GANSZO: GAUSS-NEWTON STEP-SIZED ZEROTH-ORDER OPTIMIZATION FOR MEMORY-EFFICIENT LLM FINE-TUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning large language models at scale is increasingly constrained by the memory cost of backpropagation. Zeroth-order (ZO) optimizers (e.g., MeZO) alleviate this bottleneck by relying solely on forward passes, but they often suffer from slow convergence. Recent works such as HiZOO utilize diagonal Hessian estimators as preconditioner for acceleration, yet this incurs additional memory costs comparable to training parameters. Moreover, the noisy diagonal Hessian estimates often lead to training instabilities. In this work, we introduce the **Ga**uss-**N**ewton **s**tepsized **Z**eroth-**O**rder (GaNsZO) algorithm, a *memory- and computation-efficient* ZO optimizer. GaNsZO estimates the directional second derivative along the descent direction and uses it as a stepsize scaling factor, reducing memory overhead while improving robustness. To estimate efficiently, GaNsZO exploits the model's composite structure and computes Gauss-Newton quadratic form along that direction, saving one forward pass compared to standard central-difference schemes. Empirically, on OPT and Llama models, GaNsZO consistently outperforms HiZOO in terms of accuracy and training stability. In certain tasks (e.g., SST-2), GaNsZO approaches the performance of first-order fine-tuning. In terms of efficiency, GaNsZO reduces memory consumption by up to 43% compared to HiZOO, while requiring significantly fewer forward passes to reach comparable loss, leading to up to $6\times$ end-to-end speedup in selected settings. Finally, we show that GaNsZO combines naturally with PEFT and recent ZO techniques (e.g., LOZO) and produces further improvements.

1 INTRODUCTION

Fine-tuning large language models (LLMs) has become a central component in adapting large pretrained models to specific downstream tasks Raffel et al. (2020). While first-order optimizers such as Adam Kingma & Ba (2015) and AdamW Loshchilov & Hutter (2019) are commonly used in fine-tuning, they require full backpropagation through the model and thus incur substantial memory overhead as the activations have to be stored. These limitations have driven increasing interest in parameter- and memory-efficient approaches Hu et al. (2022); Ding et al. (2023); Zhao et al. (2024). Zeroth-order (ZO) optimization Nesterov & Spokoiny (2017), exemplified by MeZO Malladi et al. (2023), provides a promising alternative by replacing gradient computation with forward-pass-only perturbation estimates. As a result, MeZO reduces memory consumption to the level of inference, making large-scale fine-tuning feasible under tight memory budgets.

Despite these advantages, ZO methods typically exhibit slow convergence due to the noisy gradient estimates in high-dimensional, ill-conditioned loss landscapes. To accelerate MeZO, recent work has proposed Hessian-informed zeroth-order optimizers such as HiZOO Zhao et al. (2025b). These methods inject curvature information—commonly via diagonal Hessian approximations—to accelerate training. While effective, they introduce new practical challenges. First, tracking moving averages of diagonal Hessian estimates requires storing a value per parameter, resulting in $O(\#\text{parameters})$ memory overhead that undermines one of the primary benefits of ZO algorithms. Second, Hessian estimation in high dimension is notoriously noisy. The large variance in Hessian estimation can induce significant training instabilities, forcing practitioners to rely on delicate hyperparameter selection or additional heuristics such as clipping Zhao et al. (2025a); Liu et al. (2024) to stabilize the estimator.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Besides, most Hessian-informed ZO methods use central-difference estimators for second derivatives, which requires three forward passes per iteration and thus increases per-step cost. This motivates the following question: *Can we robustly incorporate second-order information into ZO with minimal memory and computation overhead?*

We begin by examining the behavior of true Hessian and the diagonal Hessian estimators used in HiZOO. We find that the empirical distribution of estimated diagonal entries deviates substantially from that of true diagonal Hessian and attribute this to the inherent curvature structure of LLMs: most LLM diagonal Hessian entries are near zero, while a small fraction dominate in magnitude. Stabilizing diagonal estimation therefore requires conservative initialization and small exponential moving average (EMA) coefficients, which inevitably limit the estimator's ability to track true diagonal Hessian. On the other hand, we find that the estimated diagonal Hessian in HiZOO concentrates near a single scalar, suggesting that fully estimating the diagonal may be unnecessary. Taken together, these findings motivate shifting focus from costly per-parameter preconditioning to a stepsize-centric strategy that exploits second-order information.

In this paper, we introduce **Ga**uss-**N**ewton **s**tepsized **Z**eroth-**O**rder (GaNsZO) optimizer, a *memory- and computation-efficient* ZO optimizer. The main contributions are:

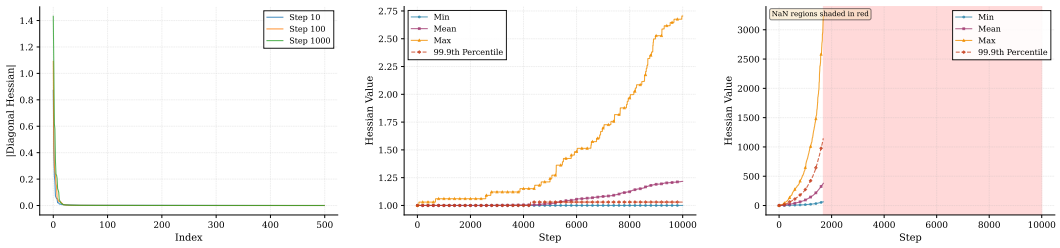
1. GaNsZO avoids storing a full diagonal Hessian and instead estimates the curvature only along the current descent direction, using this scalar quantity as a stepsize scaling factor. This drastically reduces the memory requirement while improving robustness to estimation noise.
2. GaNsZO leverages the composite structure of LLMs: by approximating the output-layer Jacobian using ZO perturbations and combining it with low-cost backpropagation of the final layer, GaNsZO computes the Gauss-Newton quadratic form along the descent direction. This produces an effective curvature estimate while saving one forward pass per iteration compared to central-difference methods, offering a significant reduction in wall-clock time.
3. On multiple fine-tuning benchmarks, GaNsZO empirically reduces memory consumption by up to 43% compared to Hessian-informed ZO baselines such as HiZOO, while consistently improving accuracy and stability. In terms of wall-clock time, GaNsZO delivers up to a $6\times$ speedup over HiZOO.

We also demonstrate that GaNsZO integrates effectively with modern ZO algorithms, yielding additional gains and highlighting its practicality and broad applicability for LLM fine-tuning.

2 Related Works

Fine-tuning LLMs. Full finetune LLMs with standard first-order optimizer is effective but memory-inefficient. This motivates parameter- and memory-efficient fine-tuning methods. Parameter-efficient fine-tuning (PEFT) reduces trainable parameters by updating only a small subset of weights or introducing lightweight modules, including adapters Houlsby et al. (2019), low-rank updates such as LoRA Hu et al. (2022), and prompt or prefix tuning Lester et al. (2021); Li & Liang (2021). Low-rank optimization methods like GaLore reduce optimizer memory by projecting gradients into a compact subspace while preserving full-parameter updates Zhao et al. (2024); He et al. (2025). Despite their effectiveness, these approaches still rely on backpropagation and maintain optimizer states.

Zeroth-Order Optimization. Zeroth-order optimization methods, which utilize objective function value oracle to estimate the gradient, have been applied in various fields in machine learning, including adversarial attack and defense Chen et al. (2017); Ilyas et al. (2018), automated ML Wang et al. (2022) and reinforcement learning Vemula et al. (2019). Recently, MeZO Malladi et al. (2023) demonstrated the practicality of ZO fine-tuning at scale, and subsequent work improved convergence or efficiency via momentum, variance reduction, and structured perturbations Jiang et al. (2024); Gautam et al. (2024); Yue et al. (2025). Among these, a promising direction is to incorporate second-order information into ZO optimization. For example, HiZOO Zhao et al. (2025b) accelerates convergence by estimating diagonal Hessian via central-difference. However, such methods require storing per-parameter Hessian statistics and often suffer from instability due to noisy Hessian estimates. Variants such as block-coordinate HiZOO Yu et al. (2025) and HELENE Zhao et al. (2025a) partially mitigate memory or stability issues. In this work, we close this gap by proposing a Gauss-Newton stepsized ZO



(a) Top-500 sorted absolute diagonal Hessian values across training steps.

(b) HiZOO's Hessian values estimation with $\alpha = 1e-8$.

(c) HiZOO's Hessian values estimation with $\alpha = 1e-6$.

Figure 1: Top-500 sorted absolute diagonal Hessian values and HiZOO estimator statistics for the 5th-layer self-attention W_Q in OPT-1.3B on the CB task. (a) Ground-truth top-500 $|\mathrm{diag}(\nabla^2 L)|$. (b)–(c) HiZOO estimated min/mean/max/99.9th percentile for $\alpha = 10^{-8}$ and $\alpha = 10^{-6}$ respectively.

optimizer that preserves low memory and computation costs while improving estimation robustness and training stability.

Second-Order Ideas for Training Neural Networks. Second-order methods can accelerate convergence but exact Newton-type or natural-gradient approaches are typically infeasible for large models due to high computational and memory costs. As a result, practical approximations such as Kronecker-factored methods Martens & Grosse (2015); Zhang et al. (2023) and diagonal curvature estimators Yao et al. (2021); Jahani et al. (2022); Liu et al. (2024) have been proposed. Beyond diagonal preconditioning, several works have shown that direction-wise, locally optimal stepsizes can effectively incorporate second-order information and accelerate optimization Schaul et al. (2013); Yehudai et al. (2023). Our work extends this stepsize-centric perspective to the zeroth-order setting.

3 Motivation and Analysis

3.1 Preliminaries

We denote the network parameters as $\boldsymbol{\theta} \in \mathbb{R}^d$, the empirical training loss as $L(\boldsymbol{\theta}) = g \circ h(\boldsymbol{\theta})$, where $h : \mathbb{R}^d \to \mathbb{R}^m$ denotes the network function without the output linear layer and $g : \mathbb{R}^m \to \mathbb{R}$ the map from the linear layer input to the scalar loss. ZO methods only query $L(\cdot)$ at chosen parameter values and form gradient estimates. Let $\mathbf{z} \in \mathbb{R}^d$ denote a random perturbation (e.g., $\sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) vector and $\mu > 0$ a smoothing scale. Two common gradient estimators are the forward and central estimators:

$$\hat{\nabla} L_{\mathrm{fwd}}(\boldsymbol{\theta}; \mathbf{z}) = \frac{L(\boldsymbol{\theta} + \mu\mathbf{z}) - L(\boldsymbol{\theta})}{\mu}\mathbf{z}, \qquad \hat{\nabla} L_{\mathrm{ctr}}(\boldsymbol{\theta}; \mathbf{z}) = \frac{L(\boldsymbol{\theta} + \mu\mathbf{z}) - L(\boldsymbol{\theta} - \mu\mathbf{z})}{2\mu}\mathbf{z}.$$

With learning rate η, MeZO utilizes a random number generator and updates the parameters as $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\hat{\nabla} L(\boldsymbol{\theta}_t)$.

To accelerate MeZO, many works propose to incorporate second-order information, mainly through approximating diagonal Hessian and updating with such precondition. For instance, HiZOO builds a diagonal preconditioner $\boldsymbol{\Sigma}$ from ZO queries so that $\boldsymbol{\Sigma}^{-1} \approx \nabla^2 L(\boldsymbol{\theta})$. $\boldsymbol{\Sigma}$ is estimated using three-point central differences from three function evaluations $L(\boldsymbol{\theta} + \mu\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{z}), L(\boldsymbol{\theta}), L(\boldsymbol{\theta} - \mu\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{z})$. Concretely, the diagonal Hessian estimation at $\boldsymbol{\theta}$ is calculated as

$$\hat{\boldsymbol{\Sigma}}_t(\boldsymbol{\theta}) = \frac{\Delta L}{2\mu^2}\left(\boldsymbol{\Sigma}_t^{-1/2}\mathbf{z}\mathbf{z}^T\boldsymbol{\Sigma}_t^{-1/2} - \boldsymbol{\Sigma}_t^{-1}\right), \quad \Delta L = L(\boldsymbol{\theta} + \mu\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{z}) + L(\boldsymbol{\theta} - \mu\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{z}) - 2L(\boldsymbol{\theta}).$$

Then $\boldsymbol{\Sigma}$ is updated through exponential moving average (EMA) to mitigate the estimation noise:

$$\boldsymbol{\Sigma}_{t+1}^{-1} = (1 - \alpha_t)\boldsymbol{\Sigma}_t^{-1} + \alpha_t \left|\hat{\boldsymbol{\Sigma}}_t\right|,$$

where α_t is a smoothing scale. In practice, this scheme requires three forward passes per iteration and the storage of $\boldsymbol{\Sigma}_t$.

3.2 REVISITING HIZOO ESTIMATORS AND TRUE HESSIAN

While HiZOO empirically achieves faster convergence than MeZO by harnessing second-order information, perhaps surprisingly, we observe that nearly all entries of HiZOO estimators concentrate near 1 (see Figure 1b) during training under the default EMA coefficient $\alpha_t = 10^{-8}$ in Zhao et al. (2025b). This implies that the diagonal preconditioner in HiZOO may provide only limited additional information relative to a scalar scaling factor. When enlarging α_t, we encounter certain training instabilities. The updates become extremely noisy in magnitude, and as shown in Figure 8 in Zhao et al. (2025b), the loss increases when α_t is larger than 10^{-6}.

