
PseuZO: Pseudo-Zeroth-Order Algorithm for Training Deep Neural Networks

Pengyun Yue^{1,*}, Xuanlin Yang^{1,5,*}, Mingqing Xiao^{1,4}, Zhouchen Lin^{1,2,3,†}

¹State Key Lab of General AI, School of Intelligence Science and Technology, Peking University

²Institute for Artificial Intelligence, Peking University

³Pazhou Laboratory (Huangpu), Guangzhou, Guangdong, China

⁴Microsoft Research Asia

⁵Zhongguancun Academy

yuepy@pku.edu.cn, xuanlinyang@stu.pku.edu.cn, {mingqing_xiao, zlin}@pku.edu.cn

Abstract

Zeroth-order Optimization (ZO) has received wide attention in machine learning, especially when computing full gradient is expensive or even impossible. Recently, ZO has emerged as an important paradigm for memory-efficient fine-tuning of large language models (LLMs), circumventing the memory overhead of backpropagation. However, existing ZO gradient estimators exhibit dimension-dependent variance scaling as $\Theta(d)$, leading to dimension-dependent convergence rates without further assumptions on the objective function, which is prohibitive for large-scale LLM parameters. To address this problem, we present a Pseudo-Zeroth-Order (PseuZO) framework for optimizing composite objective functions, especially large-scale models: $\min_{\mathbf{x} \in \mathcal{X}} \mathcal{F}(\mathbf{x}) = \mathbb{E}_{\mathbf{z}} g \circ h(\mathbf{x}; \mathbf{z})$, where h represents complex, high-dimensional representations and g is a task-specific loss. While existing zeroth-order methods estimate gradients with final loss functions, our PseuZO algorithm estimate the Jacobian matrix of $h(\mathbf{x})$ with the model output $\mathbf{o} = h(\mathbf{x})$, and the gradient of the loss function on model output $\mathbf{e} = \nabla_{\mathbf{o}} g(\mathbf{o})$, and apply exponential moving average on Jacobian estimators to reduce the variance. Moreover, we use the sliding window technique to reduce memory costs. Our algorithm achieves an $O(\max\{\alpha_1 L \epsilon^{-2}, \alpha_1 L \sigma_g^2 \epsilon^{-4}\})$ convergence rate, where α_1 is the effective dimension of \mathcal{F} . Experimental results demonstrate that PseuZO outperforms MeZO and MeZO-SVRG in classification, multiple choice and generation tasks in both full-parameter and PEFT fine-tuning settings by boosting convergence in the early stages of training. For instance, under the same computation time, with respect to SST2 task, PseuZO gets 9.8% higher accuracy than MeZO (91.2% v.s. 82.4%). With the sliding window technique, our PseuZO achieves 70% \sim 80% memory reduction compared to FO-SGD for different model sizes as PseuZO only introduced a small dimension-independent memory overhead, which enables efficient scaling of the model size. The code is available at <https://github.com/YangBigMn/PseuZO>.

1 Introduction

Zeroth-order optimization [45, 20, 36] has served as a core technique for problems where gradient calculations are impractical. These methods rely solely on function evaluations, making them uniquely suited for black-box scenarios like adversarial attacks [6, 59], reinforcement learning [8, 14], and hyperparameter tuning [23, 37]. Compared to first-order and higher-order optimization methods,

*Equal contribution.

†Corresponding author.

their key strength lies in avoiding gradient computations—a critical advantage when optimizing complex systems where automatic differentiation is infeasible or prohibitively expensive. For modern deep neural networks, this approach significantly reduces memory demands by eliminating backpropagation’s computational and memory cost. Recent advances have reinvigorated zeroth-order methods as practical tools for deep neural network training, particularly in resource-constrained environments where traditional optimization strategies struggle to scale.

Large language models (LLMs) represent the pinnacle of deep neural network architectures, achieving state-of-the-art performance across diverse language understanding and generation tasks [48, 50, 5, 16]. The standard paradigm of pretraining on web-scale corpora followed by task-specific fine-tuning has become ubiquitous, enabling these models to adapt to specialized domains. However, conventional first-order fine-tuning approaches employing full-parameter optimization through backpropagation face critical scalability barriers: the memory overhead for storing optimizer states and activation values grows with model parameters and context length, becoming prohibitive for large-scale models. This challenge has driven the emergence of parameter-efficient fine-tuning (PEFT) techniques [31, 25, 35, 55] that strategically update only subsets of model weights. Another way to reduce memory costs is using memory-efficient zeroth-order optimization (MeZO)[34]. MeZO introduces a paradigm shift by operating purely through forward-pass evaluations. By eliminating backpropagation while maintaining competitive task adaptation capability, MeZO addresses the dual requirements of memory conservation and optimization stability in resource-constrained scenarios, enabling the deployment of massive LLMs in practical applications.

While zeroth-order optimization provides a gradient-free alternative for fine-tuning large neural networks, its practical adoption faces fundamental limitations. Zeroth-order algorithms exhibit catastrophic performance when training from scratch. Even for fine-tuning tasks, the convergence rate of classical zeroth-order methods scales linearly with parameter dimension d , becoming prohibitive for modern architectures where d routinely exceeds 10^{10} . This dimension dependence persists in practice even when leveraging low-effective dimensionality theories [56] or the sparsity of the model structure [24].

In this paper, we propose a Pseudo-Zeroth-Order (PseuZO) framework for optimizing composite objective functions:

$$\min_{\mathbf{x} \in \mathcal{X}} \mathbb{E}_{\mathbf{z} \sim \mathcal{D}} g \circ h(\mathbf{x}; \mathbf{z}). \quad (1)$$

This problem formulation is prevalent in deep neural network training, where h represents complex, high-dimensional representations and g is a task-specific loss. Unlike traditional zeroth-order methods, PseuZO methods estimate the Jacobian matrix of $h(\mathbf{x})$ with the model output $\mathbf{o} = h(\mathbf{x})$. Compared to the value of the composite function $g \circ h$, the output of h provides more information about the function h . We also apply exponential moving average on Jacobian estimators to reduce the variance. Finally, we obtain the gradient estimator with the Jacobian estimator and the gradient of the loss function on model output $\mathbf{e} = \nabla_{\mathbf{o}} g(\mathbf{o})$, which can often be computed explicitly. We demonstrate the efficacy of PseuZO methods both theoretically and empirically. In theory, we prove that PseuZO method finds an ϵ -stationary point with $O(\max\{\alpha_1 L \epsilon^{-2}, \alpha_1 L \sigma_2^2 \epsilon^{-4}\})$ function evaluations, where α_1 is the effective dimension of the objective function \mathcal{F} . In our experiments, PseuZO not only converges faster, but also attains a precision improvement of up to 9.8% compared to MeZO. With the sliding window technique, PseuZO only needs a small dimension-independent extra memory overhead compared to MeZO, which enables efficient scaling of the model size. Additionally, we incorporate PseuZO with LoRA and prefix-tuning to show that PseuZO is also compatible with PEFT techniques.

We summarize our main contributions below:

1. We propose PseuZO optimization framework, which uses the differentiation of model outputs to compute a stochastic Jacobian estimator, and apply exponential moving average to reduce the variance. In practice, we use the sliding window technique to reduce memory costs.
2. We proved that the convergence rate of PseuZO method is not explicitly dependent on the parameter size. We theoretically prove that our PseuZO optimization method finds an ϵ -stationary point in $O(\max\{\alpha_1 L \epsilon^{-2}, \alpha_1 L \sigma_2^2 \epsilon^{-4}\})$ function evaluations, where α_1 is the effective dimension.

3. We conduct solid and comprehensive experiments which show that PseuZO outperforms ICL and MeZO across multiple tasks, including classification, multi-classification and generation in terms of convergence speed. With the sliding window technique, PseuZO only shows a small parameter-size-independent memory overhead compared to MeZO for instruction fine-tuning tasks. Moreover, we find that PseuZO is compatible with PEFT like LoRA and prefix-tuning, and results show that PseuZO+PEFT also outperforms MeZO+PEFT across classification, multi-classification and generation tasks in terms of convergence speed.

2 Related work

Zeroth-order optimization. Zeroth-order optimization [45, 20, 36] has been widely studied in the field of machine learning, and has been used in black-box optimization [22, 6, 59], adversarial attacks [9, 49], etc. Most zeroth-order methods are designed based on first-order [36] or higher-order methods [56], and are often d times slower where d is the dimension of the problem. To mitigate the curse of dimensionality, several works proposed effective dimension [56, 34], and characterize the convergence rate with the effective dimension of the problem. Many studies also consider reformulating the neural network at a relatively small scale to solve a simpler optimization problem [30, 47, 7], and then utilize block coordinate descent (BCD) [6] or ADMM [33] without the need for gradients. Recently, MeZO [34] successfully applied zeroth-order optimization to fine-tuning extremely large language models by efficiently estimating gradients in memory. After that, many works attempt to improve the performance of MeZO by reducing variance [19] or introducing estimated second-order information [60]. In the research of Spike Neural Network (SNN), [54] used the model output to estimate the Jacobian matrix, but their work was designed from biological applicability, and was not able to save memory costs. Inspired by [54], we designed our PseuZO optimization framework.

Memory-efficient backpropagation. As LLMs are typically fine-tuned by first-order algorithms like SGD [42] and Adam [28], many new methods or techniques have been proposed to solve the memory overhead problem, e.g. sparsifying gradients [46, 53] and quantization [15]. Other useful techniques to save memory for activation values or optimizer states like Gradient Checkpoint [18], Flash Attention [12] and Zero Redundancy Optimizer (ZeRO) [41, 39, 40]. However, these methods either sacrifice precision or require more computation time.

Gradient-free adaptation of LLMs. Language models can understand language and learn to communicate with humans after the pre-training phase. They can then generalize to tasks without training and this form adaptation that requires appropriate prompt designs is called in-context learning (ICL). Another paradigm is to estimate first-order or second-order information only using inference. Besides MeZO estimating first-order information by two forward processes, HiZOO leverages three forward processes to estimate second-order information considering heterogeneous curvatures across different parameter dimensions [60].

