, the gradient of Gaussian smoothed f is given by*

$$\nabla f_{\epsilon,\mathbf{\Sigma}}(\mathbf{x}) = \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{\Sigma})}\left[\frac{f(\mathbf{x}+\epsilon\mathbf{u}) - f(\mathbf{x}-\epsilon\mathbf{u})}{2\epsilon}\mathbf{\Sigma}^{-1}\mathbf{u}\right]. \tag{1}$$

Proposition 3.3 shows that $\nabla f_{\epsilon,\mathbf{\Sigma}}(\mathbf{x}) = \hat{\nabla} f(\mathbf{x})$ when $\mathbf{\Sigma} = \mathbf{I}_d$. As $\epsilon \to 0$, $\nabla f_{\epsilon,\mathbf{\Sigma}}(\mathbf{x})$ approximates the true gradient:

$$\begin{aligned}\lim_{\epsilon\to 0}\nabla f_{\epsilon,\mathbf{\Sigma}}(\mathbf{x}) &= \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{\Sigma})}[\langle\nabla f(\mathbf{x}),\,\mathbf{u}\rangle\mathbf{\Sigma}^{-1}\mathbf{u}]\\ &= \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{\Sigma})}[\mathbf{\Sigma}^{-1}\mathbf{u}\mathbf{u}^\intercal\nabla f(\mathbf{x})] = \nabla f(\mathbf{x})\,.\end{aligned} \tag{2}$$

3.3 REINFORCE with Leave-One-Out Baseline

To reduce variance, score function estimators are typically used with a control variate b, referred to as a baseline that is independent of $\mathbf{z}_k$:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \frac{1}{K}\sum_{k=1}^{K}(f(\mathbf{z}_k) - b)\nabla_{\boldsymbol{\theta}}\log p(\mathbf{z}_k;\boldsymbol{\theta})\,,$$

where $\mathbf{z}_k \sim p(\mathbf{z};\boldsymbol{\theta})$. For $K \geq 2$, the RLOO estimator sets b to the leave-one-out average of function values, $\sum_{j\neq k} f(\mathbf{z}_j)/(K-1)$, leveraging multiple evaluations of f to reduce variance. The estimator can be equivalently expressed as

$$\frac{1}{K-1}\sum_{k=1}^{K}\left(f(\mathbf{z}_k) - \frac{1}{K}\sum_{j=1}^{k} f(\mathbf{z}_j)\right)\nabla_{\boldsymbol{\theta}}\log p(\mathbf{z}_k;\boldsymbol{\theta})\,.$$

4 Methods

This section presents a detailed derivation of LOREN.

4.1 Preconditioning via Evolution Strategies

Consider the following preconditioned gradient update:

$$\mathbf{x} \leftarrow \mathbf{x} - \eta \tilde{\mathbf{H}}^{-1} \nabla f(\mathbf{x}),$$

where $\tilde{\mathbf{H}}$ is a symmetric positive definite matrix approximating the curvature information. By replacing the perturbation vector $\mathbf{u}$ in (2) with the scaled Gaussian $\tilde{\mathbf{H}}^{-1/2}\mathbf{u}$, we get

$$\begin{aligned}
\tilde{\mathbf{H}}^{-1} \nabla f(\mathbf{x}) &= \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)}[\tilde{\mathbf{H}}^{-1/2}\mathbf{u}\mathbf{u}^\intercal \tilde{\mathbf{H}}^{-1/2} \nabla f(\mathbf{x})] \\
&= \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)}[\langle \nabla f(\mathbf{x}),\, \tilde{\mathbf{H}}^{-1/2}\mathbf{u}\rangle \tilde{\mathbf{H}}^{-1/2}\mathbf{u}] \\
&= \mathbb{E}_{\tilde{\mathbf{u}}\sim\mathcal{N}(\mathbf{0},\tilde{\mathbf{H}}^{-1})}\left[\langle \nabla f(\mathbf{x}),\, \tilde{\mathbf{u}}\rangle \tilde{\mathbf{u}}\right] \\
&\approx \mathbb{E}_{\tilde{\mathbf{u}}\sim\mathcal{N}(\mathbf{0},\tilde{\mathbf{H}}^{-1})}\left[\frac{f(\mathbf{x}+\epsilon\tilde{\mathbf{u}}) - f(\mathbf{x})}{\epsilon}\tilde{\mathbf{u}}\right]. \quad (3)
\end{aligned}$$