Moreover, we examine the ground-truth Hessian in Figure 1a: most diagonal entries are near zero while a small fraction dominate in magnitude[1]. In fact, since most entries of the true diagonal Hessian are near zero, a small perturbation on the estimated entry σ_i can result in a substantial change in σ_i^{-1}. This inherent structure poses an essential stability issue in approximating diagonal Hessian not only for HiZOO but also for other precondition methods. For example, beyond ZO optimizer, Sophia Liu et al. (2024) also uses diagonal Hessian estimation. For stability, it utilizes clipping to upper bound the update magnitudes, where the reported clip rates can exceed 60%. For HiZOO, as shown in Figure 1c, when $\alpha_t = 10^{-6}$, due to the unstable diagonal Hessian estimate, optimization becomes numerically unstable and encounters loss explosion in our setting. To ensure stability, HiZOO has to take very small EMA coefficient and conservative initialization (e.g., initialized as 1), which results in the coarse approximation and yields only partial acceleration.

4 OUR METHOD

4.1 MOTIVATION

While diagonal preconditioning struggles to capture accurate Hessian information, another important factor in optimization - the *stepsize* - has received relatively less attention in the zeroth-order setting. In the first-order literature it has been shown that a single stepsize can effectively incorporate second-order information and accelerate convergence Schaul et al. (2013); Yehudai et al. (2023). The main idea is to choose a stepsize that is (locally) optimal along the current descent direction. Specifically, for a fixed descent direction $\mathbf{z} \in \mathbb{R}^d$, consider the one-dimensional subproblem

$$\min_{\eta} \; L(\boldsymbol{\theta} - \eta \mathbf{z}). \tag{1}$$

Approximating L by its second-order Taylor expansion around $\boldsymbol{\theta}$ gives

$$L(\boldsymbol{\theta} - \eta \mathbf{z}) \approx L(\boldsymbol{\theta}) - \eta \langle \mathbf{z}, \nabla L(\boldsymbol{\theta}) \rangle + \tfrac{1}{2} \eta^2 \, \mathbf{z}^\top \nabla^2 L(\boldsymbol{\theta}) \mathbf{z}. \tag{2}$$

Minimizing the quadratic right-hand side yields the locally optimal stepsize

$$\eta^\star \;=\; \frac{\langle \mathbf{z}, \nabla L(\boldsymbol{\theta}) \rangle}{\mathbf{z}^\top \nabla^2 L(\boldsymbol{\theta}) \mathbf{z}}. \tag{3}$$

Motivated by this stepsize-centric view, we aim to obtain a robust, low-memory estimate of the directional second derivative

$$\tilde{\sigma}_t \;:=\; \frac{\partial^2}{\partial \mu^2} L(\boldsymbol{\theta}_t + \mu \mathbf{z}_t)\Big|_{\mu=0} \;=\; \mathbf{z}_t^\top \nabla^2 L(\boldsymbol{\theta}_t)\, \mathbf{z}_t, \tag{4}$$

and use it as a curvature-aware stepsize scaling factor. Note that $\mathbb{E}_{\mathbf{z}}[\tilde{\sigma}_t] = \mathrm{tr}(\nabla^2 L(\boldsymbol{\theta}_t))$ when $\mathbb{E}[\mathbf{z}\mathbf{z}^\top] = \mathbf{I}_d$. Consequently, $\tilde{\sigma}_t$ can be regarded as an aggregate measure of the diagonal Hessian, which is less sensitive to noise in individual coordinates compared to the diagonal Hessian estimator.

4.2 GAUSS-NEWTON STEPSIZED ZO

In practice, inspired by PseuZO Yue et al. (2025), we further exploit the composite structure of $L = g \circ h$ to approximate $\tilde{\sigma}_t$ in Eq. (4). Writing $\mathbf{h} = h(\boldsymbol{\theta}_t)$ and letting $\mathbf{J}_h(\boldsymbol{\theta}_t)$ denote the Jacobian of h at $\boldsymbol{\theta}_t$, a straightforward application of the chain rule yields

$$\tilde{\sigma}_t = \big(\mathbf{J}_h(\boldsymbol{\theta}_t)\mathbf{z}_t\big)^\top \nabla^2 g\big(\mathbf{h}\big) \, \big(\mathbf{J}_h(\boldsymbol{\theta}_t)\mathbf{z}_t\big) + \sum_{i=1}^{m} \frac{\partial g}{\partial h_i}\big(\mathbf{h}\big) \, \mathbf{z}_t^\top \nabla^2 h_i(\boldsymbol{\theta}_t)\, \mathbf{z}_t, \tag{5}$$

[1]This structure is also observed for the eigenvalues of the Hessian Sagun et al. (2016); Wu et al. (2017).

Algorithm 1 GaNsZO

Require: Parameters $\boldsymbol{\theta}$, loss L, network h, outer loss g, steps T, μ, η, α, $\sigma_0 = 1$, distribution $\mathcal{D}$

for $t = 0, \ldots, T-1$ **do**
  Sample batch $\mathcal{B}$ and seed s
  (A) Two forward evaluations
  Compute $\mathbf{h} \leftarrow h(\boldsymbol{\theta}; \mathcal{B})$, $l \leftarrow L(\boldsymbol{\theta}; \mathcal{B})$; perturb $\boldsymbol{\theta} \leftarrow \text{PerturbParameters}(\boldsymbol{\theta}, \mu, s)$
  Compute $\mathbf{h}_+ \leftarrow h(\boldsymbol{\theta}; \mathcal{B})$, $l_+ \leftarrow L(\boldsymbol{\theta}; \mathcal{B})$; restore $\boldsymbol{\theta} \leftarrow \text{PerturbParameters}(\boldsymbol{\theta}, -\mu, s)$
  Form $\Delta\mathbf{o} \leftarrow (\mathbf{h}_+ - \mathbf{h})/\mu$
  (B) Gauss-Newton directional curvature
  $\boldsymbol{g}_h \leftarrow \nabla_h g(h)$; $\Delta\mathbf{o}_{\text{det}} \leftarrow \text{detach}(\Delta\mathbf{o})$
  $\phi \leftarrow \boldsymbol{g}_h \cdot \Delta\mathbf{o}_{\text{det}}$; $\mathbf{u} \leftarrow \nabla_h \phi$
  $\hat{\sigma} \leftarrow \Delta\mathbf{o}^\top \mathbf{u}$
  (C) Smoothing & ZO gradient & update
  $\sigma \leftarrow (1-\alpha)\sigma + \alpha\,\hat{\sigma}$
  Reset random number generator with seed s; sample $\mathbf{z} \sim \mathcal{D}$; $\hat{\boldsymbol{g}} \leftarrow \dfrac{l_+ - l}{\mu}\mathbf{z}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\,\sigma^{-1}\hat{\boldsymbol{g}}$
  Function PerturbParameters($\boldsymbol{\theta}$, μ, s):
    Reset random number generator with seed s
    Sample $\mathbf{z} \sim \mathcal{D}$ (e.g., Gaussian or Rademacher)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \mu\mathbf{z}$
    return $\boldsymbol{\theta}$
end for

where $\nabla^2 h_i(\boldsymbol{\theta}_t)$ is the Hessian of the i-th output coordinate of h. The first term is the Gauss-Newton quadratic form along $\mathbf{z}_t$ and often dominates Hessian Sankar et al. (2021). So we only estimate the first term and neglect the second term. For more empirical evidence, please refer to Appendix B.5.

To compute the Gauss-Newton term efficiently, we first form the finite-difference output increment

$$\Delta\mathbf{o}_t := \frac{h(\boldsymbol{\theta}_t + \mu\mathbf{z}_t) - h(\boldsymbol{\theta}_t)}{\mu} \approx \mathbf{J}_h(\boldsymbol{\theta}_t)\mathbf{z}_t.$$

Then the Gauss-Newton scalar can be approximated by $\Delta\mathbf{o}_t^\top \nabla^2 g(\mathbf{h}) \Delta\mathbf{o}_t$. Crucially, $\nabla^2 g(\mathbf{h})\,\Delta\mathbf{o}_t$ can be obtained by automatic differentiation restricted to the outer loss g:

$$\nabla^2 g(\mathbf{h})\,\Delta\mathbf{o}_t = \frac{\partial}{\partial h}\left(\frac{\partial}{\partial h} g(\mathbf{h}) \cdot \text{detach}(\Delta\mathbf{o}_t)\right).$$

Thus the Gauss-Newton term can be computed by (i) evaluating $h(\boldsymbol{\theta}_t + \mu\mathbf{z}_t)$ and $h(\boldsymbol{\theta}_t)$ to obtain $\Delta\mathbf{o}_t$, (ii) computing $\nabla g(\mathbf{h})$ and the scalar $\phi = \nabla_h g(\mathbf{h})^\top \Delta\mathbf{o}_t$, and (iii) backpropagating ϕ through g while treating $\Delta\mathbf{o}_t$ as a constant vector to obtain $\nabla_h \phi = \nabla^2 g(\mathbf{h})\,\Delta\mathbf{o}_t$. The final quadratic form is then $\hat{\sigma}_t = \Delta\mathbf{o}_t^\top \nabla^2 g(\mathbf{h})\,\Delta\mathbf{o}_t$. This procedure requires only two forward passes of the network plus two cheap auto-diff passes through the outer g, and negligible extra memory of the storage of the activation at the output of h. The scalar scaling factor can also be updated through EMA to enhance stability, i.e.,

$$\sigma_{t+1} = (1-\alpha_t)\sigma_t + \alpha_t \hat{\sigma}_t. \tag{6}$$

The overall algorithm is summarized in Algorithm 1. More discussions are provided in Appendix D.

4.3 Theoretical analysis

This section establishes the convergence of the proposed GaNsZO under mild assumptions. For simplicity we study the limiting case $\mu \to 0$ of Algorithm 1, and clip σ_{t-1} to $[c_0, c_1]$ prior to the EMA step in theoretical analysis, based on the experimental observation that σ_t remains stable. We assume that the eigenvalues of the Hessian matrix decay rapidly and employ the effective dimension $d_{\text{eff}} \approx \sup_{\boldsymbol{\theta}} \frac{\text{tr}(\nabla^2 L(\boldsymbol{\theta}))}{L_1} \ll d$ to characterize the convergence rate, where $L_1 = \|\nabla^2 L(\boldsymbol{\theta})\|_{\text{op}}$ denotes the smoothness constant. Furthermore, the stochastic noise in gradients and Hessians induced by the batch $\mathcal{B}$ is assumed to be bounded. The complete set of assumptions is detailed in Appendix C.

Theorem 4.1 provides the convergence rate of the proposed algorithm ignoring constants. The formal statement and proof is presented in Appendix C.

Table 1: Evaluation on OPT-1.3B and Llama3.1-8B using 1,024 training and 512 validation samples per task (for WSC, CB and COPA we reserve 100 validation samples and use the remainder for training). With the exception of the DROP task, where f1 is used, all other figures refer to accuracy (%). Each model block lists zeroth-order (ZO) methods first, followed by the first-order (FO) baseline separated by a rule. **Bold** entries indicate the best score among ZO methods; FO rows are provided for reference. Detailed results with std are in Appendix B.1.

Model	Method	SST-2	RTE	CB	BoolQ	WSC	WiC	MultiRC	COPA	ReCoRD	DROP	AVG
Task type		classification							multiple choice		generation	
OPT-1.3B	MeZO	88.8	62.2	68.4	63.5	58.6	57.9	56.2	74.3	71.2	22.1	62.3
	HiZOO	89.6	62.0	69.0	63.1	**60.2**	59.2	58.1	73.7	71.6	22.5	62.9
	GaNsZO$_{\alpha=1}$	90.8	62.4	70.2	**64.7**	59.0	59.7	59.1	75.3	**72.2**	24.8	63.8
	GaNsZO	**91.2**	**63.3**	70.2	64.0	58.4	**60.1**	**59.4**	**76.0**	72.0	24.8	**63.9**
	FO-Adam	93.0	68.3	70.8	69.4	63.2	63.0	61.3	76.7	71.7	26.6	66.4
Llama3.1-8B	MeZO	90.9	71.7	70.2	78.6	61.2	57.5	75.2	88.7	84.8	60.2	73.9
	HiZOO-L	90.9	71.8	70.8	77.9	60.5	57.4	75.1	**89.3**	84.8	58.5	73.7
	GaNsZO$_{\alpha=1}$	93.2	76.8	**72.0**	82.7	**61.9**	57.3	79.5	87.0	85.1	60.4	75.6
	GaNsZO	**93.7**	**77.0**	70.8	**83.4**	61.5	**58.7**	**80.3**	88.7	**85.2**	**60.8**	**76.0**
	FO-Adam	94.6	87.7	87.3	86.1	63.4	70.4	80.7	87.7	85.3	65.8	80.9

Theorem 4.1 (Informal)**.** *Under the assumptions specified in Appendix C, the proposed algorithm finds an ϵ-stationary point in $O\left(d_{\text{eff}} \cdot \max\left\{\frac{\Delta L_1}{\epsilon^2}, \frac{\delta L_1 \sigma^2}{\epsilon^4}\right\}\right)$ iterations, where d_{eff} is the effective dimension.*

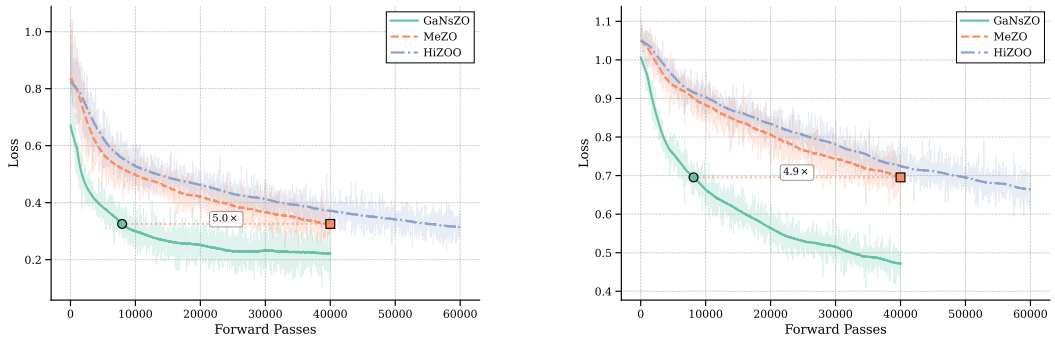
5 Experiments

Overview. We evaluate GaNsZO on autoregressive LLMs including OPT series and Llama3.1-8B across classification, multiple-choice, and generation benchmarks on GLUE and SuperGLUE data. We compare GaNsZO to MeZO, HiZOO and standard first-order fine-tuning baselines, and report accuracy, memory usage, and wall-clock training time. Empirically, GaNsZO achieves higher accuracy and faster convergence than HiZOO and MeZO, while reducing memory use by over 40% relative to HiZOO and incurring only marginal additional memory compared to MeZO. We also demonstrate GaNsZO's compatibility and superiority with PEFT methods (LoRA and prefix-tuning). Furthermore, we also evaluate the robustness of GaNsZO to different perturbation distributions (Gaussian, Rademacher) and structured low-rank directions, demonstrating the potential to combine it with other ZO distribution-based algorithms, such as DAP Ma & Huang (2025) and LOZO Chen et al. (2025). Finally we perform ablations on the EMA coefficient and alternative g and h splits to validate the stability and effectiveness of GaNsZO.