3 Pseudo-Zeroth-order algorithm framework

3.1 Zeroth-order algorithms

Zeroth-order algorithms are a class of optimization algorithms that do not require the computation of gradients. Instead, they rely on noisy function value oracles to update the parameters. This makes them useful for problems where the computation of gradients is expensive or infeasible. Typically, zeroth-order methods use noisy function values to generate gradient estimators. Suppose the objective function is $f : \mathbb{R}_d \rightarrow \mathbb{R}$. Two most common gradient estimators are RGE [36] and CGE [1]:

$$\text{RGE} : \hat{\nabla}_\mu f(\mathbf{x}) = \frac{1}{q} \sum_{i=1}^q \frac{f(\mathbf{x} + \mu \boldsymbol{\xi}_i) - f(\mathbf{x})}{\mu} \boldsymbol{\xi}_i, \quad \text{CGE} : \hat{\nabla}_\mu f(\mathbf{x}) = \frac{1}{d} \sum_{i=1}^d \frac{f(\mathbf{x} + \mu \mathbf{e}_i) - f(\mathbf{x})}{\mu} \mathbf{e}_i, \quad (2)$$

where $\boldsymbol{\xi}_i \sim N(\mathbf{0}, \mathbf{I})$ and $\{\mathbf{e}_i\}$ is a set of standard basis vector. When $\mu \rightarrow 0$, CGE become the full gradient, and RGE tends to

$$\hat{\nabla} f(\mathbf{x}) = \frac{1}{q} \sum_{i=1}^q \langle \nabla f(\mathbf{x}), \boldsymbol{\xi}_i \rangle \boldsymbol{\xi}_i. \quad (3)$$

This is an unbiased estimation of $\nabla f(\mathbf{x})$, with variance $\Theta\left(\frac{d}{q}\right)$. Without further assumptions on objective functions, optimizing with RGE or CGE needs d times more zeroth-order oracles than optimizing with first-order methods using gradient oracles, severely limiting their applicability in high-dimensional scenarios. However, if the objective function has low effective dimension or sparsity structures, zeroth-order methods can achieve a faster convergence rate [56, 60].

3.2 Pseudo-zeroth-order algorithms

In this paper, we study the composite optimization problem:

$$\min_{\mathbf{x} \in \mathcal{X}} \mathcal{F}(\mathbf{x}) := \mathbb{E}_{\mathbf{z}} g(h(\mathbf{x}; \mathbf{z})), \quad \text{where } \mathcal{X} \subseteq \mathbb{R}^{d_p}, \quad (4)$$

where $h : \mathcal{X} \rightarrow \mathcal{H} \subseteq \mathbb{R}^{d_{\text{out}}}$ defines the representation mapping and $g : \mathcal{H} \rightarrow \mathbb{R}_+$ is a loss function. In machine learning, the representation mapping $h : \mathbb{R}^{d_p} \rightarrow \mathbb{R}^{d_{\text{out}}}$ is typically parameterized as a neural network $h(\mathbf{x}; \mathbf{z})$, where \mathbf{x} denotes the learnable parameters and $\mathbf{z} \sim \mathcal{D}$ denotes the stochastic input data. The exact stochastic gradient admits the theoretical decomposition:

$$\nabla_{\mathbf{x}} \mathcal{F}(\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim \mathcal{D}} \left[\underbrace{\mathbf{J}_h^\top(\mathbf{x}; \mathbf{z})}_{\substack{\text{Stochastic Jacobian} \\ d_{\text{out}} \times d_p}} \cdot \underbrace{\nabla_{\mathbf{h}} g(h(\mathbf{x}; \mathbf{z}))}_{\substack{\text{Upstream gradient} \\ d_{\text{out}} \rightarrow \mathbb{R}}} \right]. \quad (5)$$

The implementation of stochastic first-order methods requires efficient computation of $\nabla_{\mathbf{x}} \mathcal{F}(\mathbf{x})$ via backpropagation through the compositional structure $g \circ h$. While the outer loss g admits tractable gradient computation ($\nabla_{\mathbf{h}} g$ is typically closed-form), the inner mapping's Jacobian $\mathbf{J}_h(\mathbf{x}; \mathbf{z})$ becomes computationally intractable for deep nonlinear parameterizations.

Unlike standard zeroth-order (ZO) methods that solely utilize function evaluations of $g \circ h$, our key insight stems from the asymmetric differentiability inherent in composite optimization: While acquiring exact gradients through the inner mapping $h(\mathbf{x}; \mathbf{z})$ remains computationally prohibitive due to computational constraints or non-differentiable operators, the gradient of the outer function g can be explicitly or efficiently computed.

Our method exploits composite structure's asymmetric differentiability:

- **Outer gradient:** Closed-form $\mathbf{e} = \nabla_{\mathbf{o}} g(\mathbf{o})$ for $\mathbf{o} = h(\mathbf{x}; \mathbf{z})$;
- **Inner estimation:** Zeroth-order approximation for h 's Jacobian.

The PZO gradient estimator combines both components:

$$\nabla_{\mu}^{\text{PZO}} \mathcal{F}(\mathbf{x}; \mathbf{z}) = \boldsymbol{\xi} \left(\frac{h(\mathbf{x} + \mu \boldsymbol{\xi}; \mathbf{z}) - h(\mathbf{x}; \mathbf{z})}{\mu} \right)^\top \mathbf{e}, \quad (6)$$

where $\boldsymbol{\xi} \sim N(\mathbf{0}, \mathbf{I})$. As $\mu \rightarrow 0$, this converges to $\nabla^{\text{PZO}} \mathcal{F}(\mathbf{x}; \mathbf{z}) = (\boldsymbol{\xi}^\top \mathbf{J}_h^\top \mathbf{e}) \boldsymbol{\xi} = \boldsymbol{\xi} \boldsymbol{\xi}^\top \mathbf{J}_h^\top \mathbf{e}$, which is an unbiased estimation of $\nabla \mathcal{F}(\mathbf{x})$. To reduce the variance introduced by the random vector $\boldsymbol{\xi}$, we propose a momentum-accelerated variant through exponential smoothing:

$$\mathbf{A} = (1 - \lambda) \left(\frac{h(\mathbf{x} + \mu \boldsymbol{\xi}; \mathbf{z}) - h(\mathbf{x}; \mathbf{z})}{\mu} \right)^\top \boldsymbol{\xi}^\top + \lambda \mathbf{A}, \quad (7)$$

$$\hat{\nabla}_{\mu}^{\text{PZO}} \mathcal{F}(\mathbf{x}; \mathbf{z}) = \mathbf{A}^\top \mathbf{e}. \quad (8)$$

The full algorithm is shown in Algorithm 1.

Remark 1. The gradient estimator of PseuZO $\hat{\nabla}_{\mu}^{\text{PZO}} \mathcal{F}(\mathbf{x}; \mathbf{z}) = \mathbf{A}^\top \mathbf{e}$ theoretically outperforms MeZO and its momentum variant in terms of both bias and variance, as detailed in Appendix D. PseuZO has smaller variance compared to MeZO and MeZO with momentum, and MeZO has an extra bias term caused by the curvature of g . These theoretical advantages demonstrate the effectiveness of the exact outer gradient and exponential smoothing of Jacobian estimations.

Algorithm 1 Matrix-based PseuZO Algorithm

Require: Momentum factor $\beta \in (0, 1)$, smoothing coefficient $\mu > 0$, max iterations T , initial point \mathbf{x}_0

Ensure: Gradient estimate $\hat{\nabla}_\mu^{\text{PZO}} \mathcal{F}(\mathbf{x}; \mathbf{z})$

- 1: Initialize momentum buffer $\mathbf{A}_{-1} \leftarrow \mathbf{0} \in \mathbb{R}^{d_{\text{out}} \times d_p}$
 - 2: **for** $t = 0$ **to** T **do**
 - 3: Sample random vector $\boldsymbol{\xi}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - 4: Compute forward difference $\Delta \mathbf{o}_t \leftarrow \frac{h(\mathbf{x}_t + \mu \boldsymbol{\xi}_t; \mathbf{z}_t) - h(\mathbf{x}_t; \mathbf{z}_t)}{\mu}$
 - 5: Receive noisy Jacobian estimator $\mathbf{B}_t = \Delta \mathbf{o}_t \boldsymbol{\xi}_t^\top$
 - 6: Momentum update:
 - 7: $\mathbf{A}_t \leftarrow (1 - \lambda_t) \mathbf{B}_t + \lambda_t \mathbf{A}_{t-1}$ ▷ Exponential moving average
 - 8: Compute outer gradient $\mathbf{e}_t \leftarrow \nabla_{\mathbf{o}} g(h(\mathbf{x}; \mathbf{z}_t))$
 - 9: Gradient projection: $\hat{\nabla}_\mu^{\text{PZO}} \mathcal{F}(\mathbf{x}_t; \mathbf{z}_t) \leftarrow \mathbf{A}_t^\top \mathbf{e}_t$
 - 10: Update \mathbf{x} : $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \eta \hat{\nabla}_\mu^{\text{PZO}} \mathcal{F}(\mathbf{x}; \mathbf{z}_t)$
 - 11: **end for**
-

3.3 Convergence of PseuZO algorithm

In this subsection, we deviate from the exact realization of stochastic Jacobian estimators, and assume that the PseuZO algorithm receives a noisy Jacobian estimator in each step. Under this more general setting, we prove the convergence of PseuZO method. We first list some assumptions which are necessary for our analysis and has been widely adopted in research works on optimization:

Assumption 2. \mathcal{F} has continuous Hessian matrices $\mathbf{H}(\mathbf{x})$, and satisfy the following equations:

$$\|\mathbf{H}(\mathbf{x})\|_{\text{op}} \leq L, \quad \text{tr}(\mathbf{H}(\mathbf{x})) \leq \alpha_1 L, \quad (9)$$

where α_1 is the effective dimension of \mathcal{F} . In the worst case, $\alpha_1 = d_p$.