Equation (3) demonstrates that preconditioning the gradient in ZO optimization is equivalent to drawing the perturbation vector $\tilde{\mathbf{u}}$ from an anisotropic Gaussian distribution whose covariance matrix $\boldsymbol{\Sigma}$ equals the inverse of curvature matrix $\tilde{\mathbf{H}}$, i.e., $\boldsymbol{\Sigma} = \tilde{\mathbf{H}}^{-1} = \epsilon^2\overline{\boldsymbol{\Sigma}}$. We estimate the gradient in (3) using the framework of evolution strategies (ES) (Rechenberg 1973):

$$\begin{aligned}
\arg\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &:= \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z};\boldsymbol{\theta})}[f(\mathbf{z})] \\
&= \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)}\left[f(\mathbf{x} + \epsilon\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u})\right],
\end{aligned}$$

where ϵ is a smoothing parameter and $p(\mathbf{z};\boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}, \epsilon^2\overline{\boldsymbol{\Sigma}})$ is the search distribution whose mean is the current solution (i.e., model parameters) $\mathbf{x}$ and the covariance matrix $\epsilon^2\overline{\boldsymbol{\Sigma}}$ models the inverse of the curvature matrix. The gradient of $J(\boldsymbol{\theta})$ can be calculated using the score function estimator (also known as the REINFORCE estimator (Williams 1992)), given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z};\boldsymbol{\theta})}\left[f(\mathbf{z}) \nabla_{\boldsymbol{\theta}} \log p(\mathbf{z};\boldsymbol{\theta})\right]. \quad (4)$$

4.2 Low-Rank Structured Covariance Matrices

Consider a network layer with parameters $\mathbf{x} = \mathrm{vec}(\mathbf{X}) \in \mathbb{R}^{mn}$, where $\mathbf{X} \in \mathbb{R}^{m\times n}$. While the ES framework allows capturing local curvature information, it requires maintaining and updating the covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{mn\times mn}$, which can incur prohibitive memory cost, particularly for LLMs. Second-order optimizers such as Shampoo (Gupta, Koren, and Singer 2018) and KFAC (Martens and Grosse 2015) exploit Kronecker-factored curvature approximations to efficiently estimate the curvature matrix using significantly smaller memory than storing the full matrix. Recent studies (Sankar et al. 2021; Yang, Mao, and Chaudhari 2022; Seung, Lee, and Ko 2025) have shown that the Hessian and Fisher Information matrix (FIM) of deep neural networks exhibit inherent low-rank structure. Motivated by these, we propose to estimate the curvature matrix $\tilde{\mathbf{H}} = \nabla^2_{\mathbf{x}} f(\mathbf{x}) \in \mathbb{R}^{mn\times mn}$ by

$$\tilde{\mathbf{H}} = \mathbf{I}_m \otimes (\rho\mathbf{I}_n + \mathbf{a}\mathbf{a}^\intercal), \quad (5)$$

where $\rho > 0$ is a damping factor and $\mathbf{a} \in \mathbb{R}^n$ is a learnable vector that parameterizes the curvature matrix. The damped rank-1 block-diagonal approximation admits a closed-form solution for both the inverse $\tilde{\mathbf{H}}^{-1}$ and the inverse square root $\tilde{\mathbf{H}}^{-1/2}$, enabling efficient implementation. As shown in 4.1, we leverage the curvature information directly by setting $\boldsymbol{\Sigma} = \tilde{\mathbf{H}}^{-1}$ and draw the perturbation $\tilde{\mathbf{u}} = \boldsymbol{\Sigma}^{1/2}\mathbf{u}$, where $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{mn})$,

$$\boldsymbol{\Sigma} = \mathbf{I}_m \otimes (\rho\mathbf{I}_n + \mathbf{a}\mathbf{a}^\intercal)^{-1} = \mathbf{I}_m \otimes \frac{1}{\rho}\left(\mathbf{I}_n - \frac{\mathbf{a}\mathbf{a}^\intercal}{\rho + \|\mathbf{a}\|^2}\right), \quad (6)$$

$$\boldsymbol{\Sigma}^{1/2} = \mathbf{I}_m \otimes \frac{1}{\sqrt{\rho}}\left(\mathbf{I}_n - \frac{\sqrt{\rho} + \sqrt{\rho + \|\mathbf{a}\|^2}}{\|\mathbf{a}\|^2\sqrt{\rho + \|\mathbf{a}\|^2}}\mathbf{a}\mathbf{a}^\intercal\right), \text{ and} \quad (7)$$

$$\mathrm{d}\boldsymbol{\Sigma} = \mathbf{I}_m \otimes \left(-\frac{(\mathrm{d}\mathbf{a})\mathbf{a}^\intercal + \mathbf{a}(\mathrm{d}\mathbf{a}^\intercal)}{\rho(\rho + \|\mathbf{a}\|^2)} + \frac{2\mathbf{a}^\intercal(\mathrm{d}\mathbf{a})\mathbf{a}\mathbf{a}^\intercal}{\rho(\rho + \|\mathbf{a}\|^2)^2}\right). \quad (8)$$