Setup. We implement MeZO, HiZOO, LOZO and our GaNsZO algorithm on top of the MeZO codebase to ensure a consistent training pipeline (identical data loading, prompt templates, batching, and evaluation). We finetune autoregressive LLMs (OPT series and Llama3.1-8B) on task suites derived from GLUE and SuperGLUE, using the same prompt templates as in the MeZO experiments. Unless stated otherwise, each experiment is trained for 20K steps with evaluations every 2K steps. For the primary experiments we use a batch size of 16 and, for controlled comparisons, operate on fixed random subsets of the data (1,024 examples for training and 512 for validation), with the same subsets reused across methods. Hyperparameters (learning rate η and ZO step size μ) are selected via a small grid search on the validation split. Additional details are provided in Appendix A.

5.1 Auto-Regressive Model Performance

We perform experiments on OPT-1.3B and Llama3.1-8B models on classification, multiple choice, and generation tasks. We also conduct experiments on Roberta-large in Appendix B. Specifically, we implement two versions of GaNsZO: the default GaNsZO with EMA smoothing and GaNsZO$_{\alpha=1}$ (no EMA) and compare them to MeZO, HiZOO (HiZOO-L for Llama3.1-8B to avoid OOM), and first-order fine-tuning baselines. Unless otherwise specified, we use Gaussian perturbations as the default noise distribution. Table 1 reports task-level and average accuracies. Detailed results with standard deviation over multiple seeds are provided in Appendix B.1 due to page limit. GaNsZO



(a) Training loss versus forward passes on SST-2.

(b) Training loss versus forward passes on CB.

Figure 2: Training loss versus forward passes on OPT-1.3B for (a) SST-2 and (b) CB, comparing MeZO, HiZOO and GaNsZO. GaNsZO reaches the same losses with fewer forward passes than the baselines.

attains the highest average accuracy and outperforms MeZO and HiZOO on the majority of tasks. Notably, even without EMA operation, the EMA-free variant GaNsZO$_{\alpha=1}$ still outperforms the baselines in many settings, indicating that our directional curvature estimator is inherently robust. Adding EMA further reduces run-to-run variance and offers additional improvement. Furthermore, GaNsZO exhibits significantly faster convergence when measured in terms of forward-pass efficiency. As shown in Figure 2, on SST-2 and CB, GaNsZO reaches the same training loss using up to $5.0\times$ fewer forward passes than MeZO and HiZOO. We additionally measure the *time-to-target*: the total wall-clock training time each method requires to first reach a common target loss (MeZO's final smoothed training loss), obtained by combining the per-iteration runtime with the iteration at which the target is first attained. This provides a direct, end-to-end view of optimization efficiency. As detailed in Appendix B.4, GaNsZO reaches the target substantially faster than the baselines, yielding a geometric-mean speedup of roughly $6\times$ over HiZOO on OPT-1.3B and $4.5\times$ over HiZOO-L on Llama-3.1-8B.

5.2 Memory Overhead Comparison

Next, we compare the peak GPU memory usage of GaNsZO, HiZOO, HiZOO-L, MeZO and FO-Adam on MultiRC task on OPT series and Llama3.1-8B models. Results are shown in Figure 3. As expected, full backprop (FO-Adam) incurs substantially higher memory because it has to retain activations and the full computation graph. Among ZO algorithms, GaNsZO uses considerably less memory than HiZOO (and even less than HiZOO-L in our measurements) because it does not maintain an $O(d)$ diagonal curvature state. Compared to MeZO, GaNsZO incurs only marginal additional memory, primarily for storing output-layer activations needed to compute the Gauss-Newton quadratic form; this overhead is small across model scales. These results confirm GaNsZO's memory-efficiency advantage while delivering the performance improvements reported in Section 5.1. For a more detailed breakdown of the autodiff memory (and computation) overhead, please see Appendix B.3.

5.3 Wall Clock Time Comparison

We measure the wall-clock time on identical hardware and implementation settings and report per-iteration time in Table 2. GaNsZO's per-iteration time is nearly identical to that of MeZO since GaNsZO requires only two forward evaluations for gradient estimation and a lightweight backprop through the top layer (the scalar stepsize adds negligible overhead). By contrast, HiZOO and HiZOO-L incur noticeably higher per-iteration costs because they require three forward passes per iteration to form diagonal curvature estimates. HiZOO-L further increases cost through extra low-rank estimator processing. Combining these with the convergence curves (Figure 2), it suggests that GaNsZO yields a substantial reduction in total wall-clock training time to reach a given training loss.

Model	Algorithm	SST-2	WIC	COPA
OPT-1.3B	MeZO	0.15s	0.17s	0.12s
	HiZOO	0.30s	0.34s	0.25s
	HiZOO-L	0.33s	0.37s	0.28s
	GaNsZO	0.16s	0.18s	0.13s
Llama3.1-8B	MeZO	0.73s	0.85s	0.56s
	HiZOO	1.52s	1.70s	1.23s
	HiZOO-L	1.62s	1.80s	1.33s
	GaNsZO	0.73s	0.85s	0.56s

Table 2: Per-iteration wall-clock time (seconds) for different models and training methods on a single NVIDIA 5880 (48 GiB). All methods use batch size 16. The reported time is averaged over 100 steps.

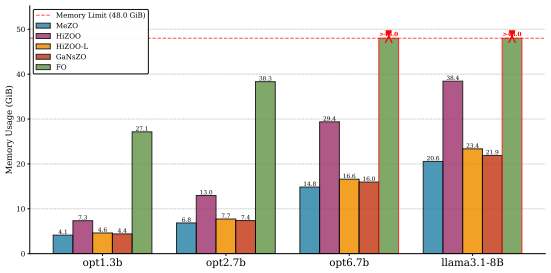


Figure 3: Peak GPU memory usage (GiB) on the MultiRC task (average input length $\approx$ 400 tokens) for a range of model scales. We compare zeroth-order methods (MeZO, HiZOO, GaNsZO) and first-order (FO) Adam; all runs use batch size 1. Bars indicate measured peak memory per method, where FO-Adam for OPT-6.7B and Llama3.1-8B exceed the 48 GiB device limit.

Table 3: Experiments on OPT-1.3B comparing PEFT methods (prefix, LoRA), alternative perturbation distributions (Rademacher), and structural update assumption (LOZO). The **bold** entry denotes the best score within that sub-block. "LOZO w/ GaNsZO" indicates applying GaNsZO's step-sizing inside the LOZO low-rank perturbation framework.

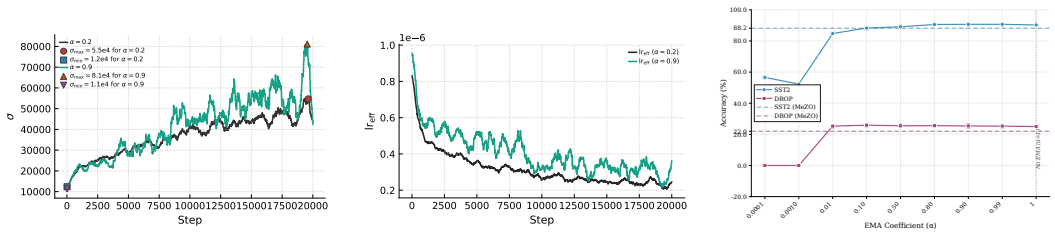
Task	**SST-2**	**RTE**	**CB**	**BoolQ**	**WSC**	**WIC**	**MultiRC**	**COPA**	**ReCoRD**	**DROP**	**AVG**
Task type	classification							multiple choice		generation	
MeZO (prefix)	88.7	64.6	69.6	**64.4**	57.7	**57.9**	56.5	72.0	70.3	22.9	62.5
HiZOO (prefix)	88.7	65.0	69.6	63.6	54.8	57.5	54.5	72.0	70.2	21.7	61.8
GaNsZO (prefix)	**90.8**	**65.0**	**71.5**	63.8	**59.6**	57.7	**56.5**	**72.0**	**70.8**	**23.6**	**63.1**
MeZO (LoRA)	88.5	59.9	69.6	63.0	54.8	58.2	58.2	76.0	70.6	23.2	62.2
HiZOO (LoRA)	88.5	**64.6**	69.6	**63.1**	54.8	**60.6**	58.2	76.0	**72.5**	23.1	63.1
GaNsZO (LoRA)	**90.4**	63.7	**73.4**	62.8	**57.7**	59.2	**58.2**	**77.0**	71.7	**24.1**	**63.8**
MeZO (Rademacher)	86.6	61.3	67.8	62.6	58.7	53.6	58.4	75.0	70.6	22.5	61.7
HiZOO (Rademacher)	87.1	61.5	67.8	61.4	54.8	**57.4**	59.4	74.0	70.8	23.1	61.7
GaNsZO (Rademacher)	**90.1**	**61.8**	**67.8**	**63.2**	**58.7**	56.6	**59.4**	**75.0**	**72.3**	**25.4**	**63.0**
LOZO	90.4	**61.3**	67.8	60.8	56.7	55.1	58.6	73.0	**71.2**	21.9	61.7
LOZO w/ GaNsZO	**91.2**	61.0	**67.8**	**64.9**	**58.7**	**57.2**	**59.2**	**73.0**	70.6	**24.6**	**62.8**

5.4 Compatible with PEFT Methods

Parameter-efficient fine-tuning (PEFT) methods reduce the number of trainable parameters and pair naturally with zeroth-order optimization. To evaluate compatibility, we apply GaNsZO, MeZO and HiZOO to PEFT settings (LoRA and prefix-tuning) on OPT-1.3B and report task performance in Table 3. The results show that GaNsZO integrates seamlessly with both PEFT mechanisms and yields consistently higher average task accuracy than the ZO baselines under the same PEFT configurations. In particular, GaNsZO with LoRA achieves the best average performance in our runs, while GaNsZO with prefix-tuning also improves over MeZO and HiZOO on most tasks.

5.5 Robustness across Perturbation Distributions and Low-Rank Directions

We further exmaine GaNsZO's potential to combine with other ZO algorithms as a learning rate scheduler. Specifically, we evaluate GaNsZO under different perturbation distributions (Rademacher; the default is Gaussian) and under structured, low-rank perturbations inspired by LOZO. Rademacher perturbation, which takes values ± 1 with equal probability, offers a computationally efficient alternative to Gaussian noise, particularly in hardware-accelerated settings. Under Rademacher perturbation, GaNsZO maintains or improves its gains over MeZO and HiZOO (see the Rademacher block in Table 3) on OPT-1.3B model, showing its robustness and potential to be utilized in other settings. More details are provided in Appendix A.



(a) Dynamics of σ_t. (b) Effective LR of GaNsZO. (c) Performance vs. α

Figure 4: (Left) Evolution of σ_t in GaNsZO under different EMA coefficient α. (Middle) Evolution of the effective LR of GaNsZO for the same settings. (Right) Task performance on SST-2 (Llama3.1-8B) and DROP (OPT-1.3B) across $\alpha \in [10^{-4}, 1]$.

Additional, we evaluate GaNsZO under the setting of LOZO to examine the effectiveness under structural perturbation assumption. LOZO samples low-rank perturbations by decomposing matrix-shaped perturbations as $\mathbf{z} = \mathbf{U}\mathbf{V}^\top$ with $\mathbf{U} \in \mathbb{R}^{m\times r}, \mathbf{V} \in \mathbb{R}^{n\times r}$ (with $r \ll m, n$), which better aligns with low-rank gradient structure. Because GaNsZO treats the curvature estimate as a stepsize scaling factor, it can be applied directly on top of LOZO. Table 3 reports that combining GaNsZO with LOZO improves average accuracy and convergence speed relative to LOZO alone. These results demonstrate GaNsZO's flexibility as a lightweight, distribution-agnostic curvature-aware scheduler.

5.6 Analysis and Ablations

Behavior of the Curvature Scalar. We examine the evolution of the curvature scalar σ_t and the corresponding effective learning rate η/σ_t during training. Figure 4a and 4b shows these quantities during training of OPT-1.3B on SST-2. We observe that σ_t exhibits some fluctuations over iterations, as it inherently depends on the stochastic sampling. However, it is noted that σ_t exhibits substantially lower variance than HiZOO: over 10k training iterations, the ratio $\max_t \sigma_t / \min_t \sigma_t \approx 8$ for OPT-1.3B under EMA 0.9, while the HiZOO estimator shows much larger variation across coordinates under EMA 10^{-6}, with $\max_i h_{t,i} / \min_i h_{t,i} \approx 60$ (Figure 1c). The corresponding effective learning rate also exhibits a relatively smooth decay trend. More experiments are provided in Appendix B.6.