Assumption 3. Denote $\hat{\nabla} \mathcal{F}(\mathbf{x}_t) = \nabla_{\mathbf{x}} g(h(\mathbf{x}_t; \mathbf{z}_t))$. The randomness of the Jacobian estimator \mathbf{B}_t can be decoupled into the following parts:

$$\mathbf{B}_t = (\mathbf{J}_t + \mathbf{D}_t) \mathbf{N}_t + \mathbf{M}_t, \quad (10)$$

where \mathbf{J}_t is the true Jacobian matrix of h at \mathbf{x}_t , and:

- \mathbf{N}_t represents the randomness introduced by the PZO Jacobian estimator:

$$\mathbb{E} \mathbf{N}_t = \mathbf{I}, \quad \mathbb{E} \|\mathbf{N}_t^\top \mathbf{a}\|_{\mathbf{M}}^2 \leq 3 \text{tr}(\mathbf{M}) \cdot \|\mathbf{a}\|^2, \quad (11)$$

where \mathbf{a} is an arbitrary vector.

- \mathbf{D}_t represents the randomness of data:

$$(\mathbf{J}_t + \mathbf{D}_t)^\top \mathbf{e}_t = \hat{\nabla} \mathcal{F}(\mathbf{x}_t), \quad \mathbb{E} \mathbf{D}_t = 0, \quad \mathbb{E} \|\mathbf{D}_t^\top \mathbf{e}_t\|^2 \leq \sigma_2^2; \quad (12)$$

- \mathbf{M}_t represents the noise introduced by the two-point estimation of the function value, controlled by μ :

$$\mathbb{E} \|\mathbf{M}_t^\top \mathbf{e}_t\|^2 \leq \frac{\mu^2 L_h^2 L_g^2}{2} (d_p + 6)^3. \quad (13)$$

According to the analysis on the two-point gradient estimators in [36], $\mathbf{B}_t = \left(\frac{h(\mathbf{x} + \mu \boldsymbol{\xi}; \mathbf{z}) - h(\mathbf{x}; \mathbf{z})}{\mu} \right) \boldsymbol{\xi}^\top$ satisfies Assumption 3, where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Now, we propose the informal version of our convergence theorem in Theorem 4. For the formal version and the proof of Theorem 4, please refer to the Appendix C.

Theorem 4 (Informal). Under Assumption 2 and 3, if μ , ϵ and β_t satisfy certain conditions, Algorithm 1 finds an ϵ -stationary point with $O\left(\max\{\alpha_1 L \epsilon^{-2}, \alpha_1 L \sigma_2^2 \epsilon^{-4}\}\right)$ function value computations.

Remark 5. The convergence rate of Algorithm 1 is α_1 times slower than the standard SGD algorithm, where α_1 is not explicitly dependent on the dimension of the problem.

3.4 Sliding window-based PseuZO algorithm

As it is memory inefficient to store such a large-scale tensor \mathbf{A}_t (outer product of Gaussian noise and an output tensor) introduced in PseuZO, we propose several techniques to reduce the memory cost and obtain Sliding Window-based PseuZO shown in Algorithm 2.

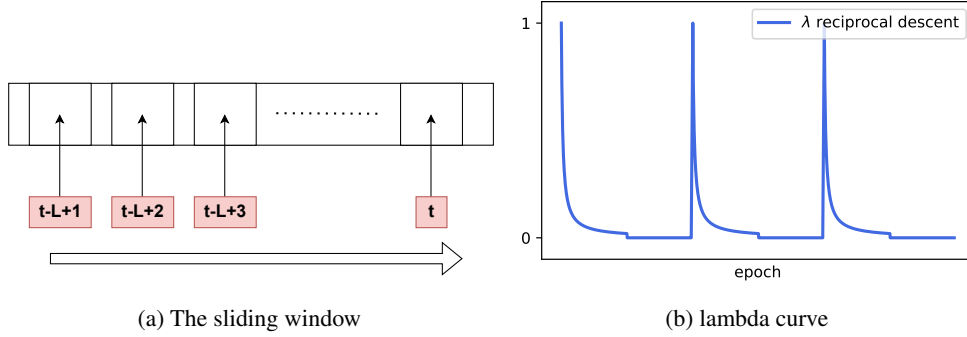


Figure 1: (a) is a schematic diagram of the sliding window with length L . We can obtain the corresponding coefficients of all sliding window units by expanding the iterative L times. (b) is a λ variation curve with three cycles. Each cycle means a restart operation and when λ is close to zero, we set $\lambda = 0$ to reduce time consumption.

Sliding window technique. To tackle the memory overhead of PseuZO, we use a sliding window to store random seeds and output tensors in the last few steps. When the sliding window shown in Figure 1a is determined, the corresponding coefficient for each unit in the sliding window can be obtained by expanding the EMA formula. On the one hand, we can resample the same Gaussian noise ξ with the same seed so there is no need to store these random vectors [34]. On the other hand, as $0 < \lambda < 1$, especially when λ is smaller, the weight for older information is close to zero in the EMA step of PseuZO method. Therefore, we truncate outdated information with the sliding window technique.

Changing the storage target. However, storing several output tensors with a shape of the vocabulary size is still unacceptable, as the vocabulary size is generally large among LLMs. So we choose to store the last hidden state with a much smaller size and trace its path to the final loss to obtain its gradient without excessive memory overhead. For OPT1.3B, the last hidden state (i.e. the input tensor for **lm_head**) size is 2048 while the vocabulary size is larger than 50000 [58]. Thus we can further reduce memory overhead caused by the sliding window.

Periodic dynamic changing of λ . To meet the convergence requirements, λ needs to gradually decrease to zero. We take advantage of a restart operation to further boost convergence and thus λ is designed in a periodic changing manner as shown in Figure 1b. When λ approaches zero, PseuZO gradually degenerates to ZO method and thus we directly transform to MeZO instead. If we consistently use PseuZO throughout the entire process, the time cost is roughly $2\times$ that of MeZO. However, with appropriate design for λ , the time cost can be reduced to almost the same as MeZO.

4 Experiments

In this section, we evaluate Sliding Window-based PseuZO on a variety of typical fine-tuning tasks by comparing performance against MeZO [34], MeZO-SVRG [19], HiZOO-L [60] and memory overhead against MeZO-SVRG as well as FO-SGD. Experimental results show that: 1) The peak memory usage of PseuZO is significantly smaller than MeZO-SVRG and FO-SGD; 2) Through the sliding window technique, with a fixed memory overhead that is independent of the model size compared to MeZO, PseuZO has much better performance than MeZO, MeZO-SVRG and HiZOO-L. 3) PseuZO is also compatible with PEFT techniques like LoRA [25] and prefix-tuning [31].

Setup. We implement PseuZO, MeZO-SVRG and HiZOO-L in the MeZO framework with appropriate adjustment for fair comparison. We conduct comprehensive experiments in various tasks on large auto-regressive language models like opt-1.3B [58] and the same prompt design as MeZO is utilized which is effective and fair for comparison for various datasets including GLUE [52] and SuperGLUE

Algorithm 2 Sliding Window-based PseuZO Algorithm

Require: Momentum factor formula $\lambda(t) \in (0, 1)$, smoothing coefficient $\epsilon > 0$, max iterations T , initial point θ_0 , sliding window length L , coefficients $\{u_k\}_{k=1}^L$

Ensure: Gradient estimate $\hat{\nabla}_\epsilon^{\text{PZO}} \mathcal{F}(\mathbf{x}; \mathbf{z})$

```

1: Initialize the sliding window as a deque  $D(\text{maxlen}=L)$ 
2: for  $t = 0$  to  $T$  do
3:   Update  $\lambda \leftarrow \lambda(t)$ 
4:   Compute coefficients  $u_k \leftarrow \lambda^{k-1}(1 - \lambda)$ 
5:   Sample random seed  $s_t$  and corresponding vector  $\xi_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}; s_t)$ 
6:   Compute forward difference  $\Delta \mathbf{o}_t \leftarrow \frac{h(\mathbf{x}_t + \epsilon \xi_t; \mathbf{z}_t) - h(\mathbf{x}_t; \mathbf{z}_t)}{\epsilon}$ 
7:   Compute outer gradient  $\mathbf{e}_t \leftarrow \nabla_{\mathbf{o}} g(h(\mathbf{x}_t; \mathbf{z}_t))$ 
8:   Update sliding window  $D.\text{append}(s_t, \Delta \mathbf{o}_t)$ 
9:   Gradient projection initialization:  $\hat{\nabla}_\epsilon^{\text{PZO}} \mathcal{F}(\mathbf{x}_t; \mathbf{z}_t) \leftarrow \mathbf{0}$ 
10:  for  $k = 1$  to  $L$  do ▷ Iterate sliding window
11:     $s, \Delta \mathbf{o} \leftarrow D_k$ 
12:    Resample  $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I}; s)$ 
13:    Accumulate  $\hat{\nabla}_\epsilon^{\text{PZO}} \mathcal{F}(\mathbf{x}_t; \mathbf{z}_t) \leftarrow \hat{\nabla}_\epsilon^{\text{PZO}} \mathcal{F}(\mathbf{x}_t; \mathbf{z}_t) + u_k \langle \Delta \mathbf{o}, \mathbf{e}_t \rangle \xi$ 
14:  end for
15:  Update  $\theta$ :  $\theta_{t+1} \leftarrow \theta_t - \eta \hat{\nabla}_\epsilon^{\text{PZO}} \mathcal{F}(\theta_t; \mathbf{x}_t)$ 
16: end for

```

Task Task type	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	DROP
	classification							multiple choice		generation
Zero-shot	53.5	53.0	39.3	45.7	43.3	51.5	45.4	75.0	70.5	11.2
ICL	80.0	53.0	46.4	58.7	47.1	51.1	46.2	69.0	71.0	20.4
MeZO-SVRG	61.5	55.5	74.0	60.3	52.0	50.0	53.0	54.0	50.1	0.0
MeZO (10K steps)	82.4	54.3	76.0	60.7	51.0	50.9	54.9	74.0	57.6	20.3
MeZO (20K steps)	88.4	58.8	76.0	63.8	53.0	51.3	53.9	73.0	58.9	20.3
HiZOO-L	88.1	54.9	69.0	64.8	51.2	58.0	58.2	73.0	58.8	23.3
PseuZO (10K steps)	91.2	58.0	77.0	64.3	58.0	54.5	54.7	78.0	60.0	23.5
PseuZO (20K step)	90.7	63.3	75.0	67.0	57.0	59.7	60.6	76.0	60.9	24.5
FO-SGD	92.4	67.8	94.0	60.8	52.6	47.4	53.8	76.0	57.2	26.0

Table 1: Experiments on OPT-1.3B with 1024 training samples and 512 evaluation samples. When training, for WSC, CB and COPA, they have much less total samples and thus we set aside 100 evaluation samples and use the rest for training. The **bold** number represents the highest evaluation performance excluding FO-SGD.

