Leveraging Coordinate Momentum in SignSGD and Muon: Memory-Optimized Zero-Order LLM Fine-Tuning

Grigoriy Evseev
MIPT, Moscow

Egor Petrov
MIPT, Moscow

Veprikov Andrey
MIPT, Moscow
ISP RAS, Moscow

Aleksandr Beznosikov
MIPT, Moscow
Innopolis University, Innopolis

Abstract

Fine-tuning Large Language Models (LLMs) is essential for adapting pre-trained models to downstream tasks. Yet traditional first-order optimizers such as Stochastic Gradient Descent (SGD) and Adam incur prohibitive memory and computational costs that scale poorly with model size. In this paper, we investigate zero-order (ZO) optimization methods as a memory- and compute-efficient alternative, particularly in the context of parameter-efficient fine-tuning techniques like LoRA. We propose `JAGUAR SignSGD`, a ZO momentum-based algorithm that extends ZO SignSGD, requiring the same number of parameters as standard ZO SGD and only $\mathcal{O}(1)$ function evaluations per iteration. To the best of our knowledge, this is the first study to establish rigorous convergence guarantees of SignSGD in the stochastic ZO case. We further propose `JAGUAR Muon`, a novel ZO extension of the Muon optimizer that leverages the matrix structure of model parameters, and we provide its convergence rate under arbitrary stochastic noise. Through extensive experiments on challenging LLM fine-tuning benchmarks, we demonstrate that proposed algorithms meet or exceed the convergence quality of standard first-order methods, achieving significant memory reduction. Our theoretical and empirical results establish new ZO optimization methods as a practical and theoretically grounded approach for resource-constrained LLM adaptation.

1 Introduction

Fine-tuning pre-trained large language models (LLMs) has become the de-facto standard in modern natural language processing, enabling rapid adaptation to diverse downstream tasks with minimal labeled data [48, 54]. These models, often trained on massive corpora, achieve state-of-the-art results when fine-tuned on specific applications, including question answering, summarization, and dialogue generation. The fine-tuning setup can be considered as a stochastic unconstrained optimization problem of the form

$$f^* := \min_{x \in \mathbb{R}^d} \left\{ f(x) := \mathbb{E}_{\xi \sim \mathcal{D}} \left[f(x, \xi) \right] \right\}, \tag{1}$$

where f is the loss function, x are parameters of the ML model, $\mathcal{D}$ is the data distribution available for training and $f(x, \xi)$ is the loss with data point ξ.

The de facto standard for solving (1) is the use of first-order (FO) optimization methods. These approaches assume access to the stochastic gradient $\nabla f(x, \xi)$. Classical FO methods, such as Stochastic Gradient Descent (SGD) [1] and Adam [29], remain the most widely used techniques for model adaptation due to their efficiency and compatibility with the backpropagation algorithm. Nevertheless, in contemporary fine-tuning tasks, alternative FO algorithms are often preferred.

SignSGD has recently emerged as a standard optimization method for fine-tuning LLMs, owing to its simplicity, memory efficiency, and surprising empirical effectiveness across a range of adaptation tasks. It was first rigorously analyzed in the FO setting by [5] and [2]. These works showed that compressing each gradient component to its sign reduces per-iteration communication costs and can empirically outperform SGD [57]. Minimal memory usage and straightforward hyperparameter tuning make SignSGD an attractive choice for memory-constrained fine-tuning of LLMs.

Also, a recent trend in optimization for LLMs is to represent optimization parameters in matrix form rather than as vectors [4, 3, 46]. Algorithms such as Shampoo [22] and SOAP [61] have demonstrated superior performance on LLM training tasks compared to Adam and SGD [10], which operate in an element-wise manner and do not utilize the underlying structure of the model parameters. Currently, the canonical matrix-based optimization algorithm is Muon [28, 36, 33], which integrates the principles of Shampoo and SOAP but does not employ any preconditioning matrices [28]. The central idea of this method is to project the gradient at each iteration onto the space of semi-orthogonal matrices using the Newton–Schultz algorithm [4].

However, as LLMs continue to scale, backpropagation procedure, necessary for FO methods, becomes increasingly expensive in terms of memory consumption. For instance, the memory cost of computing gradients during the training of OPT-13B is reported to be more than an order of magnitude larger than that of inference [68]. This imbalance poses a serious bottleneck for deploying LLM fine-tuning in resource-constrained environments such as edge devices [67, 18], consumer-grade GPUs [35, 63], or large-scale distributed settings [23]. To mitigate this challenge, other FO methods have been proposed to lower memory consumption. One of the first proposed methods is AdaFactor [56], which reduces memory consumption by storing only one value per block. Also, memory usage reduction can be achieved by quantization for optimizer states to the lower-precision representations [12, 32] and fusing the backward operation with the optimizer update [40].