Ablations on EMA coefficients. As shown in Table 1, GaNsZO$_{\alpha=1}$ achieves better average accuracy than MeZO and HiZOO. To further assess robustness, we sweep the EMA coefficient α across a broad range and report performance on representative tasks (SST-2 on Llama3.1-8B and DROP on OPT-1.3B). Figure 4c plots per-task performance as a function of α. Note that $\alpha = 1$ corresponds to the EMA-free variant, while $\alpha = 0$ freezes the scalar (no update). As long as α lies within a wide practical range (e.g., $[0.1, 1]$), GaNsZO remains stable and consistently outperforms MeZO. This range is substantially less restrictive than the small decay values required by HiZOO. This corroborates that our directional curvature estimator is both low-variance and robust in practice, simplifying hyperparameter tuning for real tasks.

6 Conclusions

We present GaNsZO, a memory- and computation-efficient ZO optimizer that uses a scalar formed from the directional Gauss–Newton quadratic form to scale stepsize. GaNsZO eliminates the need to store $O(d)$ diagonal curvature state and reduces per-iteration forward evaluations, yielding faster convergence and substantially lower memory and wall-clock cost than existing Hessian-informed ZO baselines. Future directions include a more accurate Hessian approximation, combining GaNsZO with quantization or other system optimizations, and a deeper theoretical analysis of estimator variance and convergence.

AI use statement

In this work, we used generative AI tools to implement methods, to assist with translation, to support qualitative and thematic data analysis, and to interpret results. We have not used generative AI tools

to assist in the writing of proofs, to help develop theoretical models or conceptual frameworks, to formulate mathematical claims, to provide critical ingredients for proving those claims, to propose or refine hypotheses, to design or provide feedback on our research methodology or experiments, or to clean and reformat datasets; the generation of synthetic data sets is not applicable to this work, as all experiments use existing corpora rather than AI-generated data. Additionally, we used generative AI tools to edit the paper for readability. We have reviewed all AI-assisted work: all AI-assisted implementations were verified and tested for correctness by the authors against reference optimizer implementations; and every AI-assisted translation, data-analysis summary, and result interpretation was independently confirmed against the underlying experimental logs and figures. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

Our anonymized source code, including the optimizer implementations and the training and evaluation scripts, is available at`https://anonymous.4open.science/r/GaNsZO-3463/`.

References

Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.

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

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 *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, 2005.

Sizhe Dang, Yangyang Guo, Yanjun Zhao, Haishan Ye, Xiaodong Zheng, Guang Dai, and Ivor Tsang. FZOO: Fast zeroth-order optimizer for fine-tuning large language models towards adam-scale speed. *arXiv preprint arXiv:2506.09034*, 2025.

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

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*, 2023.

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

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

Yutong He, Pengrui Li, Yipeng Hu, Chuyan Chen, and Kun Yuan. Subspace optimization for large language models with convergence guarantees. In *International Conference on Machine Learning*, 2025.

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

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 *International Conference on Learning Representations*, 2022.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, 2018.

Majid Jahani, Sergey Rusakov, Zheng Shi, Peter Richtárik, Michael W. Mahoney, and Martin Takác. Doubly adaptive scaled algorithm for machine learning using second-order information. In *International Conference on Learning Representations*, 2022.

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 *AAAI Conference on Artificial Intelligence*, 2024.

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 *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

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

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*, 2021.

Hector J Levesque, Ernest Davis, and Leora Morgenstern. The Winograd schema challenge. *KR*, 2012.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021.

Qiyao Liang, Jinyeop Song, Yizhou Liu, Jeff Gore, Ila Fiete, Risto Miikkulainen, and Xin Qiu. The blessing of dimensionality in llm fine-tuning: A variance-curvature perspective. *arXiv preprint arXiv:2602.00170*, 2026.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International Conference on Learning Representations*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Shaocong Ma and Heng Huang. Revisiting zeroth-order optimization: Minimum-variance two-point estimators and directionally aligned perturbations. In *International Conference on Learning Representations*, 2025.

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*, 2023.

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

Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 2017.

Mohammad Taher Pilehvar and José Camacho-Collados. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, 2011.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the Hessian in deep learning: Singularity and beyond. *ArXiv preprint arXiv:1611.07476*, 2016.

Adepu Ravi Sankar, Yash Khasbage, Rahul Vigneswaran, and Vineeth N Balasubramanian. A deeper look at the Hessian eigenspectrum of deep neural networks and its applications to regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In *International Conference on Machine Learning*, 2013.

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 *Conference on Empirical Methods in Natural Language Processing*, 2013.

Qian-Yuan Tang, Yufei Gu, Yunfeng Cai, Mingming Sun, Ping Li, Zeke Xie, et al. Investigating the overlooked hessian structure: From CNNs to LLMs. In *International Conference on Machine Learning*, 2025.

Anirudh Vemula, Wen Sun, and J Bagnell. Contrasting exploration in parameter and action space: A zeroth-order optimization perspective. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, 2018.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems*, 2019.

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

Lei Wu, Zhanxing Zhu, et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *ArXiv preprint arXiv:1706.10239*, 2017.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. AdaHessian: An adaptive second order optimizer for machine learning. In *AAAI conference on artificial intelligence*, 2021.

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

Gilad Yehudai, Alon Cohen, Amit Daniely, Yoel Drori, Tomer Koren, and Mariano Schain. Locally optimal descent for dynamic stepsize scheduling. *ArXiv preprint arXiv:2311.13877*, 2023.

Zhiyuan Yu, Yifei Cheng, Liang Ding, Xinmei Tian, Li Shen, and Dacheng Tao. Memory efficient block coordinate descent method for forward-only second-order finetuning of LLM models. OpenReview preprint, 2025.

Pengyun Yue, Xuanlin Yang, Mingqing Xiao, and Zhouchen Lin. PseuZO: Pseudo-zeroth-order algorithm for training deep neural networks. In *Conference on Neural Information Processing Systems*, 2025.

Lin Zhang, Shaohuai Shi, and Bo Li. Eva: Practical second-order optimization with Kronecker-vectorized approximation. In *International Conference on Learning Representations*, 2023.

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 arXiv:1810.12885*, 2018.

Huaqin Zhao, Jiaxi Li, Yi Pan, Shizhe Liang, Xiaofeng Yang, Fei Dou, Tianming Liu, and Jin Lu. Helene: Hessian layer-wise clipping and gradient annealing for accelerating fine-tuning LLM with zeroth-order optimization. In *Conference on Empirical Methods in Natural Language Processing*, 2025a.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *International Conference on Machine Learning*, 2024.

Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor W. Tsang. Second-order fine-tuning without pain for LLMs: A Hessian informed zeroth-order optimizer. In *International Conference on Learning Representations*, 2025b.

A Details of Experiments in Main Text

Datasets. We evaluate on classification, multiple-choice and generative tasks drawn from GLUE and SuperGLUE Wang et al. (2018; 2019). The individual tasks are SST-2 Socher et al. (2013), RTE Dagan et al. (2005), CB De Marneffe et al. (2019), BoolQ Clark et al. (2019), WSC Levesque et al. (2012), WiC Pilehvar & Camacho-Collados (2019), MultiRC Khashabi et al. (2018), COPA Roemmele et al. (2011), ReCoRD Zhang et al. (2018) and DROP Dua et al. (2019). For each task, we randomly select 1024 training samples and 512 validation samples. For small datasets with very limited examples (WSC, CB and COPA) we reserve 100 examples for validation and use the remainder for training. We also adopt the same prompt design approach as MeZO Malladi et al. (2023), following GPT-3 and PromptSource with minor modifications. Most experiments are conducted on a single NVIDIA 5880 48GiB GPU, while some experiments requiring full finetune are conducted on a single NVIDIA H800 80GiB GPU.

Hyperparameters. Unless otherwise noted, each zeroth-order (ZO) algorithm is trained for 20K optimization steps and FO-Adam is trained for 5 epochs. For tokenization, we set a uniform maximum length MAX_LEN $= 2048$. For ZO methods we use a batch size of 16, and for FO-Adam we use the largest batch size that fits the GPU under identical system settings. To ensure a fair comparison, hyperparameters such as perturbation length μ and learning rate η are selected by a small grid search for each Zeroth-order algorithm. Specifically, for HiZOO and HiZOO-L, we sweep Hessian smoothing coefficients and decay factors, and for LOZO we sweep the low-rank dimension r and perturbation interval. For GaNsZO we use an EMA coefficient of $\alpha = 0.9$ by default (ablation in Figure 4c shows that GaNsZO is insensitive to α over a wide range). Final hyperparameter settings and search ranges are listed in Table 4.

Rademacher Distribution and LOZO. A Rademacher perturbation $\mathbf{z} \in \mathbb{R}^d$ has independent entries taking values ± 1 with equal probability:

$$\Pr(z_i = 1) = \Pr(z_i = -1) = \tfrac{1}{2}, \qquad i = 1, \ldots, d.$$

Rademacher vectors satisfy $\mathbb{E}[\mathbf{z}] = 0$ and $\mathbb{E}[\mathbf{z}\mathbf{z}^\top] = \mathbf{I}_d$, so standard finite-difference ZO estimators (e.g., SPSA-style or two-point quotients) remain asymptotically unbiased in the small-μ limit. There are two complementary motivations for considering Rademacher directions. From a theoretical perspective, recent analyses suggest settings in which Rademacher perturbations can match or even improve statistical properties of Gaussian probes Ma & Huang (2025). From a systems perspective, Rademacher sampling is often easier and more hardware-friendly to generate on common ML stacks, enabling faster and more memory-efficient implementations in practice Dang et al. (2025).

In our experiments, we evaluate GaNsZO, MeZO and HiZOO using both Gaussian and Rademacher perturbations. Empirically, Rademacher perturbations perform slightly worse than Gaussian in our controlled fine-tuning setup (See Table 3); however, GaNsZO's relative advantage over MeZO and HiZOO becomes even more pronounced under Rademacher sampling. We do not yet fully characterize the cause of this effect, but the result underscores GaNsZO's practical compatibility with alternative perturbation schemes.

In LLM training, it has been observed that the gradients of the matrix-shaped parameters demonstrate certain low-rankness. To facilitate this inherent property, LOZO (low-rank ZO) Chen et al. (2025) replaces dense matrix-shaped perturbations by low-rank factors: for a weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$, LOZO samples $\mathbf{U} \in \mathbb{R}^{m\times r}$ and $\mathbf{V} \in \mathbb{R}^{n\times r}$ with $r \ll \min(m, n)$ and uses the rank-r perturbation $\mathbf{U}\mathbf{V}^\top$. The corresponding low-rank finite-difference gradient estimator for the collection of weight matrices $\mathbf{W} := \{\mathbf{W}_k\}_{k=1}^K$ may be written compactly as

$$\hat{\nabla}_{\mathbf{W}_k} L(\mathbf{W}) := \frac{L\left(\{\mathbf{W}_k + \mu\mathbf{U}_k\mathbf{V}_k\}_{k=1}^K\right) - L\left(\{\mathbf{W}_k - \mu\mathbf{U}_k\mathbf{V}_k\}_{k=1}^K\right)}{2\mu} \cdot \left(\mathbf{U}_k\mathbf{V}_k^\top / r_k\right). \quad (7)$$

In order to leverage momentum and reduce memory usage, in practice, LOZO updates $\mathbf{V}$ periodically and only samples $\mathbf{U}$ within each period.

Importantly, the choice of perturbation distribution or structural updates is orthogonal to our curvature-based scalar preconditioner. Empirically (Table 3), combining GaNsZO with Rademacher or LOZO perturbations preserves or improves accuracy and convergence speed relative to the corresponding baselines, demonstrating GaNsZO's versatility as a distribution-agnostic, lightweight curvature-aware optimizer.

Table 4: The hyperparameter grids for OPT-1.3B and Llama3.1-8B; the weight decay is set to zero.

Experiment	Hyperparameters	Values
MeZO	Batch size	16
	Learning rate	$\{1e-6, 1e-7\}$
	μ	$1e-3$
MeZO (prefix)	Batch size	16
	Learning rate	$\{1e-2, 1e-3\}$
	μ	$1e-1$
	#prefix tokens	5
MeZO (LoRA)	Batch size	16
	Learning rate	$\{1e-4, 5e-5\}$
	μ	$1e-2$
	(r, α)	(8,16)
HiZOO, HiZOO-L	Batch size	16
	Learning rate	$\{1e-6, 1e-7\}$
	μ	$1e-3$
	α_{HiZOO}	$\{1e-7, 1e-8\}$
HiZOO (prefix)	Batch size	16
	Learning rate	$\{1e-2, 1e-3\}$
	μ	$1e-1$
	#prefix tokens	5
	α_{HiZOO}	$\{1e-7, 1e-8\}$
HiZOO (LoRA)	Batch size	16
	Learning rate	$\{1e-4, 5e-5\}$
	μ	$1e-2$
	(r, α)	(8,16)
	α_{HiZOO}	$\{1e-7, 1e-8\}$
LOZO	Batch size	16
	Learning rate	$\{1e-6, 1e-7\}$
	μ	$1e-3$
	r	$\{1, 2\}$
	Step interval	$\{50, 100\}$
GaNsZO	Batch size	16
	Learning rate	$\{1e-2, 1e-3\}$
	μ	$1e-3$
	α_{EMA}	0.9
GaNsZO (prefix)	Batch size	16
	Learning rate	$\{1e-2, 1e-3\}$
	μ	$1e-1$
	#prefix tokens	5
	α_{EMA}	0.9
GaNsZO (LoRA)	Batch size	16
	Learning rate	$\{1e-2, 1e-3\}$
	μ	$1e-2$
	(r, α)	(8,16)
	α_{EMA}	0.9
FO-Adam (OPT-1.3B)	Batch size	4,8
	Learning rate	$\{1e-4, 5e-5\}$
FO-Adam+LoRA (Llama3.1-8B)	Batch size	1,2
	Learning rate	$\{1e-4, 5e-5\}$

Table 5: Complete results with standard deviations of Table 1. All models are run using three random seeds:{0, 42, 2026}. GaNsZO$_1$ is the abbreviation of GaNsZO$_{\alpha=1}$.