4.3 Search Distribution Parameter Updates

Let $p(\mathbf{z};\boldsymbol{\theta})$ denote the search distribution, modeled as a multivariate Gaussian $\mathcal{N}(\mathbf{x}, \boldsymbol{\Sigma})$ in LOREN, where $\boldsymbol{\theta} = [\mathbf{x}^\intercal, \mathrm{vec}(\boldsymbol{\Sigma})^\intercal]^\intercal$ and $\boldsymbol{\Sigma}$ is defined as in (6). Our goal is to update the parameters $\boldsymbol{\theta}$ of search distribution $p(\mathbf{z};\boldsymbol{\theta})$ such that the expected loss $J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z};\boldsymbol{\theta})}[f(\mathbf{z})]$ of the underlying model is minimized. Let $\mathcal{L} = \log p(\mathbf{z};\boldsymbol{\theta})$. The differential of $\mathcal{L}$ is given by:

$$\mathrm{d}\mathcal{L} = \frac{1}{2}\boldsymbol{\Sigma}^{-1}(\mathbf{Z} - \boldsymbol{\Sigma})\boldsymbol{\Sigma}^{-1} : \mathrm{d}\boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \mathbf{x}) : \mathrm{d}\mathbf{x},$$

where $\mathbf{Z} = (\mathbf{z} - \mathbf{x})(\mathbf{z} - \mathbf{x})^\intercal$ and $\mathrm{d}\mathbf{Z} = -\mathrm{d}\mathbf{x}(\mathbf{z} - \mathbf{x})^\intercal - (\mathbf{z} - \mathbf{x})(\mathrm{d}\mathbf{x})^\intercal$. Applying (6) and (8) gives

$$\begin{aligned}
\mathrm{d}\mathcal{L} &= (\mathbf{Z} - \boldsymbol{\Sigma}) : (\mathbf{I}_m \otimes -(\mathrm{d}\mathbf{a})\mathbf{a}^\intercal)) \\
&\quad + (\mathbf{Z} - \boldsymbol{\Sigma}) : \left(\mathbf{I}_m \otimes \left(-\frac{\mathbf{a}^\intercal\,\mathrm{d}\mathbf{a}}{\rho}\mathbf{a}\mathbf{a}^\intercal\right)\right) \\
&\quad + \frac{1}{2}(\mathbf{Z} - \boldsymbol{\Sigma}) : \left(\mathbf{I}_m \otimes \left(\frac{2\mathbf{a}^\intercal\,\mathrm{d}\mathbf{a}}{\rho}\mathbf{a}\mathbf{a}^\intercal\right)\right) \\
&\quad + \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \mathbf{x}) : \mathrm{d}\mathbf{x} \\
&= -\sum_{i=1}^{m}(\mathbf{Z} - \boldsymbol{\Sigma})_{ii}\mathbf{a} : \mathrm{d}\mathbf{a} + \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \mathbf{x}) : \mathrm{d}\mathbf{x}, \quad (9)
\end{aligned}$$

where the last equality is due to Proposition 4.1.

Proposition 4.1. *Let $\mathbf{A} \in \mathbb{R}^{mn\times mn}$ be a symmetric matrix, and $\mathbf{B} \in \mathbb{R}^{n\times n}$. Then we have $\mathrm{tr}(\mathbf{A}(\mathbf{I}_m \otimes \mathbf{B})) = \mathrm{tr}((\sum_{i=1}^{m}\mathbf{A}_{ii})\mathbf{B})$, where $\mathbf{A}_{ij}$ denotes the submatrix located at the (i,j)-th block position when $\mathbf{A}$ is viewed as an $m \times m$ block matrix with each block of size $n \times n$.*

From (9), we obtain

$$\begin{aligned}
\nabla_{\mathbf{x}} \log p(\mathbf{z};\boldsymbol{\theta}) &= \boldsymbol{\Sigma}^{-1}(\mathbf{z} - \mathbf{x}), \\
\nabla_{\mathbf{a}} \log p(\mathbf{z};\boldsymbol{\theta}) &= \sum_{i=1}^{m}(\mathbf{Z} - \boldsymbol{\Sigma})_{ii}\mathbf{a}.
\end{aligned}$$

Applying the reparameterization trick $\mathbf{z} = \mathbf{x} + \boldsymbol{\Sigma}^{1/2}\mathbf{u}$, where $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{mn})$, and (7) yields

$$\nabla_{\mathbf{a}} \log p(\mathbf{z}; \boldsymbol{\theta}) = \sum_{i=1}^{m} (\boldsymbol{\Sigma}^{1/2}(\mathbf{I}_{mn} - \mathbf{u}\mathbf{u}^\intercal)\boldsymbol{\Sigma}^{1/2})_{ii}\mathbf{a} = \sum_{i=1}^{m} \frac{\left(\mathbf{M}_i\mathbf{a} - \kappa(\mathbf{a}^\intercal\mathbf{M}_i\mathbf{a})\mathbf{a}\right)}{\sqrt{\rho}\sqrt{\rho + \|\mathbf{a}\|^2}}, \quad (10)$$

where $\kappa = \left(\sqrt{\rho} + \sqrt{\rho + \|\mathbf{a}\|^2}\right) / \left(\|\mathbf{a}\|^2\sqrt{\rho + \|\mathbf{a}\|^2}\right)$, $\mathbf{M}_i = \mathbf{u}_i\mathbf{u}_i^\intercal - \mathbf{I}_n$, and $\mathbf{u}_i \in \mathbb{R}^n$ denotes the i^{th} subvector of $\mathbf{u} \in \mathbb{R}^{mn}$ from index $(i-1)n+1$ to in, for $i = 1, \ldots, m$. The FIM $\mathbf{F}_{\mathbf{x}}$ has a simple form and is given by

$$\mathbf{F}_{\mathbf{x}} = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z};\boldsymbol{\theta})}[\nabla_{\mathbf{x}} \log p(\mathbf{z}; \boldsymbol{\theta}) \nabla_{\mathbf{x}} \log p(\mathbf{z}; \boldsymbol{\theta})^\intercal] = \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_{mn})}[\boldsymbol{\Sigma}^{-1/2}\mathbf{u}\mathbf{u}^\intercal\boldsymbol{\Sigma}^{-1/2}] = \boldsymbol{\Sigma}^{-1}.$$