Nevertheless, the most memory efficient methods are based on Zero-Order (ZO) optimization technique, which avoids backpropagation entirely by estimating gradients using only forward passes. This flexibility allows to treat the model as a black box, optimizing performance with minimal assumptions about its architecture or implementation details. Recent studies [41] have demonstrated the practical benefits of this approach: for example, the MeZO algorithm applies classical ZO SGD [19] to fine-tune LLMs while maintaining four times lower memory requirements than traditional FO methods [66]. In the ZO methods it is assumed that we only have access to the values of the stochastic function $f(x, \xi)$ from (1) [17, 19]. Within LLMs pretraining or fine-tuning context, oracles are forward passes with small perturbations in parameters of the model. To estimate gradients authors use finite differences:

$$\nabla f(x, \xi) \approx \widetilde{\nabla} f(x, \xi) = \frac{f(x + \tau e, \xi) - f(x - \tau e, \xi)}{2\tau} e, \tag{2}$$

where $\tau > 0$ is a small parameter, frequently referenced to as a smoothing parameter, and $e \in \mathbb{R}^d$ is some random vector [44, 13].

2 Related Work and Our Contributions

ZO gradient estimators. The simplest zero-order gradient estimator employs estimate (2) as the stochastic gradient, however, even this approach presents specific challenges, particularly regarding the selection of an appropriate distribution from which to sample the random vector e. The most commonly employed distributions include a uniform distribution over the unit sphere: $e \sim RS(1)^d_{\|\cdot\|}$ [17, 44], a Gaussian distribution with zero mean and identity covariance matrix: $e \sim \mathcal{N}(0, I)$ [44, 19], and standard basis one-hot vectors [13, 55]. Also, some papers [34] utilize the so-called full coordinate estimate, which approximates the gradient across all basis vectors. However, this approach requires $\mathcal{O}(d)$ calls to the zero-order oracle, making it impractical for large-scale fine-tuning tasks. Despite the prevalence of these approaches, alternative and more complicated sampling strategies have also been explored.

In [52, 45], the authors explore low-dimensional perturbations within random subspaces. The central concept of random subspace methods involves generating the perturbation vector e within a subspace spanned by a projection matrix $P \in \mathbb{R}^{d\times r}$ and a low-dimensional random vector $\tilde{e} \in \mathbb{R}^r$: $e = P\tilde{e}$. Typically, P and $\tilde{e}$ are sampled from a Gaussian distribution and $r \ll d$. The primary motivation for this method lies in the fact that gradients during the fine-tuning process exhibit a low-dimensional structure. In [39, 62], the authors employ a masked random vector e, wherein at each iteration a random mask with r non-zero elements $m_r \in \{0, 1\}^d$ is generated and applied element wise to a Gaussian vector e. This procedure accelerates the optimization step, as only the parameters corresponding to the active entries in m_r are updated, rather than the entire parameter set. In contrast, the authors of [21] depart from random mask sampling at each iteration and instead select an optimal mask m_r prior to training, according to a specific criterion. Consequently, under the update rule (2), only the parameters $x \odot m_r$ are modified during optimization. In our approach, we similarly do not utilize all coordinates of the random vector e in each estimation of (2), instead, we select a single coordinate at each step. However, unlike previous works [39, 62, 21], we do not discard information from the remaining coordinates, but accumulate information from previous iterations.

In our work, we employ the JAGUAR zero-order gradient estimation technique [60], which integrates the concept of sampling one-hot basis vectors with the utilization of a SAGA-like momentum update [11]. This approach facilitates convergence in stochastic setting by leveraging memory from past iterations, while using the same amount of memory as standard zero-order methods like ZO SGD (MeZO) [42]. In the original paper [60], the authors do not incorporate a momentum parameter, discarding coordinate information from previous iterations. In contrast, we introduce a momentum parameter, $0 \le \beta \le 1$ (see Algorithms 1 and 2), which controls the utilization of gradients from past iterations. We demonstrate that adding this momentum β allows the method to converge in the stochastic non-convex case (see Theorems 3.5 and 3.6).

Momentum techniques. Numerous zero-order methods in the literature incorporate momentum techniques in various forms. However, these approaches typically introduce multiple additional variables of dimension d. Because zero-order methods are often chosen for fine-tuning tasks to save memory, the inclusion of such extra variables becomes a critical limitation in these settings. In [25] authors use variance reduction technique SPIDER [16] that uses approximately $5d$ parameters: $2d$ for ZO gradients, $2d$ for model parameters and $1d$ for momentum. In [7, 26], the authors employ the Adam optimization technique [29], which is frequently used for stochastic non-convex optimization problems [8, 15]. However, this technique incurs a significant memory overhead, requiring $4d$ parameters. The paper [50] utilizes classical heavy-ball momentum within a zero-order framework, provided, only demonstrating almost sure convergence to a constant in the non-convex setting. It is worth noting that numerous other zero-order techniques exist in the literature to achieve convergence when the function f is convex [20, 44, 13], satisfies conditions like PL [50] or ABG [49], or in deterministic settings [20]. Since our focus is on fine-tuning problems, which fall under the non-convex case, we will not discuss these methods in detail.