[51] benchmarks. We run all experiments for **10K** steps and evaluate performance of the model every **2K** steps for HiZOO-L and MeZO-SVRG. In order to ensure that MeZO and PseuZO are sufficiently convergent, we run PseuZO and MeZO for **10K** and **20K** steps, respectively. We choose $K = 16$ as the batch size and randomly select 1024 samples for training and 512 samples for evaluation. All experiments are run on a single Nvidia A800 40GiB GPU.

Task	SST-2	RTE	CB	BoolQ	WSC	WIC	COPA
MeZO	84.4	58.0	76.0	64.0	54.0	52.5	88.0
PseuZO (ours)	91.8	58.4	77.0	68.9	58.0	55.3	90.0

Table 2: Experiments on OPT-6.7B for PseuZO versus MeZO. The **bold** number represents the better evaluation performance.

Task	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	DROP
Task type	classification							multiple choice	generation	
MeZO+LoRA	90.8	59.1	76.0	64.6	50.0	52.7	54.5	81.0	59.5	22.9
PseuZO+LoRA	91.2	58.8	79.0	65.8	51.0	51.8	54.3	83.0	60.2	25.7
MeZO+prefix	71.0	51.7	45.0	57.6	54.0	50.6	50.8	75.0	57.4	15.5
PseuZO+prefix	80.7	53.1	71.0	61.7	49.0	51.0	52.7	75.0	57.8	21.2

Table 3: Experiments on OPT-1.3B for comparison between MeZO+PEFT and PseuZO+PEFT. PEFT is either LoRA or prefix fine-tuning.

4.1 Auto regressive model performance

PseuZO performs much better than MeZO, MeZO-SVRG and HiZOO-L in the classification, multiple choice and generation tasks shown in Table 1. As MeZO-SVRG needs to traverse all samples to obtain the full-batch gradient [19], large numbers of iterations are required or it will have an expensive memory overhead. However, MeZO-SVRG still performs worse than PseuZO, especially for multiple choice and generation tasks with a small batch size and a large full batch size. For HiZOO-L [60], it not only requires three forward propagations, but also requires low-rank processing of matrix parameters, which increases the time overhead. In fact, running HiZOO-L and MeZO-SVRG for only 10K steps takes much longer than running PseuZO for 20K steps. As shown in Appendix D, PseuZO takes advantage of the small bias and variance to achieve fast convergence, further reducing the gap with FO-SGD and even outperforms FO-SGD in many tasks.

PseuZO is also compatible with other memory efficient techniques, like LoRA and prefix tuning, and the corresponding results are shown in Table 3. As both LoRA and prefix tuning have much fewer parameters to optimize (0.1% and 0.01% of the original number of parameters respectively for OPT1.3B) so that the number of convergence steps required will be lower, the performance gap between MeZO+PEFT and PseuZO+PEFT is much smaller than that of MeZO and PseuZO.

4.2 Memory usage and time computation

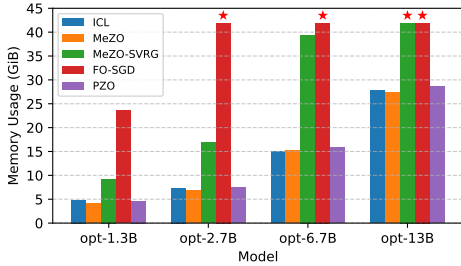


Figure 2: Memory overhead for different models under various ZO algorithms and FO-SGD. ★ means out of memory.

t_{PseuZO}	OPT1.3B	OPT2.7B	OPT6.7B
10	9300	9600	9900
20	8700	9400	9700
50	6800	8400	9300
100	3600	7000	8700
150	400	5100	8000

Table 4: t_{PseuZO} and its corresponding number of the total steps to guarantee the time that does not exceed 10K execution time for MeZO.

PseuZO scales up to almost the same model size as MeZO can scale up to as PseuZO only introduces a small dimension-independent memory overhead. Memory usage for different scale models is illustrated in Figure 2 which verifies our conclusion that compared to MeZO, excessive memory overhead of PseuZO is independent of the model size. Additionally, we also compare memory overhead for the different batch size and max length. As shown in Table 5, the memory requirement of PseuZO is more sensitive to the batch size and max length due to the sliding window. However, as model size has a greater impact on memory, PseuZO still needs less memory than MeZO-SVRG (MeZO-SVRG needs to store full parameters during training [19]).

The computation time required for each epoch of PseuZO is approximately twice that of each epoch of MeZO for OPT-1.3B but the gap for large scale models will gradually decrease. We fix the total steps of MeZO as 10K and obtain the corresponding total steps of PseuZO for different

Method	Memory Usage in GiB for OPT-1.3B					Memory Usage in GiB for OPT-6.7B				
	Fixed context length (cl=128)			Fixed batch size (bs=16)		Fixed context length (cl=128)			Fixed batch size (bs=16)	
	bs=16	bs=32	bs=64	cl=256	cl=512	bs=16	bs=32	bs=64	cl=256	cl=512
FO-SGD	30.30					OOM				
MeZO	5.35	7.58	11.87	9.06	17.03	17.09	20.35	27.51	21.28	27.01
MeZO-SVRG	10.55	12.76	17.55	12.46	19.67	OOM				
PseuZO	7.61	10.58	17.85	11.69	19.36	18.08	23.22	33.52	25.16	32.74

Table 5: Memory usage (GiB) comparison on BoolQ for different ZO methods with OPT1.3B and OPT-6.7B showing that PseuZO can scale up to larger models as MeZO.

Dataset	SPSA	ZO _{sp}	PseuZO	PseuZO (w/LL)	BP
MNIST	86.4 ± 0.15	87.8 ± 0.09	98.7 ± 0.02	/	98.5 ± 0.02
CIFAR-10	41.3 ± 0.74	42.6 ± 0.69	82.5 ± 0.15	88.7 ± 0.13	89.9 ± 0.06
CIFAR-100	5.39 ± 0.69	7.61 ± 0.73	61.4 ± 0.14	68.5 ± 0.13	71.9 ± 0.09

Table 7: Training from scratch on typical computer vision classification datasets for various feedback methods. We do not use local loss for MNIST as there are only two hidden layers.

t_{PseuZO} under the same computation time where t_{PseuZO} is the number of total epochs using PseuZO and results are shown in Table 4.

4.3 Ablation study

As introduced before, our sliding window has four key factors denoted as 1) L : sliding window length; 2) t_{PseuZO} : number of epochs to execute PseuZO for each cycle; 3) R : number of cycles; 4) $\lambda(t)$: the formula for λ to descend with respect to epoch t . We perform ablation studies on SST2 to explore their individual impact on precision and the results are shown in Figure 3 and Table 6. If we keep λ a small constant like $\lambda_{\min} = 0.1$, the accuracy is close to that of MeZO. In fact, when $\lambda \rightarrow 0$, PseuZO gradually degenerates to MeZO without momentum. More ablation experiments, including MeZO with momentum and PseuZO without momentum, can be found in Appendix A. We empirically found that performance is largely insensitive to the sliding window parameters. Since there is no need to set L , t_{PseuZO} and R too large, smaller values are chosen to balance low memory and time overhead.

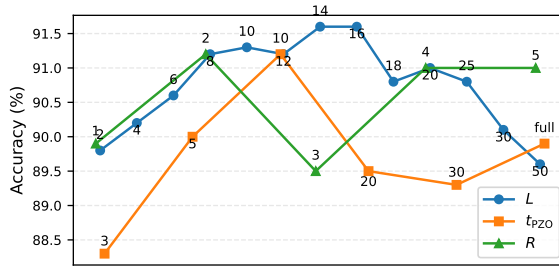


Figure 3: Ablation studies for L , t_{PseuZO} and R . "full" means using PseuZO for all epochs.

	$\lambda(t)$	Accuracy(%)
Constant1	$\lambda(t) = \lambda_{\max}$	83.9
Constant2	$\lambda(t) = \lambda_{\min}$	82.7
Linear	$\lambda(t) = \lambda_{\max}(1 - \frac{t}{t_{\text{PseuZO}}})$	90.3
Reciprocal	$\lambda(t) = \lambda_{\max} \frac{1}{1+0.5t}$	91.2

Table 6: Ablation study for different descent formula $\lambda(t)$ when $t < t_{\text{PseuZO}}$.

4.4 Training from scratch

During ZO optimization, the variance introduced by the large number of parameters is difficult to control and thus an appropriate prompt design is significant to guide generation [34, 19] for instruction fine-tuning. Furthermore, it remains a problem for ZO algorithms like SPSA to train from scratch even for small datasets. However, with a simple reformulation named Node Perturbation [32], PseuZO shows great performance on these tasks. The reformulation details and further explanation

can be found in Appendix B. As shown in Table 7, PseuZO in Node Perturbation manner outperforms SPSA and other signal feedback methods and with incorporation of local learning [26], PseuZO can even perform as well as BP. It demonstrates the potential of PseuZO to be used for more difficult but significant settings.

5 Conclusion

In this paper, we propose a new algorithm framework PesuZO and apply it to various instruction fine-tuning tasks for LLMs. According to our theoretical analyses, PseuZO finds an ϵ -stationary point in $O(\max\{\alpha_1 L \epsilon^{-2}, \alpha_1 L \sigma_2^2 \epsilon^{-4}\})$ function evaluations. Experimental results demonstrate that PseuZO outperforms MeZO and MeZO-SVRG among various tasks. We propose sliding window technique, making the memory overhead independent of the model size. As a limitation, PseuZO needs to store the sliding window and the extra memory is sensitive to the batch size and max length. A possible fix to this problem is to add an low-dimension auxiliary layer as the new storing target. Though this method can further reduce memory overhead, it changes the model structure which might reduce its representation capability.