Model Task type	Method	SST-2	RTE	CB	BoolQ	WSC	WiC	MultiRC	COPA	ReCoRD	DROP	AVG
		classification							multiple choice		generation	
OPT-1.3B	MeZO	88.8 ± 0.6	62.2 ± 3.4	68.4 ± 1.0	63.5 ± 1.2	58.6 ± 1.0	57.9 ± 1.7	56.2 ± 1.4	74.3 ± 1.5	71.2 ± 1.0	22.1 ± 0.6	62.3 ± 0.5
	HiZOO	89.6 ± 0.8	62.0 ± 2.5	69.0 ± 1.0	63.1 ± 2.2	**60.2** ± 2.0	59.2 ± 3.2	58.1 ± 1.6	73.7 ± 0.6	71.6 ± 0.8	22.5 ± 0.7	62.9 ± 0.6
	GaNsZO$_1$	90.8 ± 0.6	62.4 ± 1.4	70.2 ± 1.0	**64.7** ± 0.6	59.0 ± 2.6	59.7 ± 2.0	59.1 ± 0.3	75.3 ± 1.5	**72.2** ± 0.5	**24.8** ± 0.5	63.8 ± 0.4
	GaNsZO	**91.2** ± 0.7	**63.3** ± 1.3	**70.2** ± 1.0	64.0 ± 1.6	58.4 ± 2.3	**60.1** ± 3.0	**59.4** ± 0.8	**76.0** ± 1.0	72.0 ± 0.8	24.8 ± 0.9	**63.9** ± 0.5
	FO-Adam	93.0 ± 1.0	68.3 ± 3.1	70.8 ± 2.7	69.4 ± 3.5	63.2 ± 1.4	63.0 ± 1.2	61.3 ± 2.5	76.7 ± 2.1	71.7 ± 0.8	26.6 ± 1.1	66.4 ± 0.7
Llama3.1-8B	MeZO	90.9 ± 1.4	71.7 ± 2.9	70.2 ± 1.0	78.6 ± 1.7	61.2 ± 2.0	57.5 ± 1.8	75.2 ± 2.7	88.7 ± 2.1	84.8 ± 0.9	60.2 ± 3.6	73.9 ± 0.7
	HiZOO-L	90.9 ± 1.2	71.8 ± 2.4	70.8 ± 1.0	77.9 ± 1.5	60.5 ± 2.6	57.4 ± 2.3	75.1 ± 2.9	**89.3** ± 2.3	84.8 ± 0.9	58.5 ± 2.3	73.7 ± 0.7
	GaNsZO$_1$	93.2 ± 0.6	76.8 ± 1.3	**72.0** ± 2.7	82.7 ± 0.9	**61.9** ± 1.4	57.3 ± 2.7	79.5 ± 2.8	87.0 ± 1.0	85.1 ± 1.0	60.4 ± 2.3	75.6 ± 0.6
	GaNsZO	**93.7** ± 0.8	**77.0** ± 1.7	70.8 ± 1.0	**83.4** ± 0.6	61.5 ± 1.9	**58.7** ± 1.7	**80.3** ± 2.3	88.7 ± 2.1	**85.2** ± 1.1	**60.8** ± 2.2	**76.0** ± 0.5
	FO-Adam	94.6 ± 1.0	87.7 ± 2.7	87.3 ± 2.1	86.1 ± 2.1	63.4 ± 1.0	70.4 ± 3.6	80.7 ± 1.8	87.7 ± 1.2	85.3 ± 1.2	65.8 ± 1.2	80.9 ± 0.6

Table 6: Experiments on Roberta-large (350M parameters), $k = 512$. The **bold** number represents the highest evaluation performance.

Task Task type	SST-2	SST-5	SNLI	MNLI	TREC	RTE	AVG
	sentiment		natural language inference			topic	
MeZO	93.5	**55.1**	85.4	79.5	91.3	77.9	80.5
MeZO (LoRA)	90.2	44.3	74.2	65.7	56.6	71.8	67.1
MeZO (prefix)	93.3	54.4	83.9	75.5	90.6	69.1	77.8
HiZOO	93.6	54.1	80.4	**79.6**	89.2	72.2	78.2
HiZOO (LoRA)	90.0	42.6	73.6	66.5	52.2	72.0	66.2
HiZOO (prefix)	92.8	55.4	84.2	75.6	90.6	69.1	78.0
GaNsZO	**96.9**	54.9	**85.8**	79.3	**93.2**	**79.1**	**81.5**
GaNsZO (LoRA)	90.1	45.1	75.3	67.1	58.4	73.3	68.2
GaNsZO (prefix)	93.4	54.3	84.4	77.2	91.2	74.1	79.1

B Additional Experiments

B.1 More Finetune Results

Complete Results of Table 1. We report full experimental results with standard deviations across three random seeds: {0, 42, 2026}. For each seeds, we run independent experiments under identical hyperparameter settings and compute the mean and standard deviation over seeds. The multi-seed evaluation validates that the improvements of GaNsZO over MeZO and HiZOO are consistent and statistically reliable, rather than arising from favorable initialization or stochastic effects.

Experiment on RoBERTa-Large. We have primarily focused on autoregressive LLMs such as OPT and Llama variants in the main text. To evaluate the architectural generality of GaNsZO beyond autoregressive LLMs, we apply the same ZO protocol to a masked language model, RoBERTa-large. We fine-tune MeZO, HiZOO and GaNsZO on several downstream classification tasks (GLUE-style) using the same hyperparameter tuning procedure as in the main experiments and the hyperparameter setting can be found in Table 7. Results in Table 6 show that GaNsZO consistently outperforms the ZO baselines across the evaluated tasks. The EMA-free variant (GaNsZO$_{\alpha=1}$) also improves upon MeZO and HiZOO on most tasks, while adding EMA reduces variability across runs. These findings suggest that the benefits of GaNsZO are not limited to autoregressive architectures but extend to masked-language models as well.

B.2 Different choices of g and h

Our GaNsZO estimator exploits the composite structure $L = g \circ h$ and by default it includes the linear head inside h and treat g as mapping from the last-hidden representation to the scalar loss. An alternative is to let h produce the model logits and take g as the outer loss (i.e., compute curvature / autodiff from the logits, we refer to this below as "GaNsZO (Logits)"). We ablate these two choices on OPT-1.3B in Table 8. Empirically, our default GaNsZO achieves better accuracy than the "logits" variant. We attribute this to the fact that GaNsZO performs the zeroth-order perturbation in a

Table 7: Hyperparameter grids for RoBERTa-large (350M). Weight decay is set to zero and batch size is 64.

Method	Hyperparameter	Values
MeZO, HiZOO	Learning rate	$\{1e{-}5, 1e{-}6, 1e{-}7\}$
	μ	$1e{-}3$
	α_{HiZOO}	$\{1e{-}7, 1e{-}8\}$
GaNsZO	Learning rate	$\{1e{-}2, 1e{-}3\}$
	μ	$1e{-}3$
	α_{EMA}	0.9
MeZO, HiZOO (prefix)	Learning rate	$\{1e{-}2, 5e{-}3, 1e{-}3\}$
	μ	$1e{-}1$
	#prefix tokens	5
GaNsZO (prefix)	Learning rate	$\{1e{-}2, 1e{-}3\}$
	μ	$1e{-}1$
	#prefix tokens	5
MeZO, HiZOO (LoRA)	Learning rate	$\{1e{-}4, 5e{-}5, 1e{-}5\}$
	μ	$1e{-}2$
	(r, α)	$(8, 16)$
GaNsZO (LoRA)	Learning rate	$\{1e{-}2, 1e{-}3\}$
	μ	$1e{-}2$
	(r, α)	$(8, 16)$

Table 8: Accuracy of GaNsZO with different autodiff targets (g and h choices) on OPT-1.3B.

Method	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	DROP	AVG
GaNsZO (Logits)	90.0	62.1	69.6	64.1	**59.6**	55.7	59.2	76.0	**72.7**	25.1	63.4
GaNsZO	**90.6**	**63.1**	**69.6**	**65.7**	58.7	**58.5**	**59.2**	**77.0**	72.2	**25.4**	**64.0**

lower-dimensional feature space while leveraging additional automatic differentiation, which leads to more precise and lower-variance curvature estimates.

We also compare the peak memory of both variants. Specifically, we report the peak GPU memory (GiB) consumption on OPT series on the MultiRC task (average input length ≈400 tokens, batch size 1) in Table 9. Perhaps counter-intuitively, the default GaNsZO uses less peak memory than the "logits" variant. A plausible explanation is that the dimension of logits (which equals the vocabulary size) is usually an order of magnitude larger than the hidden-state dimension. Forming finite-difference outputs and computing the Gauss-Newton quadratic form on such high-dimensional logits therefore incurs substantial memory overhead, which can exceed the memory required to store activations and computation graphs when backpropagating to the last hidden state.

B.3 Overhead of Outer-Loss Autodiff

In this section we provide a clearer breakdown of the additional memory and computation overhead introduced by the outer-loss autodiff step in GaNsZO. From a theoretical perspective, our default

Table 9: The GPU memory consumption (GiB) across different model scales on MultiRC (average input length ≈ 400 tokens) with batch size 1. We compare GaNsZO when differentiating from *Logits* (denoted as GaNsZO (Logits)) versus *last_hidden_state* (denoted as GaNsZO, default setting).

Model	**OPT-1.3B**	**OPT-2.7B**	**OPT-6.7B**	**Llama3.1-8B**
GaNsZO (Logits)	5.50 GiB	8.22 GiB	16.57 GiB	24.24 GiB
GaNsZO	4.40 GiB	7.38 GiB	15.96 GiB	21.88 GiB

Table 10: Memory breakdown of the outer-loss autodiff step. "Forward" denotes the forward-pass memory, "Autograd" denotes the memory with outer-loss autodiff enabled, and "Peak" is the peak memory usage.

Model	Parameter	Last hidden state	Logits	Forward	Autograd	Peak
OPT (SST-2)	2.45 GiB	8 MiB	200 MiB	2.13 GiB	2.59 GiB	5.04 GiB
Llama (SST-2)	14.96 GiB	18 MiB	558 MiB	5.69 GiB	6.40 GiB	21.36 GiB
OPT (WiC)	2.45 GiB	10 MiB	251 MiB	2.88 GiB	2.96 GiB	5.41 GiB
Llama (WiC)	14.96 GiB	20 MiB	620 MiB	5.69 GiB	7.86 GiB	22.82 GiB

Table 11: Runtime breakdown of the outer-loss autodiff step. We report the time cost of computing σ_t and the total per-step runtime.

Model	Two-forward time	Obtain σ_t time	Three-perturb time	One-step total time
OPT (SST-2)	0.13 s	0.02 s	0.01 s	0.16 s
Llama (SST-2)	0.62 s	0.11 s	0.01 s	0.74 s
OPT (WiC)	0.15 s	0.02 s	0.01 s	0.18 s
Llama (WiC)	0.71 s	0.11 s	0.02 s	0.85 s

design computes the curvature from the last hidden state. As a result, the additional memory consumed by GaNsZO over MeZO mainly comes from storing the activations at the input and output of the last hidden state, as well as the temporary workspace introduced during the autodiff process (e.g., in the cross-entropy loss computation). The additional tensor activations are approximately $O(bsv + bsd)$, where b, s, d, and v denote the batch size, sequence length, hidden dimension, and vocabulary size, respectively. In typical ZO settings, b and s are small, so this overhead remains much smaller than the total number of model parameters.

Empirically, a detailer breakdown of the memory overhead is shown in Table 10 and computation overhead is shown in Table 11 under the settings of Table 1. The additional cost of GaNsZO over MeZO is mainly reflected in the memory gap between the Forward and Autograd settings, as well as the extra latency required to compute σ_t. Overall, the results show that the additional memory introduced by outer-loss autodiff is moderate compared to the full model footprint, and the extra latency for computing σ_t is small relative to the total per-step runtime. This confirms that GaNsZO incurs only limited overhead over MeZO in practice.

B.4 Wall-Clock Time-to-Target Analysis

To make the efficiency claim directly verifiable, we report an end-to-end wall-clock *time-to-target* comparison on both OPT-1.3B and Llama-3.1-8B. For each method we smooth its training-loss trajectory with the same procedure used for our convergence curves, take the final (lowest) smoothed loss reached by MeZO as a common target, and report the wall-clock training time required for each method to first reach this target. Wall-clock time is computed from the measured per-iteration runtime together with the iteration at which the target is first achieved. All methods use the same batch size, perturbation scale ($\mu = 10^{-3}$), and random seed. On OPT-1.3B the baseline is HiZOO, whereas on Llama-3.1-8B the baseline is HiZOO-L, following the convention used throughout the paper. The results are reported in Table 12 (OPT-1.3B) and Table 13 (Llama-3.1-8B); all times are in seconds.