Thus, the natural gradient w.r.t. $\mathbf{x}$ is given by

$$\mathbf{F}_{\mathbf{x}}^{-1}\nabla_{\mathbf{x}} \log p(\mathbf{z}; \boldsymbol{\theta}) = \boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}(\mathbf{z} - \mathbf{x}) = \boldsymbol{\Sigma}^{1/2}\mathbf{u}. \quad (11)$$

We now derive the score function estimates of ZO gradient $\hat{\nabla}_{\mathbf{x}} f(\mathbf{x})$ and $\hat{\nabla}_{\mathbf{a}} f(\mathbf{x})$. Using the fact that the gradient of Gaussian smoothed $f(\mathbf{x})$ corresponds to the SPSA estimator as given in (1), we obtain

$$\hat{\nabla}_{\mathbf{x}} f(\mathbf{x}) = \nabla_{\mathbf{x}} \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_d)} \left[f(\mathbf{x} + \epsilon\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u})\right] = \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_{mn})} \left[f(\mathbf{x} + \epsilon\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u})\epsilon^{-1}\overline{\boldsymbol{\Sigma}}^{-1/2}\mathbf{u}\right].$$

From (11), the natural gradient $\mathbf{F}_{\mathbf{x}}^{-1}\hat{\nabla}_{\mathbf{x}} f(\mathbf{x})$ is given by

$$\mathbf{F}_{\mathbf{x}}^{-1}\hat{\nabla}_{\mathbf{x}} f(\mathbf{x}) = \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_{mn})} \left[f(\mathbf{x} + \epsilon\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u})\epsilon^{-1}\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u}\right],$$

where, from (7),

$$\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u} = \frac{1}{\epsilon}\left\{\mathbf{I}_m \otimes \frac{1}{\sqrt{\rho}}\left(\mathbf{I}_n - \kappa\mathbf{a}\mathbf{a}^\intercal\right)\right\}\mathbf{u} = \frac{1}{\epsilon}\sum_{i=1}^{m} \frac{1}{\sqrt{\rho}}\left(\mathbf{I}_n - \kappa\mathbf{a}\mathbf{a}^\intercal\right)\mathbf{u}_i. \quad (12)$$

Similarly, we have

$$\hat{\nabla}_{\mathbf{a}} f(\mathbf{x}) = \mathbb{E}_{\mathbf{u}\sim\mathcal{N}(\mathbf{0},\mathbf{I}_{mn})} \left[f(\mathbf{x} + \epsilon\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u})\nabla_{\mathbf{a}} \log p(\mathbf{z}; \boldsymbol{\theta})\right],$$

where $\nabla_{\mathbf{a}} \log p(\mathbf{z}; \boldsymbol{\theta})$ is given by (10).

4.4 Algorithm

The key steps of LOREN are summarized in the pseudocode presented in Algorithm 1. The algorithm begins by sampling K perturbation vectors $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, where the dimensionality $d = mn$ for layers with matrix-valued parameters and $d = m$ for layers with vector-valued parameters. The loss function f is then evaluated using the scaled perturbations $\epsilon\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u}_k$ as shown in (12) (Lines 4–5). In Lines 7–8, LOREN applies the RLOO estimator to compute the variance reduced gradients w.r.t. both the mean $\mathbf{x}$ and covariance parameters $\mathbf{a}$. Using the gradient estimates, Lines 9–10 simultaneously update $\mathbf{x}$ and $\mathbf{a}$.

Algorithm 1: LOREN

Input: Dataset $S = \{\xi_1, \ldots, \xi_n\}$, Initialization $\mathbf{x}_0 \in \mathbb{R}^d$, $\mathbf{a}_0 \sim \mathcal{N}(0, \mathbf{I}_d)$, number of iterations T, learning rates $\{\eta, \nu\}$, smoothing ϵ, damping ρ, number of forward passes K