Acknowledgments and Disclosure of Funding

Z. Lin was supported by National Key R&D Program of China (2022ZD0160300), the NSF China (No. 62276004) and the State Key Laboratory of General Artificial Intelligence.

References

- [1] Zeyuan Allen-Zhu, Zheng Qu, Peter Richtárik, and Yang Yuan. Even Faster Accelerated Coordinate Descent Using Non-Uniform Sampling, May 2016. arXiv:1512.09103 [cs, math, stat].
- [2] Stephen H Bach, Victor Sanh, Zheng-Xin Yong, Albert Webson, Colin Raffel, Nihal V Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, et al. Promptsource: An integrated development environment and repository for natural language prompts. *arXiv preprint arXiv:2202.01279*, 2022.
- [3] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1, 2009.
- [4] Andrew Brock, Soham De, and Samuel L Smith. Characterizing signal propagation to close the performance gap in unnormalized resnets. *arXiv preprint arXiv:2101.08692*, 2021.
- [5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.
- [6] HanQin Cai, Yuchen Lou, Daniel McKenzie, and Wotao Yin. A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization. In *International Conference on Machine Learning*, pages 1193–1203. PMLR, 2021.
- [7] Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In *Artificial Intelligence and Statistics*, pages 10–19. PMLR, 2014.
- [8] Yekun Chai, Shuohuan Wang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. Clip-tuning: Towards derivative-free prompt learning with a mixture of rewards. *arXiv preprint arXiv:2210.12050*, 2022.
- [9] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26, 2017.

- [10] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.
- [11] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer, 2005.
- [12] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.
- [13] Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124, 2019.
- [14] Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. Rlprompt: Optimizing discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.
- [15] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.
- [16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.
- [17] 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.
- [18] Jianwei Feng and Dong Huang. Optimal gradient checkpoint search for arbitrary computation graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11433–11442, 2021.
- [19] Tanmay Gautam, Youngsuk Park, Hao Zhou, Parameswaran Raman, and Wooseok Ha. Variance-reduced zeroth-order methods for fine-tuning language models. *arXiv preprint arXiv:2404.08080*, 2024.
- [20] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.
- [21] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9, 2007.
- [22] Jean-Bastien Grill, Michal Valko, and Rémi Munos. Black-box optimization of noisy functions with unknown smoothness. *Advances in Neural Information Processing Systems*, 28, 2015.
- [23] Bin Gu, Guodong Liu, Yanfu Zhang, Xiang Geng, and Heng Huang. Optimizing large-scale hyperparameters via automated learning algorithm. *arXiv preprint arXiv:2102.09026*, 2021.
- [24] Wentao Guo, Jikai Long, Yimeng Zeng, Zirui Liu, Xinyu Yang, Yide Ran, Jacob R Gardner, Osbert Bastani, Christopher De Sa, Xiaodong Yu, et al. Zeroth-order fine-tuning of llms with extreme sparsity. *arXiv preprint arXiv:2406.02913*, 2024.
- [25] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- [26] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14:424, 2020.

- [27] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, 2018.
- [28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [29] Hector J Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. *KR*, 2012:13th, 2012.
- [30] Jia Li, Cong Fang, and Zhouchen Lin. Lifted proximal operator machines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4181–4188, 2019.
- [31] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.
- [32] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276, 2016.
- [33] Zhouchen Lin, Risheng Liu, and Zhixun Su. Linearized alternating direction method with adaptive penalty for low-rank representation. *Advances in Neural Information Processing Systems*, 24, 2011.
- [34] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.
- [35] Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072, 2024.
- [36] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.
- [37] Anthony Nguyen and Krishnakumar Balasubramanian. Stochastic zeroth-order functional constrained optimization: Oracle complexity and applications. *INFORMS Journal on Optimization*, 5(3):256–272, 2023.
- [38] Mohammad Taher Pilehvar and Jose Camacho-Collados. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*, 2018.
- [39] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.
- [40] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–14, 2021.
- [41] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. {Zero-offload}: Democratizing {billion-scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564, 2021.
- [42] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.
- [43] 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*, pages 90–95, 2011.

- [44] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.
- [45] James C Spall. A stochastic approximation algorithm for large-dimensional systems in the kiefer-wolfowitz setting. In *Proceedings of the 27th IEEE Conference on Decision and Control*, pages 1544–1548. IEEE, 1988.
- [46] Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *International Conference on Machine Learning*, pages 3299–3308. PMLR, 2017.
- [47] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In *International conference on machine learning*, pages 2722–2731. PMLR, 2016.
- [48] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [49] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 742–749, 2019.
- [50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.
- [51] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in Neural Information Processing Systems*, 32, 2019.
- [52] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.
- [53] Bingzhen Wei, Xu Sun, Xuancheng Ren, and Jingjing Xu. Minimal effort back propagation for convolutional neural networks. *arXiv preprint arXiv:1709.05804*, 2017.
- [54] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online pseudo-zeroth-order training of neuromorphic spiking neural networks. *arXiv preprint arXiv:2407.12516*, 2024.
- [55] Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. Corda: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:71768–71791, 2024.
- [56] Pengyun Yue, Long Yang, Cong Fang, and Zhouchen Lin. Zeroth-order optimization with weak dimension dependency. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 4429–4472. PMLR, 2023.
- [57] 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.
- [58] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

- [59] Yimeng Zhang, Yuguang Yao, Jinghan Jia, Jinfeng Yi, Mingyi Hong, Shiyu Chang, and Sijia Liu. How to robustify black-box ml models? a zeroth-order optimization perspective. *arXiv preprint arXiv:2203.14195*, 2022.
- [60] Yanjun Zhao, Sizhe Dang, Haishan Ye, Guang Dai, Yi Qian, and Ivor W Tsang. Second-order fine-tuning without pain for llms: A hessian informed zeroth-order optimizer. *arXiv preprint arXiv:2402.15173*, 2024.

NeurIPS Paper Checklist

1. Claims

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

Answer: [\[Yes\]](#)

Justification: Our Abstract and Introduction section do not claim any contribution out of scope. All the contributions mentioned are supported in the later sections, see Section 1

Guidelines:

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

2. Limitations

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

Answer: [\[Yes\]](#)

Justification: We discuss the limitations of this work in Section 5

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

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

Answer: [\[Yes\]](#)

Justification: We provide the full set of assumptions in Section 3 and a complete (and correct) proof can be found in the Appendix.

Guidelines:

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

4. Experimental result reproducibility

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

Answer: [\[Yes\]](#)

Justification: The complete implementation details to reproduce our experimental results are described in the Appendix A, B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

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

Answer: [Yes]

Justification: Our code is available at <https://github.com/YangBigMn/PseuZO>.

Guidelines:

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

6. Experimental setting/details

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

Answer: [Yes]

Justification: We specify all the training and test details in the Appendix A,B.

Guidelines:

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

7. Experiment statistical significance

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

Answer: [Yes]

Justification: We report the mean and standard deviation values in Table 7 by running each result for 5 times with different seeds. For the other results, repeated experiments of fine-tuning for various tasks will consume too much time and computing resource.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

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

Answer: [Yes]

Justification: We provide the GPU type in the Appendix A.

Guidelines:

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

9. Code of ethics

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

Answer: [Yes]

Justification: We obey the NeurIPS Code of Ethics in every respect.

Guidelines:

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

10. Broader impacts

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

Answer: [NA]

Justification: Our study is about a new Zeroth-order algorithm for neural networks. No societal impact is concerned by our work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

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

Answer: [NA]

Justification: Our work poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

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

Answer: [Yes]

Justification: We have cited all the papers whose datasets are used in our experiment. Details can be found in the Appendix A,B.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

Answer: [NA]

Justification: We do not release new assets.

Guidelines:

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

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

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

Answer: [NA]

Justification: No experiment involves crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

Answer: [NA]

Justification: No experiment involves crowdsourcing or research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [No]

Justification: We only use LLMs for writing, editing or formatting purposes.

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

A Auto Regressive Model Fine-tuning

A.1 Datasets

We run experiments on various types of tasks including classification, multiple choice and generation, in GLUE [52] and SuperGLUE [51]. They are SST-2 [44], RTE [11, 21, 3], CB [13], BoolQ [10], WiC [38], WSC [29], MultiRC [27], COPA [43], ReCoRD [57] and also a generation task DROP [17]. We choose batch size $BS = 16$ and for OPT1.3B we choose the max sequence length $MAX_LEN = 2048$ and for OPT6.7B, $MAX_LEN = 512$. For each dataset, we randomly choose 1024 training samples and 512 evaluation samples. As the total samples for WSC, CB and COPA are insufficient, we set aside 100 evaluation samples and use the rest for training. As for the prompt design, we utilize the effective template in MeZO [34] which follows GPT-3 [5], PromptSource [2] with minor changes. We run all experiments with one Nvidia A800 40GiB GPU.

A.2 Hyperparameters

We run MeZO-SVRG and HiZOO-L for **10K** steps while run MeZO and PseuZO for **10K** steps and **20K** steps, respectively. For PseuZO, as the ablation study 4.3 shows, it is not sensitive to the hyperparameters for the sliding window. Therefore, for OPT1.3B, we choose sliding window length $L = 16$, $t_{\text{PseuZO}} = 10$, using reciprocal descent λ and running for 2 cycles. In this way, under the same computation time, we can run PseuZO for **9.3K** steps which implies a similar time overhead between MeZO and PseuZO. As the inference time for OPT-6.7B is much higher than that of OPT-1.3B, we can adjust $t_{\text{PseuZO}} = 50$. For FO-SGD, we only run 5 epochs and evaluate each epoch.

For in-context learning, we add 32 examples for the model to learn. As the choice of ϵ does not appear to significantly affect performance [34], we search the grid for the best learning rate η . For FO-SGD, we utilize a linear scheduled learning rates but use the constant learning rate for ZO algorithms. Considering the fewer parameters for LoRA and prefix-tuning, we appropriately adjust to a larger learning rate following MeZO and MeZO-SVRG. The final hyperparameter grid is shown in Table 10.