On OPT-1.3B, GaNsZO reaches the common target substantially faster than HiZOO on every task, with per-task speedups ranging from $2.5\times$ (DROP) to $16.6\times$ (WSC), corresponding to a geometric-mean end-to-end speedup of approximately $\mathbf{6.0}\times$ over HiZOO and approximately $\mathbf{5.0}\times$ over MeZO. On Llama-3.1-8B, across the tasks on which all three methods cleanly reach the common target, GaNsZO attains a geometric-mean speedup of approximately $\mathbf{4.5}\times$ over HiZOO-L, and approximately $\mathbf{4.4}\times$ over MeZO across all ten tasks. These results are consistent across both model scales and confirm that the efficiency benefit of directional curvature-aware step-size adaptation carries over to larger models. On both scales, the per-task speedups are smallest on the generation-style tasks (e.g., DROP), where all methods converge toward a high-entropy loss plateau and the absolute loss gaps near the target are small.

Table 12: Wall-clock time-to-target on OPT-1.3B (seconds; lower is better). The common target is MeZO's final smoothed training loss. The last row reports the end-to-end speedup of GaNsZO over HiZOO.

Method	SST-2	RTE	CB	BoolQ	WSC	WiC	MultiRC	COPA	ReCoRD	DROP
MeZO	2921	5905	8188	9022	4200	1332	14725	2216	4918	11703
HiZOO	4488	6317	13489	10918	4518	1748	6468	3932	5052	18363
GaNsZO	**729**	**956**	**2334**	**1800**	**272**	**197**	**941**	**1180**	**1105**	**7312**
Speedup vs. HiZOO	6.2×	6.6×	5.8×	6.1×	16.6×	8.9×	6.9×	3.3×	4.6×	2.5×

Table 13: Wall-clock time-to-target on Llama-3.1-8B (seconds; lower is better). The common target is MeZO's final smoothed training loss, following the same protocol as the OPT-1.3B comparison. The last row reports the end-to-end speedup of GaNsZO over HiZOO-L.

Method	SST-2	RTE	CB	BoolQ	WSC	WiC	MultiRC	COPA	ReCoRD	DROP
MeZO	13941	29332	22978	24457	21354	16988	26200	10334	25022	26826
HiZOO-L	27646	23687	37081	30275	†	5927	44617	†	20597	22046
GaNsZO	**4774**	**3608**	**9630**	**11382**	**1326**	**445**	**6287**	**8369**	**9507**	**10582**
Speedup vs. HiZOO-L	5.8×	6.6×	3.9×	2.7×	—	13.3×	7.1×	—	2.2×	2.1×

† On WSC and COPA, the retained HiZOO-L training runs terminated before reaching MeZO's final loss, so a reliable time-to-target cannot be computed for HiZOO-L on these two tasks; we mark these cells as unavailable. GaNsZO still reaches the target on both (1326 s and 8369 s, respectively).

B.5 Hessian and Guass-Newton Behavior

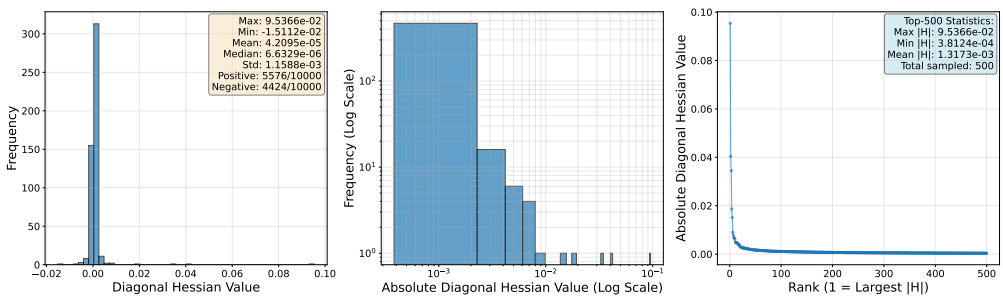
Hessian Behavior in LLM. In Section 3.2, we identify that the diagonal Hessian of a weight matrix $\mathbf{W}_Q$ exhibits a strongly skewed distribution. In this section we further investigate the statistical structure of the diagonal Hessian on a representative parameter matrix—the key projection $\mathbf{W}_K$ (of the 7th self-attention layer). We uniformly sample 10,000 coordinate positions within this matrix, and for each sampled coordinate we compute the second derivative of the training loss with respect to that parameter using automatic differentiation on a fixed mini-batch of sufficiently large batchsize.

Figure 5 presents thorough results of the distribution of the sampled entries, including the statistical characteristics (mean, median, etc.) of the raw diagonal hessian value. $\mathbf{W}_K$ exhibits a similar long-tailed distribution throughout fine-tuning to the $\mathbf{W}_Q$ in Figure 1a. To deal with this near-zero-dominated structure, stabilization methods such as damping, clipping or Trust-region step selection are usually required for second-order optimization algorithms, undermining their practicality. By contrast, GaNsZO focuses on the curvature along descent direction. Under common perturbation settings, this scalar can be interpreted as an aggregate or mean (up to a $1/d$ factor) measure of diagonal Hessian ($\mathbb{E}_{\mathbf{z}}[\tilde{\sigma}_t] = \mathrm{tr}(\nabla^2 L(\boldsymbol{\theta_t}))$). Because it aggregates information across coordinates, this directional Gauss–Newton scalar is less sensitive to per-coordinate noise and hence yields more stable preconditioning in practice.

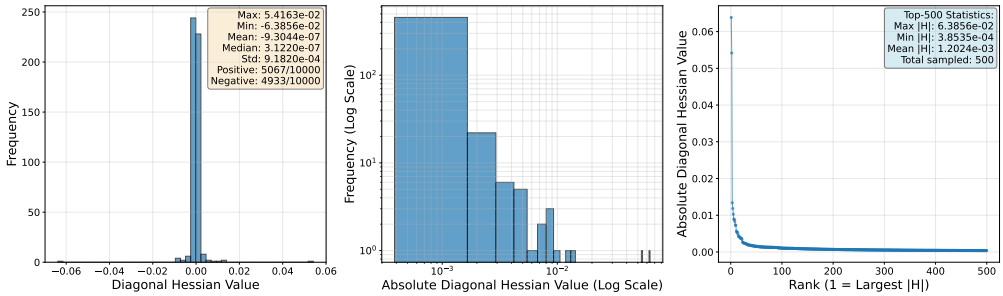
Gauss-Newton versus Full Hessian Curvature. In Section 4.2, we take the Gauss-Newton (GN) matrix as an approximation of the full Hessian and use it for the curvature estimation. We here provide empirical evidence that the GN curvature closely tracks the full-Hessian curvature.

Since computing the full Hessian and GN matrix is prohibitively expensive for LLMs, we compare their directional curvatures along 200 sampled directions. We consider both random sampling and stratified sampling, where directions are drawn from different parameter groups (including embedding, attention, and FC layers), at checkpoints 4K and 10K. For a direction $\mathbf{z}$, we compare the Hessian curvature $\mathbf{z}^\top \mathbf{H}\mathbf{z}$ with the GN curvature $\mathbf{z}^\top \mathbf{G}\mathbf{z}$, and report their absolute and relative errors, together with sign agreement.

As shown in Table 14 and Figure 6, the GN directional curvatures generally remain close to those of the full Hessian. In particular, we observe small absolute and relative errors in many sampled directions, and the agreement becomes stronger at later training stages (e.g., 10K). These results



(a) SST-2, at Step 220



(b) CB, at Step 370

Figure 5: Distribution of diagonal Hessian values on SST-2 and CB during fine-tuning of OPT-1.3B. We sample 10,000 entries from the W_K parameters in the 7th self-attention layer as an empirical approximation. *Top*: Full distribution of second-order derivatives. *Bottom*: Distribution of absolute values (log scale) and the top-500 largest absolute values.

Table 14: Directional curvature comparison between the full Hessian and the Gauss-Newton term on OPT-1.3B. We report absolute and relative errors, together with sign agreement.

Model	Sampling	Abs. err.	Med. abs.	Std. abs.	Rel. err.	Med. rel.	Std. rel.	Sign (%)
OPT-1.3B (4K)	Random	3.7×10^{-6}	0	1.5×10^{-5}	0.26	0	0.70	80
OPT-1.3B (4K)	Stratified	4.1×10^{-5}	1.1×10^{-8}	5.0×10^{-4}	0.41	0	1.46	67
OPT-1.3B (10K)	Random	5.6×10^{-6}	0	3.1×10^{-5}	0.19	0	0.89	85
OPT-1.3B (10K)	Stratified	1.6×10^{-5}	4.9×10^{-8}	9.0×10^{-5}	0.29	0	0.57	73

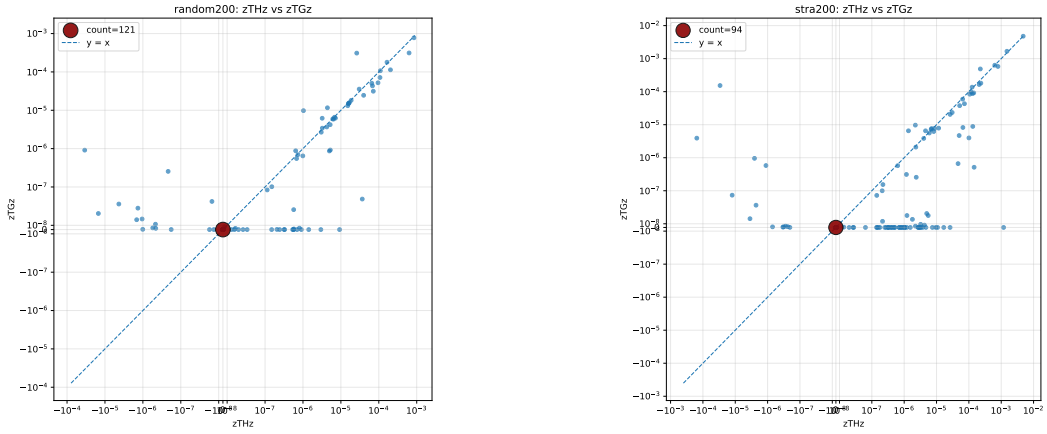
provide empirical support for using the GN term as a practical surrogate for the Hessian curvature in our analysis.

B.6 Diagnostics on the Curvature Scalar

Building on the temporal analysis of the curvature scalar in Section 5.6, we provide additional diagnostics on σ_t, focusing on its stability over time and its comparison with coordinate-wise curvature estimators.

To assess robustness, we compare the stability of the effective step size in GaNsZO against HiZOO. Since the two methods operate at different granularities, we align their scales by comparing the scalar effective step size η/σ_t in GaNsZO with the coordinate-wise effective step sizes $\eta/h_{t,i}$ in HiZOO. Specifically, we summarize the coordinate-wise behavior of HiZOO using the coordinate-averaged effective step size $m_t = \mathrm{mean}_i(\eta/h_{t,i})$, and report its across-iteration variance under two EMA settings, $\alpha_1 = 10^{-6}$ and $\alpha_2 = 10^{-8}$. We additionally report representative coordinate-wise statistics to capture per-parameter variability that is not reflected in the averaged quantity.

As shown in Table 15, GaNsZO achieves consistently low variance in the effective step size, comparable to or smaller than HiZOO across both EMA settings. Notably, while HiZOO may exhibit very small variance under aggressive smoothing (e.g., α_2), this often corresponds to degenerate or



(a) Random sampling

(b) Stratified sampling

Figure 6: Directional curvature comparison between the full Hessian and the Gauss-Newton term on OPT-1.3B. The concentration of points near the origin (the red dot) indicates close agreement in many sampled directions.

Table 15: Across-iteration variance of the effective step size, complementing the temporal diagnostics in the main text. For GaNsZO, we report $\mathrm{Var}(\eta/\sigma_t)$. For HiZOO, we report the variance of the coordinate-averaged effective step size $m_t = \mathrm{mean}_i(\eta/h_{t,i})$ under two EMA settings ($\alpha_1 = 10^{-6}$, $\alpha_2 = 10^{-8}$). We also include representative coordinate-wise statistics to illustrate per-parameter variability not captured by the averaged quantity.

Model	GaNsZO var(η/σ_t)	HiZOO var(m_t), α_1	HiZOO var(m_t), α_2	Coord. 1 (α_1)	Coord. 2 (α_2)
OPT-1.3B	3.6×10^{-14}	5.7×10^{-14}	4.8×10^{-44}	1.5×10^{-13}	9.9×10^{-15}
Llama3.1-8B	1.1×10^{-14}	3.4×10^{-14}	3.2×10^{-16}	1.5×10^{-14}	7.8×10^{-15}

near-constant effective step sizes, rather than stable yet adaptive behavior. These results indicate that GaNsZO provides a more stable and well-conditioned effective step size, avoiding both excessive coordinate-wise variability and over-smoothing effects, thereby improving robustness compared to coordinate-wise curvature estimators.

C PROOFS.

This section proves the convergence of the proposed algorithms. For analytical convenience, we consider the limit as $\mu \to 0$, and clip σ_{t-1} to $[c_0, c_1]$ prior to the EMA step. Under these conditions, Algorithm 1 reduces to Algorithm 1.

Remark C.1. During the experiment, we found that θ_t remained stable with minor fluctuations. Therefore, the clipping of σ_t is unnecessary in practice.

To analyze the convergence of Algorithm 2, we need the following assumptions:

Assumption C.2 (Effective dimension)**.** For any $\boldsymbol{\theta}$, there exists a neighborhood $B(\boldsymbol{\theta}, r(\boldsymbol{\theta}))$ such that

$$\nabla^2 L(\boldsymbol{\theta}) \preceq \hat{\nabla}^2 L(\boldsymbol{\theta}) \preceq L_1 \mathbf{I}, \mathrm{tr}(\hat{\nabla}^2 L(\boldsymbol{\theta})) \leq d_{\mathrm{eff}} L_1. \tag{8}$$

Without loss of generality, we assume that in Algorithm 2, the iterate always satisfies $\boldsymbol{\theta}_{t+1} \in B(\boldsymbol{\theta}_t, r(\boldsymbol{\theta}_t))$. If not, $\boldsymbol{\theta}_{t+1}$ can be clipped to this neighborhood.