1 **for** $t = 0$ **to** $T - 1$ **do**
2 Sample mini-batch $\mathcal{B}_t$
3 **for** $k = 1$ **to** K **do**
4 Sample $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$
5 $f^k \leftarrow f(\mathbf{x}_t + \epsilon\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u}_k; \mathcal{B}_t)$
6 **end**
7 $\mathbf{g}(\mathbf{x}_t) \leftarrow \frac{1}{\epsilon(K-1)}\sum_{k=1}^{K}\left(f^k - \frac{1}{K}\sum_{j=1}^{K} f^j\right)\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u}_k$
8 $\mathbf{g}(\mathbf{a}_t) \leftarrow \frac{1}{K-1}\sum_{k=1}^{K}\left(f^k - \frac{1}{K}\sum_{j=1}^{K} f^j\right)\nabla_{\mathbf{a}} \log p(\mathbf{z}_k; \boldsymbol{\theta})$, where $\mathbf{z}_k = \mathbf{x}_k + \epsilon\overline{\boldsymbol{\Sigma}}^{1/2}\mathbf{u}_k$
9 $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta\mathbf{g}(\mathbf{x}_t)$
10 $\mathbf{a}_{t+1} \leftarrow \mathbf{a}_t - \nu\mathbf{g}(\mathbf{a}_t)$
11 **end**
12 **return** $\mathbf{x}_T$

Method	MeZO-Adam	MeZO-SVRG	LOZO	HiZOO	LOREN
Cost	$\mathcal{O}(mn)$	$\mathcal{O}(mn)$	$\mathcal{O}(nr)$	$\mathcal{O}(mn)$	$\mathcal{O}(n)$

Table 1: Additional memory requirement compared to MeZO.

Memory Complexity Table 1 shows the additional memory overhead of MeZO variants relative to MeZO. MeZO-Adam, MeZO-SVRG, and HiZOO each require $\mathcal{O}(mn)$ extra space, while LOZO incurs $\mathcal{O}(nr)$ overhead to store low-rank gradient components, where r is the rank. In contrast, LOREN reduces the memory cost to $\mathcal{O}(n)$ by maintaining only low-rank covariance components. This memory efficiency enables the use of heavyball momentum (Polyak 1964), from which LOREN benefits through acceleration. We refer to this momentum variant simply as LOREN in Section 5.

4.5 Convergence Analysis

The following theorem shows that LOREN can converge to a stationary point at a rate of $\mathcal{O}(1/\sqrt{T})$.

Theorem 4.2. *Assuming the L-smoothness of the objective function f and bounded variance of gradient estimates, the sequence of parameter vectors $\{\mathbf{x}_t\}$ generated by Algorithm 1 with $\eta = \frac{1}{8\sqrt{T}L(\max_t \operatorname{tr}(\boldsymbol{\Sigma}_t) + 2\rho^{-1})}$ satisfies*

$$\min_{t=1:T} \mathbb{E}\left[\|\nabla f(\mathbf{x}_t; \xi_t)\|^2\right] \leq \frac{32L(\max_t \operatorname{tr}(\boldsymbol{\Sigma}_t) + 2\rho^{-1})\left(f(\mathbf{x}_1; \xi_1) - f(\mathbf{x}_*; \xi_*)\right)}{\sqrt{T}\alpha_{\min}} + \frac{\sigma^2}{\sqrt{T}\alpha_{\min}} + \mathcal{O}(\epsilon^2),$$

	DistilBERT (66M) — FP32				
Task	MNLI	QNLI	SST-2	CoLA	Avg
MeZO	39.9±0.2	48.5±0.6	62.1±0.2	67.0±0.4	54.4
MeZO-Adam	41.2±1.5	71.1±2.2	78.4±1.7	67.8±1.9	64.6
MeZO-SVRG	42.0±1.5	64.0±2.4	73.6±2.7	66.6±0.9	61.6
LOZO	40.2±0.3	53.0±1.4	61.0±1.5	67.0±0.6	55.3
HiZOO	40.0±0.2	64.5±8.0	78.7±0.9	67.0±1.0	62.6
LOREN	39.8±0.0	73.0±2.0	81.7±1.0	67.2±0.8	**65.4**
	RoBERTa-large (355M) — FP32				
Task	MNLI	QNLI	SST-2	CoLA	Avg
MeZO	39.8±0.3	71.6±1.5	54.8±0.6	67.2±0.3	58.4
MeZO-Adam	51.6±1.5	80.1±1.8	84.8±3.7	77.9±1.5	**73.6**
MeZO-SVRG	39.8±0.3	59.1±6.7	55.2±0.5	67.3±0.5	54.6
LOZO	41.3±2.2	70.8±1.7	54.6±0.5	69.5±1.8	59.1
HiZOO	43.1±0.9	70.2±2.5	73.2±5.5	70.2±1.4	64.2
LOREN	44.3±1.4	76.3±1.5	86.1±3.1	73.8±0.4	70.1

Table 2: Experimental results on DistilBERT and RoBERTa. Reported metrics include best accuracy (%) with standard deviation over 5 runs and the averaged accuracy across 4 benchmark tasks from GLUE.

where $\alpha_{\min} = (\rho + \max_t \|\mathbf{a}_t\|^2)^{-1}$ *is the smallest eigenvalue of* $\mathbf{\Sigma}_t$.