A.3 More Results for OPT-6.7B

Table 8 shows the complete results for the larger OPT-6.7B. For FO-SGD, the memory footprint is much larger than that of both MeZO and PseuZO, especially for the generation task, which requires a longer context window and thus for OPT-6.7B, we just compare PseuZO against MeZO and ICL to show PseuZO’s superiority.

Figure 4a, 4b show the loss curve for OPT-1.3B and OPT-6.7B on SST-2 which further verifies our claim that PseuZO has a faster descent speed than MeZO especially in the first few epochs.

Task	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	DROP
Task type	classification							_multiple choice _		generation
Zero-shot	61.2	54.9	53.6	59.4	37.5	51.2	44.5	82.0	75.9	17.5
ICL	84.1	61.4	57.1	63.0	46.1	51.9	47.8	81.0	77.1	26.7
MeZO	84.4	58.0	76.0	64.0	54.0	52.5	55.8	88.0	70.5	26.3
PseuZO (ours)	91.8	58.4	77.0	68.9	58.0	55.3	57.1	90.0	71.5	27.5

Table 8: Experiments on OPT-6.7B with 1024 training samples and 512 evaluation samples. When training, for WSC, CB and COPA, they have total much less samples and thus we set aside 100 evaluation samples and use the rest for training. The **bold** number represents the highest evaluation performance.

A.4 More Ablation Experiments

As we mentioned above, PseuZO seems to degenerate to MeZO without momentum when the coefficient $\lambda \rightarrow 0$. Table 9 shows the relevant ablation experiments including MeZO with momentum,

MeZO with the sliding window and PseuZO without momentum ($\lambda = 0$). Experimental results show that PseuZO is effective by applying a moving average to the Jacobian. Simply applying a moving average to MeZO or just using the idea of the Jacobian alone is not particularly effective.

Task	SST-2	RTE	CB	BoolQ	WSC	WIC	MultiRC	COPA	ReCoRD	DROP
Task type	classification						_multiple choice _		generation	
MeZO	82.4	54.3	76.0	60.7	51.0	50.9	54.9	74.0	57.6	20.3
MeZO w/ Momentum	81.4	53.9	78.0	60.5	51.0	50.8	54.9	75.0	57.8	22.1
MeZO w/ the Sliding Window	85.3	54.3	79.0	62.5	51.0	54.9	55.6	73.0	60.2	20.1
PseuZO w/o Momentum	84.6	57.2	75.0	64.2	50.0	51.3	57.2	75.0	59.6	24.1
PseuZO (ours)	91.2	58.0	77.0	64.3	58.0	54.5	54.7	78.0	60.0	23.5

Table 9: Experiments on OPT-1.3B with 1024 training samples and 512 evaluation samples. In order to demonstrate the effect of performing a moving average on the Jacobian matrix, we conducted additional ablation experiments.

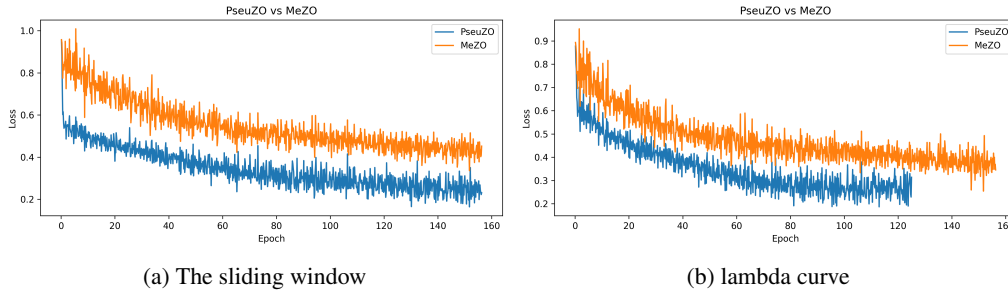


Figure 4: (a) is the loss curve for SST-2 task of OPT-1.3B model (b) is the loss curve for SST2 task of OPT-6.7B model.

Experiment	Hyperparameters	Values
MeZO	Batch size	16
	Learning rate	$\{1 \times 10^{-6}, 1 \times 10^{-7}\}$
	ϵ	1×10^{-3}
MeZO (prefix)	Batch size	16
	Learning rate	$\{1 \times 10^{-2}, 1 \times 10^{-3}\}$
	ϵ	1×10^{-1}
	#prefix tokens	5
MeZO (LoRA)	Batch size	16
	Learning rate	$\{1 \times 10^{-4}, 5 \times 10^{-5}\}$
	ϵ	$1e-2$
	(r, α)	(8,16)
MeZO-SVRG	Batch size	16
	Learning rate (η_1)	$\{1 \times 10^{-4}, 5 \times 10^{-5}\}$
	Learning rate (η_2)	$\{1 \times 10^{-6}, 1 \times 10^{-7}\}$
	ϵ	1×10^{-3}
	q	2
PseuZO	Batch size	16
	Learning rate	$\{1 \times 10^{-6}, 1 \times 10^{-7}\}$
	ϵ	1×10^{-3}
PseuZO (prefix)	Batch size	16
	Learning rate	$\{1 \times 10^{-2}, 1 \times 10^{-3}\}$
	ϵ	1×10^{-1}
	#prefix tokens	5
PseuZO (LoRA)	Batch size	16
	Learning rate	$\{1 \times 10^{-4}, 5 \times 10^{-5}\}$
	ϵ	1×10^{-2}
	(r, α)	(8,16)
FO-SGD	Batch size	8
	Learning rate	$\{1 \times 10^{-4}, 5 \times 10^{-5}, 1 \times 10^{-5}\}$

Table 10: The hyperparameter grids for OPT-1.3B and OPT6.7B and the weight decay is set to zero.

B Node Perturbation

In ZO optimization like SPSA or Weight Perturbation introduced next part, appropriate noise such as Gaussian and Rademacher noise is added to parameters to be optimized. This approach is slow to converge due to a large variance. Therefore, following [54], an appropriate reformulation is incorporated into PseuZO especially oriented to small data classification tasks to verify its fast convergence ability. In detail, if the l -th and the $(l + 1)$ -th activation layer output are denoted as x^l and x^{l+1} respectively, considering the neural network formulation with a FC layer as an example, $x^{l+1} = \phi(W^l x^l)$ and we can calculate gradients:

$$\nabla_{W^l} \ell = (\nabla_{x^{l+1}} \ell \odot \phi'(W^l x^l)) x^{l\top}. \quad (14)$$

We can actually add noise into activation value x^{l+1} rather than W^l to obtain $\nabla_{x^{l+1}} \ell$ by PseuZO and then compute $\nabla_{W^l} \ell$ through a simple formula which has a smaller variance than directly estimating weight gradients [26], and the exponential moving average further reduces the variance.

For MNIST, we leverage a FC neural network with two hidden layers composed of 2048 neurons following [30]. For CIFAR-10 and CIFAR-100, we leverage 4-layer and 9-layer Conv networks respectively. To fit respective tasks, we run 100 epochs for MNIST and 300 epochs for CIFAR-10 and CIFAR-100. We conduct comparable experiments with various signal feedback methods such

as single-point zero-order (ZO_{sp}), direct feedback alignment (DFA), DKP, which is based on DFA and updates weights similar to Kolen-Pollack learning and back propagation (BP) besides SPSA. The results in Table 7 demonstrate that the performance of traditional zero-order methods has a huge gap compared to BP, while PZO performs much better. We also compare the PseuZO with Node Perturbation against DFA with random feedback and its variant DKP and the complete experimental results are shown in Table 11.

When we incorporate local learning into PseuZO, it even has almost the same performance as BP. Unlike [54] which conducts experiments on spiking neural networks (SNNs) with surrogate gradients and the scale Weight Standard (sWS) skill[4], we need to reserve activation value and replace the sWS layer with BatchNorm (BN) layer.

Dataset	SPSA	ZO_{sp}	DFA	DKP	PseuZO	PseuZO (w/LL)	BP
MNIST	86.4 ± 0.15	87.8 ± 0.09	98.0 ± 0.03	98.3 ± 0.02	98.7 ± 0.02	/	98.5 ± 0.02
CIFAR-10	41.3 ± 0.74	42.6 ± 0.69	78.2 ± 0.30	86.5 ± 0.19	82.5 ± 0.15	88.7 ± 0.13	89.9 ± 0.06
CIFAR-100	5.39 ± 0.69	7.61 ± 0.73	47.6 ± 0.45	56.3 ± 0.24	61.4 ± 0.14	68.5 ± 0.13	71.9 ± 0.09

Table 11: Training from scratch on typical computer vision classification datasets for various feedback methods including DFA and its typical variant DKP. We do not use local loss for MNIST as there are only two hidden layers.

C Formal statement and proof of Theorem 4

We restate Assumption 2 and 3 as follows:

Assumption 6. \mathcal{F} has continuous Hessian matrices $\mathbf{H}(\mathbf{x})$, and satisfy the following equations:

$$\|\mathbf{H}(\mathbf{x})\|_{\text{op}} \leq L, \quad \text{tr}(\mathbf{H}(\mathbf{x})) \leq \alpha_1 L, \quad (15)$$

where α_1 is the effective dimension of \mathcal{F} . In the worst case, $\alpha_1 = d_p$.

Assumption 7. Denote $\hat{\nabla}\mathcal{F}(\mathbf{x}_t) = \nabla_{\mathbf{x}}g(h(\mathbf{x}_t; \mathbf{z}_t))$. The randomness of the Jacobian estimator \mathbf{B}_t can be decoupled into the following parts:

$$\mathbf{B}_t = (\mathbf{J}_t + \mathbf{D}_t)\mathbf{N}_t + \mathbf{M}_t, \quad (16)$$

where \mathbf{J}_t is the true Jacobian matrix of h at \mathbf{x}_t , and:

- \mathbf{N}_t represents the randomness introduced by the PZO Jacobian estimator:

$$\mathbb{E}\mathbf{N}_t = \mathbf{I}, \quad \mathbb{E}\|\mathbf{N}_t^\top \mathbf{a}\|_{\mathbf{M}}^2 \leq 3\text{tr}(\mathbf{M}) \cdot \|\mathbf{a}\|^2, \quad (17)$$

where \mathbf{a} is an arbitrary vector.