The effective-dimension assumption is standard in the analysis of zeroth-order optimization methods Malladi et al. (2023). Empirically, LLM Hessians are often observed to exhibit low-rank or highly skewed spectral structure, including in fine-tuning settings Liang et al. (2026). Moreover, the effective-dimension assumption may still hold even when the Hessian spectrum decays more slowly: recent studies suggest that LLM Hessians can exhibit power-law spectral behavior of the

form $\lambda_k \approx \lambda_1 k^{-s}$ for some $s > 1$ Tang et al. (2025). In this case,

$$\frac{\mathrm{tr}(\nabla^2 L)}{\lambda_{\max}(\nabla^2 L)} \approx \sum_{k=1}^{d} k^{-s},$$

which remains finite whenever $s > 1$. As shown in Tables 3 and 4 of Tang et al. (2025), the fitted s is always larger than 1. Therefore, such heavy-tailed spectra are still compatible with a bounded effective dimension.

Assumption C.3 (Hessian decomposition)**.**

$$\gamma_1 \hat{\nabla}^2 L(\boldsymbol{\theta}) \preceq \mathbb{E}_{\mathcal{B}} \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B})^\top \nabla_{\mathbf{h}} g(h(\boldsymbol{\theta}); \mathcal{B}) \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B}) \preceq \gamma_2 \hat{\nabla}^2 L(\boldsymbol{\theta}). \tag{9}$$

Assumption C.4 (stochastic gradient)**.** For batches $\mathcal{B}$, we have:

$$\mathbb{E}_{\mathcal{B}} \left(\nabla_{\boldsymbol{\theta}} g(h(\boldsymbol{\theta}; \mathcal{B})) - \nabla_{\boldsymbol{\theta}} L(\theta) \right) \left(\nabla_{\boldsymbol{\theta}} g(h(\boldsymbol{\theta}; \mathcal{B})) - \nabla_{\boldsymbol{\theta}} L(\theta) \right)^\top = \boldsymbol{\Sigma}, \tag{10}$$

and

$$\mathrm{tr}(\boldsymbol{\Sigma}) = \mathbb{E}_{\mathcal{B}} \| \nabla_{\boldsymbol{\theta}} g(h(\boldsymbol{\theta}; \mathcal{B})) - \nabla_{\boldsymbol{\theta}} L(\theta) \|^2 \leq \sigma^2. \tag{11}$$

Assumption C.5 (stochastic Hessian)**.** For batches $\mathcal{B}$, we have:

$$\beta_1 \mathbb{E}_{\mathcal{B}} \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B})^\top \nabla_{\mathbf{h}} g(h(\boldsymbol{\theta}); \mathcal{B}) \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B}) \preceq \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B})^\top \nabla_{\mathbf{h}} g(h(\boldsymbol{\theta}); \mathcal{B}) \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B}), \tag{12}$$

$$\beta_2 \mathbb{E}_{\mathcal{B}} \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B})^\top \nabla_{\mathbf{h}} g(h(\boldsymbol{\theta}); \mathcal{B}) \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B}) \succeq \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B})^\top \nabla_{\mathbf{h}} g(h(\boldsymbol{\theta}); \mathcal{B}) \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B}). \tag{13}$$

Now, we propose Theorem C.6 to analyze the convergence of Algorithm 2.

Theorem C.6. *Under Assumptions C.2, C.3, C.4 and C.5, if the step size η_t satisfies*

$$\eta_t \leq \min \left\{ \frac{\alpha \gamma_1^2 \beta_1^2}{2\gamma_2 \beta_2}, \frac{1}{12\pi} \frac{\epsilon^2}{\sigma^2} \cdot \frac{\gamma_1 \beta_1 (1-\alpha) \sigma_{t-1}}{\alpha \gamma_2 \beta_2 d_{\mathrm{eff}} L_1 + (1-\alpha)\sigma_{t-1}} \right\}, \tag{14}$$

Algorithm 2 finds an ϵ-stationary point in

$$O\left(\max \left\{ \frac{\Delta}{\epsilon^2} \frac{(c' + (1-\alpha)c_1)\, \gamma_2 \beta_2}{\gamma_1^2 \beta_1^2}, \frac{\Delta \sigma^2}{\epsilon^4} \frac{((c' + (1-\alpha)c_1)(c' + ((1-\alpha)\, c_0)}{\gamma_1 \beta_1 (1-\alpha) c_0} \right\} \right)$$

iterations, where $c' = \alpha \gamma_2 \beta_2 d_{\mathrm{eff}} L_1$.

Remark C.7. When $(1-\alpha)c_0 = \Theta\left(d_{\mathrm{eff}} L_1\right)$ and $(1-\alpha)c_1 = \Theta\left(d_{\mathrm{eff}} L_1\right)$, Theorem C.6 implies that Algorithm 2 finds an ϵ-stationary point in $O\left(d_{\mathrm{eff}} \cdot \max\left\{ \frac{\Delta L_1}{\epsilon^2}, \frac{\delta L_1 \sigma^2}{\epsilon^4} \right\}\right)$ iterations.

Proof. By Assumption C.2, we have:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z},\mathcal{B}} L(\boldsymbol{\theta}_{t+1}) - L(\boldsymbol{\theta}_t) \\
&\leq -\mathbb{E}_{\mathbf{z},\mathcal{B}} \left\langle \nabla_{\boldsymbol{\theta}} L(\theta_t), -\eta \sigma_t^{-1} \hat{\boldsymbol{g}} \right\rangle + \mathbb{E}_{\mathbf{z},\mathcal{B}} \frac{\eta^2}{2\sigma_t^2} \| \hat{\boldsymbol{g}} \|^2_{\hat{\nabla}^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t)} \\
&= -\eta_t \mathbb{E}_{\mathbf{z},\mathcal{B}} \sigma_t^{-1} \left\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \right\rangle \left\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathcal{B}), \mathbf{z} \right\rangle + \eta_t^2 \mathbb{E}_{\mathbf{z},\mathcal{B}} \frac{\left\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathcal{B}), \mathbf{z} \right\rangle^2 \mathbf{z}^\top \hat{\nabla}^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) \mathbf{z}}{2\sigma_t^2}.
\end{aligned} \tag{15}$$

For notatinal simplicity, we define

$$\begin{aligned}
\mathbf{B} &= \mathbb{E}_{\mathcal{B}} \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B})^\top \nabla_{\mathbf{h}} g(h(\boldsymbol{\theta}); \mathcal{B}) \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B}) \\
\psi_2 &= \mathbf{z}^\top \hat{\nabla}^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \mathbf{z}.
\end{aligned}$$

We use Assumption C.5 to obtain two inequalities:

$$\sigma_t \leq (1-\alpha)\sigma_{t-1} + \alpha \beta_2 \mathbf{z}^\top \mathbf{B} \mathbf{z} \leq (1-\alpha)\sigma_{t-1} + \alpha \gamma_2 \beta_2 \psi_2, \tag{16}$$

$$\sigma_t \geq (1-\alpha)\sigma_{t-1} + \alpha \beta_1 \mathbf{z}^\top \mathbf{B} \mathbf{z} \geq (1-\alpha)\sigma_{t-1} + \alpha \gamma_1 \beta_1 \psi_2. \tag{17}$$

Algorithm 2 GaNsZO ($\mu \to 0$)

Require: Parameters $\boldsymbol{\theta}$, loss L, network output h, outer loss g, steps T, perturbation μ, step size η_t, EMA rate α, $\sigma_0 = 1$, distribution $\mathcal{D}$

for $t = 0, \dots, T-1$ **do**
  Sample batch $\mathcal{B}$ and seed s
  (A) Sample random direction
  Reset RNG with seed s
  Sample $\mathbf{z} \sim \mathcal{D}$ ▷ e.g., Gaussian or Rademacher
  (B) Gauss–Newton directional curvature
  $\hat{\sigma}_t \leftarrow \mathbf{z}^\top \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B})^\top \nabla^2 g(\mathbf{h}) \mathbf{J}_h(\boldsymbol{\theta}; \mathcal{B}) \mathbf{z}$
  (C) EMA smoothing and stabilization
  $\sigma_{t-1} \leftarrow \min\{\sigma_{t-1}, c_1\}$
  $\sigma_{t-1} \leftarrow \max\{\sigma_{t-1}, c_0\}$
  $\sigma_t \leftarrow (1-\alpha)\sigma_{t-1} + \alpha\, \hat{\sigma}_t$
  (D) ZO gradient estimation
  $\hat{\boldsymbol{g}} \leftarrow \langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathcal{B}), \mathbf{z} \rangle \mathbf{z}$
  (E) Preconditioned update
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta_t\, \sigma_t^{-1}\, \hat{\boldsymbol{g}}$ ▷ c_0 is a small constant
end for

We bound the first term and the second term of (15) separately. For the first term, by (16), we have:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z},\mathcal{B}} \sigma_t^{-1} \langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle \langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathcal{B}), \mathbf{z} \rangle \\
&\geq \mathbb{E}_{\mathbf{z},\mathcal{B}} \frac{1}{(1-\alpha)\sigma_{t-1} + \alpha\gamma_2\beta_2\psi_2} \langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle \langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathcal{B}), \mathbf{z} \rangle \\
&= \mathbb{E}_{\mathbf{z}} \frac{1}{(1-\alpha)\sigma_{t-1} + \alpha\gamma_2\beta_2\psi_2} \langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle^2 . \qquad (18)
\end{aligned}$$

For the second term, by (17), we have:

$$\begin{aligned}
\mathbb{E}_{\mathbf{z},\mathcal{B}} \frac{\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathcal{B}), \mathbf{z} \rangle^2 \mathbf{z}^\top \hat{\nabla}^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) \mathbf{z}}{2\sigma_t^2} &\leq \mathbb{E}_{\mathbf{z},\mathcal{B}} \frac{\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathcal{B}), \mathbf{z} \rangle^2 \mathbf{z}^\top \hat{\nabla}^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t) \mathbf{z}}{2((1-\alpha)\sigma_{t-1} + \alpha\gamma_1\beta_1\psi_2)^2} \\
&\leq \mathbb{E}_{\mathbf{z}} \frac{\left(\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle^2 + \mathbf{z}^\top \boldsymbol{\Sigma} \mathbf{z} \right) \psi_2}{2((1-\alpha)\sigma_{t-1} + \alpha\gamma_1\beta_1\psi_2)^2} \\
&\leq \mathbb{E}_{\mathbf{z}} \frac{\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle^2 + \mathbf{z}^\top \boldsymbol{\Sigma} \mathbf{z}}{2\alpha\gamma_1\beta_1((1-\alpha)\sigma_{t-1} + \alpha\gamma_1\beta_1\psi_2)} . \qquad (19)
\end{aligned}$$

Summing up (18) and (19), we obtain:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{z},\mathcal{B}} L(\boldsymbol{\theta}_{t+1}) - L(\boldsymbol{\theta}_t) \\
&\leq -\eta_t \mathbb{E}_{\mathbf{z}} \frac{1}{(1-\alpha)\sigma_{t-1} + \alpha\gamma_2\beta_2\psi_2} \langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle^2 + \eta_t^2 \mathbb{E}_{\mathbf{z}} \frac{\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle^2 + \mathbf{z}^\top \boldsymbol{\Sigma} \mathbf{z}}{2\alpha\gamma_1\beta_1((1-\alpha)\sigma_{t-1} + \alpha\gamma_1\beta_1\psi_2)} \\
&\leq -\left(\eta_t - \frac{\eta_t^2}{2\alpha\gamma_1^2\beta_1^2/(\gamma_2\beta_2)} \right) \mathbb{E}_{\mathbf{z}} \frac{\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle^2}{(1-\alpha)\sigma_{t-1} + \alpha\gamma_2\beta_2\psi_2} + \frac{\eta_t^2}{2\alpha\gamma_1\beta_1} \mathbb{E}_{\mathbf{z}} \frac{\mathbf{z}^\top \boldsymbol{\Sigma} \mathbf{z}}{(1-\alpha)\sigma_{t-1} + \alpha\gamma_1\beta_1\psi_2} . \qquad (20)
\end{aligned}$$

For the first term of (20), by Cauchy-Schwarz inequality, we have:

$$\begin{aligned}
\mathbb{E}_{\mathbf{z}} \frac{\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle^2}{(1-\alpha)\sigma_{t-1} + \alpha\gamma_2\beta_2\psi_2} &\geq \frac{\left(\mathbb{E}_{\mathbf{z}} |\langle \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}), \mathbf{z} \rangle| \right)^2}{\mathbb{E}_{\mathbf{z}} \left((1-\alpha)\sigma_{t-1} + \alpha\gamma_2\beta_2\psi_2 \right)} \\
&= \frac{\left(\sqrt{\frac{2}{\pi}} \|\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})\| \right)^2}{\alpha\gamma_2\beta_2 \operatorname{tr}(\hat{\nabla}^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta})) + (1-\alpha)\sigma_{t-1}} \\
&= \frac{2}{\pi} \cdot \frac{\|\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})\|^2}{\alpha\gamma_2\beta_2 \operatorname{tr}(\hat{\nabla}^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta})) + (1-\alpha)\sigma_{t-1}} . \qquad (21)
\end{aligned}$$