5 Experiments

We evaluate LOREN on masked and autoregressive language models using both GLUE and SuperGLUE benchmarks, comparing it with MeZO, MeZO-Adam, MeZO-SVRG, LOZO, and HiZOO. For optimal performance, LOREN employs six forward evaluations per iteration with RLOO variance reduction. The same evaluation budget is applied to all baselines to ensure a fair and consistent comparison. While other ZO optimizers typically use only two or three forward passes per step, this default setting leads to degraded performance compared to using six passes. We conduct full-parameter fine-tuning of LLMs without using prompts, following the more challenging setting from (Gautam et al. 2024), in contrast to the prompt fine-tuning setup used in (Malladi et al. 2023; Chen et al. 2025; Zhao et al. 2025), in order to better highlight the performance gap between ZO optimizers. Early stopping (Prechelt 1996) was employed to prevent over-iteration, as ZO optimizers tend to yield diminishing performance gains once convergence is reached. All experiments were conducted on a single NVIDIA H100 or A100 GPU.

5.1 Masked Language Models

LOREN consistently improves performance across masked language models. As shown in Table 2, LOREN achieves the highest average accuracy on DistilBERT, outperforming both MeZO and LOZO by more than 10 percentage points (%p). On RoBERTa-large, LOREN ranks second overall, closely trailing MeZO-Adam, while surpassing MeZO, MeZO-SVRG, and LOZO by over 10%p, and HiZOO by 6%p.

	GPT-2-XL (1.5B) — FP32				
Task	MNLI	QNLI	SST-2	CoLA	Avg
MeZO	39.1±1.1	58.8±0.2	73.8±0.8	65.4±0.2	59.3
MeZO-Adam	50.9±1.3	72.3±4.3	91.2±0.6	71.6±0.8	71.5
MeZO-SVRG	49.4±1.0	65.2±1.0	84.0±1.6	65.8±0.2	66.1
LOZO	42.1±0.7	60.0±1.3	79.4±1.0	65.6±0.3	61.8
HiZOO	48.6±0.2	66.3±3.6	89.6±0.2	71.5±0.8	69.0
LOREN	51.2±0.3	74.6±1.2	89.8±0.8	72.0±0.7	**71.9**
	OPT-2.7B — FP32				
Task	MNLI	QNLI	SST-2	CoLA	Avg
MeZO	50.2±3.3	75.2±1.0	88.9±0.2	68.4±2.3	70.7
MeZO-Adam	45.7±1.2	72.0±3.1	86.7±2.7	68.2±1.0	68.2
MeZO-SVRG	41.2±0.2	59.4±0.8	63.5±1.8	66.2±0.2	57.6
LOZO	42.7±1.0	59.8±0.8	66.4±3.1	66.9±0.2	59.0
HiZOO	50.4±1.2	75.4±0.8	88.1±0.6	66.0±0.0	70.0
LOREN	55.1±0.7	73.8±1.6	89.5±0.8	68.0±0.0	**71.6**
	LLaMA-3-8B — BF16				
Task	RTE	BoolQ	WiC	CB	Avg
MeZO	59.6±1.5	64.2±0.8	59.2±1.9	71.4±1.5	63.6
MeZO-Adam	58.6±1.0	63.7±0.0	57.7±1.0	68.5±0.8	62.1
MeZO-SVRG	58.2±1.4	63.8±0.2	58.0±1.1	65.7±2.9	61.4
LOZO	57.0±1.2	64.8±0.6	57.2±2.1	71.4±4.4	62.6
HiZOO	57.8±0.6	63.7±0.0	59.2±1.5	66.7±3.7	61.8
LOREN	58.6±1.0	64.8±1.2	59.2±1.2	72.2±2.7	**63.7**
	OPT-13B — BF16				
Task	RTE	BoolQ	WiC	CB	Avg
MeZO	59.2±0.5	64.3±1.0	57.2±0.4	73.1±1.6	63.4
MeZO-Adam	60.8±1.4	63.4±1.3	55.7±1.2	70.2±1.7	62.5
MeZO-SVRG	59.1±0.9	65.2±0.7	57.3±0.9	69.3±2.8	62.7
LOZO	58.5±0.7	63.2±0.2	57.6±1.8	68.5±3.0	62.0
HiZOO	59.8±1.3	63.4±0.4	57.4±1.0	72.6±1.7	63.3
LOREN	60.0±0.7	63.2±0.6	57.2±0.8	73.7±1.3	**63.5**

Table 3: Experimental results on GPT-2, OPT, and LLaMA-3. Reported metrics include best accuracy (%) with standard deviation over 5 runs and the average accuracy across 4 benchmark tasks from GLUE and SuperGLUE.

5.2 Autoregressive Language Models

LOREN delivers the best overall accuracy across all model architectures. As shown in Table 3, on GPT-2-XL LOREN slightly improves over MeZO-Adam and outperforms MeZO-SVRG and HiZOO by more than 5%p. On OPT-2.7B, it leads MeZO-Adam by over 3%p and MeZO-SVRG and LOZO by over 10%p. Across SuperGLUE tasks with LLaMA-3 and OPT-13B, LOREN again achieves the highest average accuracy, consistently surpassing every baseline. We exclude MeZO-SVRG from our SuperGLUE fine-tuning because its performance gains are limited and its implementation runs considerably slower on large models.