- \mathbf{D}_t represents the randomness of data:

$$(\mathbf{J}_t + \mathbf{D}_t)^\top \mathbf{e}_t = \hat{\nabla}\mathcal{F}(\mathbf{x}_t), \quad \mathbb{E}\mathbf{D}_t = 0, \quad \mathbb{E}\|\mathbf{D}_t^\top \mathbf{e}_t\|^2 \leq \sigma_2^2; \quad (18)$$

- \mathbf{M}_t represents the noise introduced by the two-point estimation of the function value, controlled by μ :

$$\mathbb{E}\|\mathbf{M}_t^\top \mathbf{e}_t\|^2 \leq \frac{\mu^2 L_h^2 L_g^2}{4} (d_p + 6)^3, \quad (19)$$

where we assume that $h(\mathbf{x}; \mathbf{z})$ is L_h -smooth with respect to \mathbf{x} , and g is L_g -Lipschitz.

We present the formal version of Theorem 4 as follows:

Theorem 8 (Formal version of Theorem 4). *Under Assumption 6, when*

$$\mu \leq \frac{\max\{\sigma_2, \epsilon\}}{L_g L_h (d_p + 6)^{3/2}}, \quad (20)$$

$$\lambda_t \leq \min \left\{ \frac{\epsilon}{16\|\mathbf{A}_{t-1}^\top \mathbf{e}_t\| + \epsilon}, 0.99 \right\}, \quad (21)$$

$$\eta \leq \min \left\{ \frac{1}{48\alpha_1 L}, \frac{\epsilon^2}{400\alpha_1 L \sigma_2^2} \right\}, \quad (22)$$

Algorithm 1 with a noisy Jacobian estimator \mathbf{B}_t satisfying Assumption 7 finds an ϵ -stationary point in $T = \frac{1600(\mathcal{F}(\mathbf{x}_0) - \mathcal{F}^*)}{\eta\epsilon^2}$ iterations when $\frac{\epsilon^2}{400\alpha L\sigma_2^2} \leq 1$.

Proof of Theorem 4. To simplify notations, we denote $\mathbf{a}_t = h(\mathbf{x}_t)$, and the Hessian matrix of h at \mathbf{x}_t \mathbf{J}_t . By the smoothness of g , we have:

$$\mathbb{E}_t \mathcal{F}(\mathbf{x}_{t+1}) - \mathcal{F}(\mathbf{x}_t) \leq \mathbb{E}_t \left(\langle \nabla \mathcal{F}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle + \frac{1}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|_{\mathbf{H}_t}^2 \right). \quad (23)$$

Now we apply the update rule of Algorithm 1 to obtain:

$$\begin{aligned} & \mathbb{E}_t \langle \nabla \mathcal{F}(\mathbf{x}_t), \mathbf{x}_{t+1} - \mathbf{x}_t \rangle \\ &= -\eta \mathbb{E}_t \langle \nabla \mathcal{F}(\mathbf{x}_t), \mathbf{A}_t^\top \mathbf{e}_t \rangle \\ &= -\eta(1 - \lambda_t) \mathbb{E}_t \langle \nabla \mathcal{F}(\mathbf{x}_t), \mathbf{B}_t^\top \mathbf{e}_t \rangle - \eta \lambda_t \langle \nabla \mathcal{F}(\mathbf{x}_t), \mathbf{A}_{t-1}^\top \mathbf{e}_t \rangle \\ &= -\eta(1 - \lambda_t) \|\nabla \mathcal{F}(\mathbf{x})\|^2 - \eta \lambda_t \langle \nabla \mathcal{F}(\mathbf{x}_t), \mathbf{A}_{t-1}^\top \mathbf{e}_t \rangle + \eta(1 - \lambda_t) \mathbb{E}_t \langle \mathbf{J}_t^\top \mathbf{e}_t, \mathbf{M}_t^\top \mathbf{e}_t \rangle \\ &\leq -\frac{\eta(1 - \lambda_t)}{4} \|\nabla \mathcal{F}(\mathbf{x})\|^2 + \frac{\eta \lambda_t^2}{1 - \lambda_t} \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\|^2 + \frac{\eta(1 - \lambda_t)}{2} \mathbb{E}_t \|\mathbf{M}_t^\top \mathbf{e}_t\|^2, \end{aligned} \quad (24)$$

and

$$\mathbb{E}_t \frac{1}{2} \|\mathbf{x}_{t+1} - \mathbf{x}_t\|_{\mathbf{H}_t}^2 \leq \eta^2 (1 - \lambda_t)^2 \mathbb{E}_t \|\mathbf{B}_t^\top \mathbf{e}_t\|_{\mathbf{H}_t}^2 + \eta^2 \lambda_t^2 \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\|_{\mathbf{H}_t}^2. \quad (25)$$

Summing up (24) and (25), we have:

$$\begin{aligned} & \mathbb{E}_t \mathcal{F}(\mathbf{x}_{t+1}) - \mathcal{F}(\mathbf{x}_t) \\ &\leq -\frac{\eta(1 - \lambda_t)}{4} \|\nabla \mathcal{F}(\mathbf{x})\|^2 + \frac{\eta \lambda_t^2}{1 - \lambda_t} \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\|^2 + \eta^2 \lambda_t^2 \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\|_{\mathbf{H}_t}^2 \\ &\quad + \eta^2 (1 - \lambda_t)^2 \mathbb{E}_t \|\mathbf{B}_t^\top \mathbf{e}_t\|_{\mathbf{H}_t}^2. \end{aligned} \quad (26)$$

With Assumption 7, We estimate the second-order moment of $\mathbf{B}_t^\top \mathbf{e}_t$ as follows:

$$\begin{aligned} \mathbb{E}_t \|\mathbf{B}_t^\top \mathbf{e}_t\|_{\mathbf{M}}^2 &\leq 2 \mathbb{E}_t \|\mathbf{N}_t^\top \hat{\nabla} \mathcal{F}(\mathbf{x}_t)\|_{\mathbf{M}}^2 + 2 \mathbb{E}_t \|\mathbf{N}_t^\top \mathbf{M}_t^\top \mathbf{e}_t\|_{\mathbf{M}}^2 \\ &\leq 6 \text{tr}(\mathbf{M}) \left(\mathbb{E}_{\hat{\nabla} \mathcal{F}} \|\hat{\nabla} \mathcal{F}(\mathbf{x}_t)\|^2 + \mathbb{E}_t \|\mathbf{M}_t^\top \mathbf{e}_t\|^2 \right) \\ &\leq 6 \text{tr}(\mathbf{M}) \left(\|\nabla \mathcal{F}(\mathbf{x}_t)\|^2 + \sigma_2^2 + \mathbb{E}_t \|\mathbf{M}_t^\top \mathbf{e}_t\|^2 \right). \end{aligned} \quad (27)$$

Applying (27) into (26), we have:

$$\begin{aligned} & \mathbb{E}_t \mathcal{F}(\mathbf{x}_{t+1}) - \mathcal{F}(\mathbf{x}_t) \\ &\leq -\frac{\eta(1 - \lambda_t)}{4} \|\nabla \mathcal{F}(\mathbf{x})\|^2 + \frac{\eta \lambda_t^2}{1 - \lambda_t} \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\|^2 + \eta^2 \lambda_t^2 \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\|_{\mathbf{H}_t}^2 \\ &\quad + 6\eta^2 (1 - \lambda_t)^2 \text{tr}(\mathbf{H}_t) \left(\|\nabla \mathcal{F}(\mathbf{x}_t)\|^2 + \sigma_2^2 + \mathbb{E}_t \|\mathbf{M}_t^\top \mathbf{e}_t\|^2 \right) \\ &\leq -\frac{\eta(1 - \lambda_t)}{4} \|\nabla \mathcal{F}(\mathbf{x})\|^2 + \frac{\eta \lambda_t^2}{1 - \lambda_t} \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\|^2 + \eta^2 \lambda_t^2 L^2 \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\|^2 \\ &\quad + 6\eta^2 (1 - \lambda_t)^2 \alpha_1 L \left(\|\nabla \mathcal{F}(\mathbf{x}_t)\|^2 + \sigma_2^2 + \frac{\mu^2 L_h^2 L_g^2}{4} (d_p + 6)^3 \right). \end{aligned} \quad (28)$$

By the assumptions on η , λ_t and μ , we have:

$$\begin{aligned} \frac{\eta}{1 - \lambda_t} (1 + \eta(1 - \lambda_t)) \left(\lambda_t \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\| \right)^2 &\leq 2\eta(1 - \lambda_t) \left(\frac{\lambda_t}{1 - \lambda_t} \|\mathbf{A}_{t-1}^\top \mathbf{e}_t\| \right)^2 \\ &\leq \frac{1}{128} \eta(1 - \lambda_t) \epsilon^2, \end{aligned} \quad (29)$$

and

$$\begin{aligned}
6\eta^2(1-\lambda_t)^2\alpha_1L\left(\sigma_2^2 + \frac{\mu^2L_h^2L_g^2}{4}(d_p+6)^3\right) &\leq 6\eta\alpha_1L \cdot \eta(1-\lambda_t)\left(\sigma_2^2 + \frac{\sigma_2^2 + \epsilon^2}{4}\right) \\
&\leq \frac{\eta(1-\lambda_t)\epsilon^2}{32} + \eta(1-\lambda_t)\frac{15}{2}\eta\alpha_1L\sigma_2^2 \\
&\leq \frac{1}{20}\eta(1-\lambda_t)\epsilon^2.
\end{aligned} \tag{30}$$

Therefore, we have:

$$\mathbb{E}_t\mathcal{F}(\mathbf{x}_{t+1}) - \mathcal{F}(\mathbf{x}_t) \leq -\frac{\eta(1-\lambda_t)}{8}\|\nabla\mathcal{F}(\mathbf{x}_t)\|^2 + \frac{\eta(1-\lambda_t)}{16}\epsilon^2. \tag{31}$$

Taking expectation and summing over t from 0 to T , we have:

$$\mathbb{E}\mathcal{F}(\mathbf{x}_T) - \mathcal{F}(\mathbf{x}_0) \leq -\frac{\eta}{8}\mathbb{E}\sum_{t=0}^T(1-\lambda_t)\|\nabla\mathcal{F}(\mathbf{x}_t)\|^2 + \frac{\eta}{16}\epsilon^2\sum_{t=0}^T(1-\lambda_t). \tag{32}$$