For the second term of (20), according to Lemma C.8, we have:

$$\begin{aligned}\mathbb{E}_{\mathbf{z}}\frac{\mathbf{z}^\top\boldsymbol{\Sigma}\mathbf{z}}{(1-\alpha)\sigma_{t-1}+\alpha\gamma_1\beta_1\psi_2} &\le \frac{3}{d(d+2)}\operatorname{tr}(\boldsymbol{\Sigma})\operatorname{tr}\left(\left(\alpha\gamma_1\beta_1\nabla^2 L(\boldsymbol{\theta})+\frac{(1-\alpha)\sigma_{t-1}}{d}\mathbf{I}\right)^{-1}\right)\\ &\le \frac{3}{d(d+2)}\sigma^2\cdot d\cdot\frac{d}{(1-\alpha)\sigma_{t-1}}\\ &\le \frac{6\sigma^2}{(1-\alpha)\sigma_{t-1}}. \end{aligned} \tag{22}$$

Assume that $\|\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t)\| > \epsilon$, or $\boldsymbol{\theta}_t$ is already an ϵ-stationary point. Then, summing up (21) and (22), by the choice of η_t, we have:

$$\begin{aligned}&\mathbb{E}_{\mathbf{z},\mathcal{B}} L(\boldsymbol{\theta}_{t+1}) - L(\boldsymbol{\theta}_t)\\ &\le -\left(\eta_t - \frac{\eta_t^2}{2\alpha\gamma_1^2\beta_1^2/(\gamma_2\beta_2)}\right)\mathbb{E}_{\mathbf{z}}\frac{\langle\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t),\mathbf{z}\rangle^2}{(1-\alpha)\sigma_{t-1}+\alpha\gamma_2\beta_2\psi_2} + \frac{\eta_t^2}{2\alpha\gamma_1\beta_1}\mathbb{E}_{\mathbf{z}}\frac{\mathbf{z}^\top\boldsymbol{\Sigma}\mathbf{z}}{(1-\alpha)\sigma_{t-1}+\alpha\gamma_1\beta_1\psi_2}\\ &\le -\frac{3\eta_t}{4}\cdot\frac{2}{\pi}\cdot\frac{\|\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t)\|^2}{\alpha\gamma_2\beta_2 d_{\text{eff}} L_1+(1-\alpha)\sigma_{t-1}} + \frac{\eta_t^2}{2\alpha\gamma_1\beta_1}\cdot\frac{6\sigma^2}{(1-\alpha)\sigma_{t-1}}\\ &\le -\frac{\eta_t}{\pi}\frac{\|\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t)\|^2}{\alpha\gamma_2\beta_2 d_{\text{eff}} L_1+(1-\alpha)c_1}. \end{aligned} \tag{23}$$

Summing up (23) over t, we have:

$$\begin{aligned}\mathbb{E}_{\mathbf{z}_{0:T-1},\mathcal{B}_{0:T-1}} L(\boldsymbol{\theta}_{T+1}) - L(\boldsymbol{\theta}_0) &\le \sum_{t=0}^{T-1}\mathbb{E}_{\mathbf{z}_t,\mathcal{B}_t} L(\boldsymbol{\theta}_{t+1}) - L(\boldsymbol{\theta}_t)\\ &\le -\sum_{t=0}^{T-1}\frac{\eta_t}{\pi}\frac{\|\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta_t})\|^2}{\alpha\gamma_2\beta_2\operatorname{tr}(\nabla^2_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t))+(1-\alpha)c_1}\\ &\le -\sum_{t=0}^{T-1}\frac{\eta_t\|\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta_t})\|^2}{\pi(\alpha\gamma_2\beta_2 d_{\text{eff}} L_1+(1-\alpha)c_1)}. \end{aligned} \tag{24}$$

Therefore, denote $c' = \alpha\gamma_2\beta_2 d_{\text{eff}} L_1$. Algorithm 2 finds an ϵ-stationary point in

$$O\left(\max\left\{\frac{\Delta}{\epsilon^2}\frac{(c'+(1-\alpha)c_1)\gamma_2\beta_2}{\gamma_1^2\beta_1^2}, \frac{\Delta\sigma^2}{\epsilon^4}\frac{(c'+(1-\alpha)c_1)(c'+(1-\alpha)c_0)}{\gamma_1\beta_1(1-\alpha)c_0}\right\}\right)$$

iterations. □

Lemma C.8. *For a random vector $\mathbf{z}\sim N(0,\mathbf{I}_d)$, positive definite matrices $\mathbf{A}$ and $\mathbf{B}$, and a constant $c>0$, we have:*

$$\mathbb{E}_{\mathbf{z}}\frac{\mathbf{z}^\top\mathbf{A}\mathbf{z}}{\mathbf{z}^\top\mathbf{B}\mathbf{z}+c} \le \frac{3}{d(d+2)}\operatorname{tr}(\mathbf{A})\operatorname{tr}\left(\left(\mathbf{B}+\frac{c}{d}\mathbf{I}_d\right)^{-1}\right). \tag{25}$$

Proof. Since $\mathbf{z}\sim N(0,\mathbf{I}_d)$, write $\mathbf{z}=r\mathbf{u}$, where $r=\|\mathbf{z}\|$ and $\mathbf{u}=\mathbf{z}/\|\mathbf{z}\|$ is uniformly distributed on the unit sphere S^{d-1}. Then $r^2\sim\chi^2(d)$, and r and θ are independent. Thus,

$$\frac{\mathbf{z}^\top\mathbf{A}\mathbf{z}}{\mathbf{z}^\top\mathbf{B}\mathbf{z}+c} = \frac{r^2\mathbf{u}^\top\mathbf{A}\mathbf{u}}{r^2\mathbf{u}^\top\mathbf{B}\mathbf{u}+c}.$$

For fixed $\mathbf{u}$, let $\alpha=\mathbf{u}^\top\mathbf{B}\mathbf{u}>0$. Define $\varphi(x)=\frac{x}{\alpha x+c}$ for $x>0$. Since $\varphi''(x)=-\frac{2\alpha c}{(\alpha x+c)^3}<0$, φ is concave. By Jensen's inequality,

$$\mathbb{E}_r\left[\frac{r^2}{\alpha r^2+c}\right] \le \frac{\mathbb{E}[r^2]}{\alpha\mathbb{E}[r^2]+c}.$$

As $\mathbb{E}[r^2] = d$, we have

$$\mathbb{E}_r\left[\frac{r^2}{\alpha r^2 + c}\right] \leq \frac{d}{\alpha d + c}.$$

Hence,

$$\mathbb{E}\left[\frac{\mathbf{z}^\top \mathbf{A}\mathbf{z}}{\mathbf{z}^\top \mathbf{B}\mathbf{z} + c}\right] = \mathbb{E}_{\mathbf{u}}\left[\mathbf{u}^\top \mathbf{A}\mathbf{u} \cdot \mathbb{E}_r\left[\frac{r^2}{\alpha r^2 + c}\right]\right] \leq \mathbb{E}_{\mathbf{u}}\left[\mathbf{u}^\top \mathbf{A}\mathbf{u} \cdot \frac{d}{d\mathbf{u}^\top \mathbf{B}\mathbf{u} + c}\right] = \mathbb{E}_{\mathbf{u}}\left[\frac{\mathbf{u}^\top \mathbf{A}\mathbf{u}}{\mathbf{u}^\top \mathbf{B}\mathbf{u} + c/d}\right].$$

Define $\mathbf{M} = \mathbf{B} + \frac{c}{d}\mathbf{I}_d$, which is positive definite. Then $\mathbf{u}^\top \mathbf{B}\mathbf{u} + \frac{c}{d} = \mathbf{u}^\top \mathbf{M}\mathbf{u}$. For any unit vector $\mathbf{u}$ and positive definite matrix $\mathbf{M}$, the Cauchy-Schwarz inequality yields

$$1 = (\mathbf{u}^\top \mathbf{u})^2 = (\mathbf{u}^\top \mathbf{M}^{1/2} \mathbf{M}^{-1/2} \mathbf{u})^2 \leq (\mathbf{u}^\top \mathbf{M}\mathbf{u})(\mathbf{u}^\top \mathbf{M}^{-1}\mathbf{u}),$$

so

$$\frac{1}{\theta^\top \mathbf{M}\theta} \leq \theta^\top \mathbf{M}^{-1}\theta.$$

Therefore,

$$\frac{\mathbf{u}^\top \mathbf{A}\mathbf{u}}{\mathbf{u}^\top \mathbf{B}\mathbf{u} + c/d} \leq (\mathbf{u}^\top \mathbf{A}\mathbf{u})(\mathbf{u}^\top \mathbf{M}^{-1}\mathbf{u}).$$

Let $\mathbf{C} = \mathbf{M}^{-1} = \left(\mathbf{B} + \frac{c}{d}\mathbf{I}_d\right)^{-1}$. For $\mathbf{u}$ uniformly distributed on S^{d-1}, the fourth moments are:

$$\mathbb{E}[\mathbf{u}_i \mathbf{u}_j \mathbf{u}_k \mathbf{u}_l] = \frac{1}{d(d+2)}(\delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}),$$

where δ_{ab} is the Kronecker delta. For symmetric matrices $\mathbf{A}$ and $\mathbf{C}$, this implies

$$\mathbb{E}_{\mathbf{u}}\left[(\mathbf{u}^\top \mathbf{A}\mathbf{u})(\mathbf{u}^\top \mathbf{C}\mathbf{u})\right] = \frac{1}{d(d+2)}\left(\operatorname{tr}(\mathbf{A})\operatorname{tr}(\mathbf{C}) + 2\operatorname{tr}(\mathbf{A}\mathbf{C})\right).$$

Since $\mathbf{A}$ and $\mathbf{C}$ are positive definite, $\operatorname{tr}(\mathbf{A}\mathbf{C}) \leq \operatorname{tr}(\mathbf{A})\|\mathbf{C}\|_{\mathrm{op}} \leq \operatorname{tr}(\mathbf{A})\operatorname{tr}(\mathbf{C})$, where $\|\mathbf{C}\|_{\mathrm{op}}$ is the operator norm of $\mathbf{C}$. Thus,

$$\mathbb{E}_{\mathbf{u}}\left[(\mathbf{u}^\top \mathbf{A}\mathbf{u})(\mathbf{u}^\top \mathbf{C}\mathbf{u})\right] \leq \frac{1}{d(d+2)}\left(\operatorname{tr}(\mathbf{A})\operatorname{tr}(\mathbf{C}) + 2\operatorname{tr}(\mathbf{A})\operatorname{tr}(\mathbf{C})\right) = \frac{3}{d(d+2)}\operatorname{tr}(\mathbf{A})\operatorname{tr}(\mathbf{C}).$$

Putting everything together, we have:

$$\mathbb{E}\left[\frac{\mathbf{z}^\top \mathbf{A}\mathbf{z}}{\mathbf{z}^\top \mathbf{B}\mathbf{z} + c}\right] \leq \mathbb{E}_{\mathbf{u}}\left[\frac{\mathbf{u}^\top \mathbf{A}\mathbf{u}}{\mathbf{u}^\top \mathbf{B}\mathbf{u} + c/d}\right] \leq \mathbb{E}_{\mathbf{u}}\left[(\mathbf{u}^\top \mathbf{A}\mathbf{u})(\mathbf{u}^\top \mathbf{C}\mathbf{u})\right] \leq \frac{3}{d(d+2)}\operatorname{tr}(\mathbf{A})\operatorname{tr}(\mathbf{C}),$$

with $\mathbf{C} = \left(\mathbf{B} + \frac{c}{d}\mathbf{I}_d\right)^{-1}$. This completes the proof. □

D More Discussions on GaNsZO

GaNsZO incorporates second-order information through a stepsize scaling factor that approximates the directional curvature along the current descent direction. In practice, we estimate the Gauss-Newton quadratic form

$$\mathbf{z}^\top \mathbf{G}\mathbf{z} \;=\; \big(\mathbf{J}_h(\boldsymbol{\theta})\mathbf{z}\big)^\top \nabla^2 g\big(h(\boldsymbol{\theta})\big)\big(\mathbf{J}_h(\boldsymbol{\theta})\mathbf{z}\big),$$

where $L = g \circ h$ and $\mathbf{G}$ denotes the Gauss-Newton matrix. This choice has two important advantages. First, the Gauss-Newton term captures the dominant positive-semidefinite component of the curvature for common loss functions, providing a stable surrogate for directional second-order information. Second, exploiting the composite structure $g \circ h$ enables efficient computation of the directional quadratic form using zeroth-order perturbations on h combined with low-cost autodiff restricted to g, without forming or storing an $O(d)$ diagonal curvature estimate. Moreover, as g corresponds to the composition of the output linear head and the cross-entropy loss, the resulting Gauss-Newton matrix is positive definite, ensuring that the induced stepsize scaling is well-defined and positive (while the η^* in Eq. (3) may not be positive when $\mathbf{H}$ is not positive definite).

For stability, we initialize the scalar factor σ_0 as 1. This initialization can be substantially different from the true magnitude of the directional second derivative, and therefore a separate global stepsize η is still required to control the overall update scale. After computing the scalar factor σ_t, GaNsZO updates the model as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \sigma_t^{-1} \hat{\boldsymbol{g}},$$

where $\hat{\boldsymbol{g}}$ is the ZO gradient estimate. Although the preconditioner does not alter the descent direction $\hat{\boldsymbol{g}}$, it controls the magnitude of the update according the curvature information and provides more robust curvature estimation compared to HiZOO. On the other hand, while the diagonal preconditioner Σ_t^{-1} in HiZOO can theoretically adjust the descent direction to minimize variance, it only leads to inaccurate or unstable estimation. In fact, according to our experiment of true diagonal Hessian in Section 3.2, the assumption for the diagonal Hessian to accelerate (e.g., Eq. 3.1 in Ye et al. (2018)) is unlikely to hold, further indicating the possibility to use a scalar scaling factor instead of a diagonal preconditioner.

To summarize, GaNsZO provides a principled, low-cost way to incorporate second-order information into ZO updates: it approximates the locally optimal step length in the descent direction while avoiding the storage and noise issues associated with per-parameter Hessian estimates.