LOREN achieves the fastest loss minimization. The

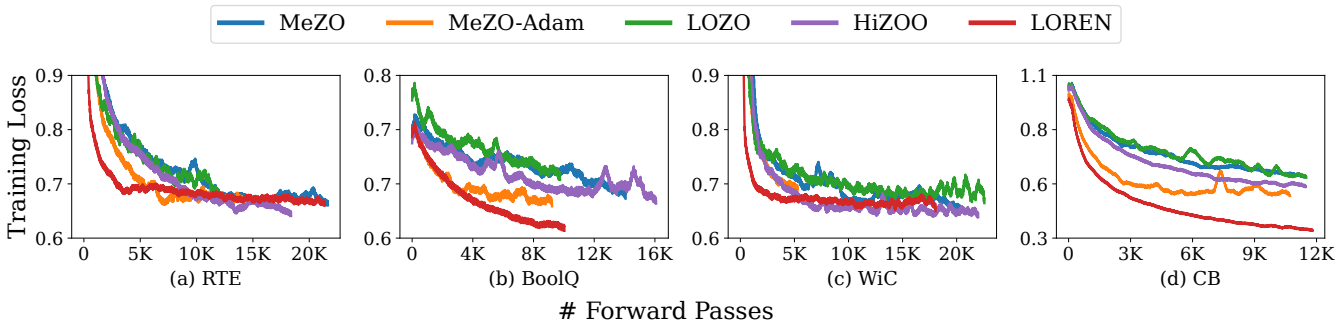

Figure 2: Training loss curves for different ZO optimizers when fine-tuning OPT-13B on SuperGLUE tasks.

Models	DistilBERT	RoBERTa-large	GPT-2-XL	OPT-2.7B	LLaMA-3-8B	OPT-13B
MeZO	0.85	2.14	16.9	17.4	18.4	32.9
MeZO-Adam	1.36 (1.60×)	4.83 (2.26×)	28.8 (1.70×)	37.3 (2.14×)	46.2 (2.51×)	76.0 (2.31×)
MeZO-SVRG	1.18 (1.39×)	4.25 (1.99×)	32.3 (1.91×)	32.6 (1.87×)	44.3 (2.41×)	74.7 (2.27×)
LOZO	0.76 (0.89×)	2.07 (0.97×)	16.8 (0.99×)	16.8 (0.97×)	17.4 (0.95×)	32.8 (1.00×)
HiZOO	1.43 (1.68×)	4.49 (2.10×)	24.3 (1.44×)	29.3 (1.69×)	36.5 (1.98×)	59.6 (1.81×)
LOREN	1.15 (1.35×)	3.73 (1.74×)	23.1 (1.37×)	27.6 (1.59×)	33.6 (1.83×)	57.5 (1.75×)

Table 4: Peak GPU memory consumption (GB) and relative usage (MeZO = 1.00). LLaMA-3-8B and OPT-13B were trained using half-precision (BF16).

training loss curves in Figure 2 confirms LOREN 's markedly faster convergence. For OPT-13B fine-tuned on SuperGLUE, LOREN demonstrates the most rapid loss reduction among ZO methods, ultimately reaching substantially lower final losses on BoolQ and CB compared to all baselines.

5.3 Memory and Training Efficiency

LOREN still maintains the affordable memory usage. As shown in Table 4, even after integrating variance reduction, curvature-aware updates, and momentum, LOREN still requires less memory than MeZO-Adam, MeZO-SVRG, and HiZOO. Although it does not match the minimal usage of MeZO or LOZO, LOREN achieves a favorable trade-off. Across six architectures, LOREN's peak memory consumption ranges from 1.35× to 1.83× that of the MeZO baseline, compared to 1.68× – 2.10× for HiZOO, 1.60× – 2.51× for MeZO-Adam, and 1.39× – 2.41× for MeZO-SVRG. These results indicate that the additional memory overhead introduced by LOREN remains relatively modest in comparison to other MeZO variants.

LOREN exhibits the highest query efficiency. Table 5 reports the number of forward passes and wall-clock time needed to reach a target accuracy when fine-tuning GPT-2-XL on SST-2 and LLaMA-3-8B on CB. We set the targets based on the lowest accuracy achieved by any ZO optimizer in each setting. In both benchmarks, LOREN requires the fewest forward-pass queries to reach the targets. In terms of wall-clock time, LOREN consistently ranks as the second fastest method, closely matching MeZO-Adam on GPT-2-XL fine-tuned for SST-2, and trailing only LOZO on LLaMA-3-8B fine-tuned for CB, while maintaining a clear runtime advantage over all other ZO methods.

Models	GPT-2-XL		LLaMA-3-8B	
	# Queries	Time	# Queries	Time
MeZO	16,752±816	3.13±0.2	5,736±1,476	0.80±0.2
MeZO-Adam	1,632±144	**0.32±0.0**	6,894±894	0.96±0.1
MeZO-SVRG	4,248±648	1.51±0.2	3,216±1,491	0.53±0.2
LOZO	10,686±1,368	0.66±0.1	3,198±888	**0.15±0.0**
HiZOO	2,232±216	0.98±0.1	4,356±1,932	0.76±0.3
LOREN	**1,320±72**	0.33±0.0	**1,512±486**	0.47±0.1

Table 5: Number of forward passes and wall-clock time (hours) required to reach 70% accuracy on SST-2 with GPT-2-XL and 65% accuracy on CB with LLaMA-3-8B. All values are reported as mean ± standard deviation over 5 independent runs.