Therefore, we have:

$$\begin{aligned}
\frac{1}{\sum_{t=0}^T(1-\lambda_t)}\sum_{t=0}^T(1-\lambda_t)\|\nabla\mathcal{F}(\mathbf{x}_t)\|^2 &\leq \frac{8}{\eta\sum_{t=0}^T(1-\lambda_t)}(\mathcal{F}(\mathbf{x}_0) - \mathcal{F}^*) + \frac{1}{2}\epsilon^2 \\
&\leq \frac{800}{\eta T}(\mathcal{F}(\mathbf{x}_0) - \mathcal{F}^*) + \frac{1}{2}\epsilon^2.
\end{aligned} \tag{33}$$

Taking $T = \frac{1600(\mathcal{F}(\mathbf{x}_0) - \mathcal{F}^*)}{\eta\epsilon^2}$, we have:

$$\frac{1}{\sum_{t=0}^T(1-\lambda_t)}\sum_{t=0}^T(1-\lambda_t)\|\nabla\mathcal{F}(\mathbf{x}_t)\|^2 \leq \epsilon^2. \tag{34}$$

Therefore, the PseuZO algorithm finds an ϵ -stationary point in T iterations. \square

Finally, Lemma 9 ensures that the Jacobian estimator $\mathbf{B}_t = \left(\frac{h(\mathbf{x}_t + \mu\boldsymbol{\xi}_t; \mathbf{z}_t) - h(\mathbf{x}_t; \mathbf{z}_t)}{\mu}\right)\boldsymbol{\xi}_t^\top$ satisfies Assumption 7:

Lemma 9. Assume that $h(\mathbf{x}; \mathbf{z})$ is L_h -smooth with respect to h , and g is L_g -Lipschitz. Define $\mathbf{J}_t + \mathbf{D}_t = \nabla_{\mathbf{x}}h(\mathbf{x}_t; \mathbf{z}_t)$ and $\mathbf{N}_t = \boldsymbol{\xi}_t\boldsymbol{\xi}_t^\top$. Then the Jacobian estimator $\mathbf{B}_t = \left(\frac{h(\mathbf{x}_t + \mu\boldsymbol{\xi}_t; \mathbf{z}_t) - h(\mathbf{x}_t; \mathbf{z}_t)}{\mu}\right)\boldsymbol{\xi}_t^\top$ can be written as $\mathbf{B}_t = (\mathbf{J}_t + \mathbf{D}_t)\mathbf{N}_t + \mathbf{M}_t$, where \mathbf{N}_t and \mathbf{M}_t satisfies Assumption 7.

Proof of Lemma 9.

$$\mathbb{E}\|\mathbf{N}_t^\top \mathbf{a}\|_{\mathbf{M}}^2 = \mathbb{E}\mathbf{a}^\top \boldsymbol{\xi}_t \boldsymbol{\xi}_t^\top \mathbf{M} \boldsymbol{\xi}_t \boldsymbol{\xi}_t^\top \mathbf{a} = \mathbf{a}^\top (\text{tr}(\mathbf{M})\mathbf{I} + 2\mathbf{M})\mathbf{a} \leq 3\text{tr}(\mathbf{M})\|\mathbf{a}\|^2. \tag{35}$$

By the Taylor expansion of $h(\mathbf{x}_t; \mathbf{z}_t)$ at \mathbf{x}_t , we have:

$$\begin{aligned}
\mathbb{E}\|\mathbf{M}_t^\top \mathbf{e}_t\|^2 &= \mathbb{E}\left\|\left(\mathbf{B}_t - (\mathbf{J}_t + \mathbf{D}_t)\mathbf{N}_t\right)^\top \mathbf{e}_t\right\|^2 \\
&= \mathbb{E}\left\|\frac{1}{\mu}\left(h(\mathbf{x}_t + \mu\boldsymbol{\xi}_t; \mathbf{z}_t) - h(\mathbf{x}_t; \mathbf{z}_t) - \mu\nabla_{\mathbf{x}}h(\mathbf{x}_t; \mathbf{z}_t)\boldsymbol{\xi}_t\right)\boldsymbol{\xi}_t^\top \mathbf{e}_t\right\|^2 \\
&\leq \frac{1}{4}\mu^2L_h^2\|\mathbf{e}_t\|^2\mathbb{E}\|\boldsymbol{\xi}_t\|^6 \\
&\leq \frac{1}{4}\mu^2L_h^2L_g^2(d_p+6)^3,
\end{aligned} \tag{36}$$

where the last line uses Lemma 1 of [36]. \square

D Theoretical advantage of PseuZO gradient estimator

In this section, we demonstrate the advantage of the PseuZO gradient estimator by comparing to MeZO and MeZO with momentum. At step t , PseuZO receives a noisy Jacobian estimation $\mathbf{B}_t = \mathbf{J}_t + \boldsymbol{\Xi}_t$, while MeZO receives a perturbed gradient $\mathbf{g}_t = (\mathbf{J}_t + \boldsymbol{\Xi}_t)^\top \mathbf{e}_t$, with $\mathbb{E}[\boldsymbol{\Xi}_t] = \mathbf{0}$, $\mathbb{E}[\boldsymbol{\Xi}_t \boldsymbol{\Xi}_t^\top] = \mathbf{N}$.

The update rule of the three algorithms are:

- **PseuZO:** $\mathbf{A}_t = (1 - \lambda)\mathbf{B}_t + \lambda\mathbf{A}_{t-1}$, $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\mathbf{A}_t^\top \mathbf{e}_t$.
- **MeZO:** $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta\mathbf{B}_t^\top \mathbf{e}_t$.
- **MeZO with momentum:** $\mathbf{m}_t = (1 - \lambda)\mathbf{B}_t^\top \mathbf{e}_t + \lambda\mathbf{m}_{t-1}$, $\mathbf{x}_{t+1} = \mathbf{x}_t - \mathbf{m}_t$.

Their corresponding gradient estimators are:

- **PseuZO:** $\mathbf{A}_t^\top \mathbf{e}_t$.
- **MeZO:** $\mathbf{B}_t^\top \mathbf{e}_t$.
- **MeZO with momentum:** $\eta\mathbf{m}_t$.

D.1 Variance analysis

The gradient estimation variance are:

- **PseuZO:** $(1 - \lambda)^2 \sum_{i=1}^t \lambda^{2(t-i)} \mathbf{e}_t^\top \mathbf{N} \mathbf{e}_t$.
- **MeZO:** $\mathbf{e}_t^\top \mathbf{N} \mathbf{e}_t$.
- **MeZO with Momentum:** $(1 - \lambda)^2 \sum_{i=1}^t \lambda^{2(t-i)} \mathbf{e}_i^\top \mathbf{N} \mathbf{e}_i$.

Theoretical implication:

PseuZO achieves strictly lower variance than both MeZO and MeZO with momentum due to two fundamental mechanisms:

1. **vs. MeZO with Momentum:** As optimization converges ($\mathbf{e}_t \rightarrow 0$), historical gradients satisfy $\mathbf{e}_i^\top \mathbf{N} \mathbf{e}_i \gg \mathbf{e}_t^\top \mathbf{N} \mathbf{e}_t$ for $i < t$. This is a significant advantage as the noise matrix \mathbf{N} often scale as the parameter size. Thus, PseuZO's *current-gradient coupling* inherently suppresses noise amplification from outdated gradients.
2. **vs. MeZO:** PseuZO's variance coefficient $\frac{(1-\lambda)(1-\lambda^{2t})}{1+\lambda}$ is strictly less than 1 for $\lambda \in (0, 1)$ and $t > 0$, while MeZO's coefficient is 1. This mathematically guarantees lower variance.

PseuZO's closed-form outer gradient exploitation uniquely minimizes both historical noise accumulation (vs. momentum) and instantaneous noise scaling (vs. MeZO), constituting its core advantage.

D.2 Bias analysis

- **PseuZO:** $(1 - \lambda) \left(\sum_{i=1}^t \lambda^{(t-i)} \mathbf{J}_i \right)^\top \mathbf{e}_t - \mathbf{J}_t^\top \mathbf{e}_t$.
- **MeZO:** Unbiased.
- **MeZO with Momentum:** $(1 - \lambda) \left(\sum_{i=1}^t \lambda^{(t-i)} \mathbf{J}_i^\top \mathbf{e}_i \right) - \mathbf{J}_t^\top \mathbf{e}_t$.

Under small learning rate assumptions (ignoring higher-order terms), with $\boldsymbol{\delta}_t = (1 - \lambda) \sum_{i=1}^t \lambda^{(t-i)} (\mathbf{x}_i - \mathbf{x}_t)$, the bias of PseuZO and MeZO+momentum gradient estimators are:

- **PseuZO bias:** $-\lambda^t \nabla f(\mathbf{x}_t) + \left(\sum_{i=1}^m (\mathbf{e}_t)_i \nabla^2 h_i(\mathbf{x}_t) \right) \boldsymbol{\delta}_t$.
- **MeZO with momentum bias:** $-\lambda^t \nabla f(\mathbf{x}_t) + \nabla^2 f(\mathbf{x}_t) \boldsymbol{\delta}_t$.

Theoretical implication: For $f(\mathbf{x}) = g(h(\mathbf{x}))$, the Hessian decomposes as:

$$\nabla^2 f(\mathbf{x}_t) = \underbrace{\sum_{i=1}^m (\mathbf{e}_t)_i \nabla^2 h_i(\mathbf{x}_t)}_{\text{PseuZO captures}} + \underbrace{\mathbf{J}_t^\top \nabla^2 g|_{\mathbf{y}=h(\mathbf{x}_t)} \mathbf{J}_t}_{\text{Extra term in MeZO}}$$

When g is convex ($\nabla^2 g \succeq 0$), MeZO's bias contains an additional positive semi-definite term. PseuZO inherently eliminates this curvature-induced bias, particularly when g exhibits strong nonlinearity – a fundamental advantage not relying on fixed Jacobian assumptions.