6 Conclusions

In this work, we proposed LOREN, the first ZO preconditioned method specifically designed to address the heterogeneous curvature problem in LLM fine-tuning by learning an anisotropic random perturbation distribution. By combining the Kronecker-factored low-rank approximation of curvature information with variance-reduced RLOO gradient estimates, LOREN effectively adapts to the geometry of complex loss landscapes in a memory-efficient manner. Empirical evaluations on LLM fine-tuning tasks demonstrate that LOREN consistently achieves higher accuracy and faster convergence across various models while maintaining a lower memory consumption compared to state-of-the-art ZO methods.

Acknowledgments

This research was supported by the National Science Foundation under Grant No. 1943046.

References

Balasubramanian, K.; and Ghadimi, S. 2018. Zeroth-Order Nonconvex Stochastic Optimization: Handling Constraints, High Dimensionality, and Saddle Points. *Foundations of Computational Mathematics*, 22.

Chen, Y.; yuan zhang; Cao, L.; Yuan, K.; and Wen, Z. 2025. Enhancing Zeroth-order Fine-tuning for Language Models with Low-rank Structures. In *International Conference on Learning Representations*.

Duchi, J. C.; Jordan, M. I.; Wainwright, M. J.; and Wibisono, A. 2015. Optimal Rates for Zero-Order Convex Optimization: The Power of Two Function Evaluations. *IEEE Transactions on Information Theory*.

Gautam, T.; Park, Y.; Zhou, H.; Raman, P.; and Ha, W. 2024. Variance-reduced Zeroth-Order Methods for Fine-Tuning Language Models. In *International Conference on Learning Representations*.

Ghadimi, S.; and Lan, G. 2013. Stochastic First- and Zeroth-Order Methods for Nonconvex Stochastic Programming. *SIAM Journal on Optimization*, 23: 2341–2368.

Gupta, V.; Koren, T.; and Singer, Y. 2018. Shampoo: Preconditioned Stochastic Tensor Optimization. In *International Conference on Machine Learning*.

Johnson, R.; and Zhang, T. 2013. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. In *Neural Information Processing Systems*.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Buy 4 REINFORCE Samples, Get a Baseline for Free! In *DeepRLStructPred@ICLR*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Malladi, S.; Gao, T.; Nichani, E.; Damian, A.; Lee, J. D.; Chen, D.; and Arora, S. 2023. Fine-Tuning Language Models with Just Forward Passes. In *Neural Information Processing Systems*.

Martens, J.; and Grosse, R. 2015. Optimizing neural networks with Kronecker-factored approximate curvature. In *International Conference on Machine Learning*.

Nesterov, Y.; and Spokoiny, V. G. 2017. Random Gradient-Free Minimization of Convex Functions. *Foundations of Computational Mathematics*, 17: 527 – 566.

Polyak, B. T. 1964. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5): 1–17.

Prechelt, L. 1996. Early Stopping-But When? In *Neural Networks*.

Rechenberg, I. 1973. Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.

Robbins, H.; and Monro, S. 1951. A Stochastic Approximation Method. *The annals of mathematical statistics*, 400–407.

Sankar, A. R.; Khasbage, Y.; Vigneswaran, R.; and Balasubramanian, V. N. 2021. A Deeper Look at the Hessian Eigenspectrum of Deep Neural Networks and Its Applications to Regularization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(11): 9481–9488.

Seung, H.; Lee, J.; and Ko, H. 2025. MAC: An Efficient Gradient Preconditioning using Mean Activation Approximated Curvature. In *IEEE International Conference on Data Mining*.

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

Wang, A.; Pruksachatkun, Y.; Nangia, N.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2019. SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems. *Neural Information Processing Systems*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. R. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *BlackboxNLP@EMNLP*.

Wierstra, D.; Schaul, T.; Peters, J.; and Schmidhuber, J. 2008. Natural Evolution Strategies. *IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 3381–3387.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8: 229–256.

Yang, R.; Mao, J.; and Chaudhari, P. 2022. Does the Data Induce Capacity Control in Deep Learning? In *Proceedings of the 39th International Conference on Machine Learning*, 25166–25197. PMLR.

Ye, H.; Huang, Z.; Fang, C.; Li, C. J.; and Zhang, T. 2018. Hessian-Aware Zeroth-Order Optimization for Black-Box Adversarial Attack. *IEEE transactions on pattern analysis and machine intelligence*.

Zhao, Y.; Dang, S.; Ye, H.; Dai, G.; Qian, Y.; and Tsang, I. 2025. Second-Order Fine-Tuning without Pain for LLMs: A Hessian Informed Zeroth-Order Optimizer. In *International Conference on Learning Representations*.