Matrix ZO optimization. In the context of zero-order optimization, transitioning to matrix-valued parameters necessitates replacing the random vector $e \in \mathbb{R}^d$ in zero-order gradient approximation (2) with a random matrix $E \in \mathbb{R}^{m\times n}$, and correspondingly, projecting this matrix E to semi-orthogonal space, as is done in the Muon algorithm [28]. Since the random matrix E is typically drawn from a known distribution, it is possible to directly sample orthogonal matrices when computing the gradient estimator (2). A similar approach has previously appeared in the zero-order optimization literature [9]; however, that work did not consider the Muon algorithm, but rather focused on sampling two orthogonal matrices $V \in \mathbb{R}^{m\times r}$ and $U \in \mathbb{R}^{n\times r}$ of rank $r \ll \min\{m, n\}$. This approach does not correspond to the decomposition of the random matrix E, as E is almost surely of full rank. Additionally, alternative techniques for sampling low-rank matrices have been proposed in the literature. For instance, in [64], a method analogous to the sampling of low-rank vectors described in [52, 45] is utilized.

We present a summary of relevant results from the existing zero-order literature in Table **??**

2.1 Our Contributions

So far, although zero-order methods have started to gain popularity in the field of LLM fine-tuning, only basic ZO methods have been considered. In this work, having considered the Jaguar technique for Momentum, we have also considered it for the ZO versions of the Muon and SignSGD algorithms. Specifically, we make the following key contributions:

- We establish the first convergence analysis for ZO SignSGD with momentum, using only $2d+1$ oracle parameters.
- We introduce ZO Muon within the zero-order setup, integrating momentum while preserving memory efficiency.
- We evaluate the proposed ZO methods on challenging LLM fine-tuning benchmarks, demonstrating their practical viability.

3 Main results

3.1 Preliminaries

We now provide several assumptions that are necessary for the analysis.

Assumption 3.1 (Smoothness)**.** The functions $f(x, \xi)$ are $L(\xi)$-smooth on the $\mathbb{R}^d$ with respect to the Euclidean norm $\|\cdot\|$, i.e., for all $x, y \in \mathbb{R}^d$ it holds that

$$\|\nabla f(x, \xi) - \nabla f(y, \xi)\|_2 \leq L(\xi)\|x - y\|_2.$$

We also assume that exists constant $L^2 := \mathbb{E}\left[L(\xi)^2\right]$.

Assumption 3.2 (Bounded variance of the gradient)**.** The variance of the $\nabla f(x, \xi)$ is bounded with respect to the Euclidean norm, i.e., there exists $\sigma > 0$, such that for all $x \in \mathbb{R}^d$ it holds that

$$\mathbb{E}\left[\|\nabla f(x, \xi) - \nabla f(x)\|_2^2\right] \leq \sigma^2.$$

We assume access only to a zero-order oracle, which returns a noisy evaluation of the function $f(x, \xi)$. This noise may originate not only from inherent randomness (stochastic noise), but also from systematic effects (deterministic noise), such as computer rounding errors. Therefore, we make a common assumption about the function $\hat{f}(x, \xi)$ returned by the oracle [14, 60].

Assumption 3.3 (Bounded oracle noise)**.** The noise in the oracle is bounded with respect to the Euclidean norm, i.e., there exists $\Delta > 0$, such that for all $x \in \mathbb{R}^d$ it holds that

$$\mathbb{E}\left[\left|\hat{f}(x, \xi) - f(x, \xi)\right|^2\right] \leq \Delta^2.$$

Assumptions 3.1 and 3.2 are standard in the theoretical analysis of stochastic non-convex zero-order optimization problems [50, 21, 39, 62]. In contrast, Assumption 3.3 is frequently omitted in the existing literature, as it is commonly presumed that $\Delta = 0$, implying access to an ideal zero-order oracle. However, this assumption does not hold in practice, as numerical errors such as machine precision inevitably introduce a non-zero perturbation. Consequently, while Δ is typically small, it is never zero, which does not allow us to restore a true gradient along the direction e in estimation (2) if we set $\tau \to 0$.

3.2 Zero-Order Momentum SignSGD with JAGUAR Gradient Approximation

In this section, we introduce zero-order SignSGD algorithm with JAGUAR gradient approximation [60] and momentum of the form:

Algorithm 1 Zero-Order Momentum SignSGD with JAGUAR [60] (`JAGUAR SignSGD`)

```
1: Parameters: stepsize (learning rate) $\gamma$, momentum $\beta$, gradient approximation parameter $\tau$,
   number of iterations $T$.
2: Initialization: choose $x^0 \in \mathbb{R}^d$ and $m^{-1} = \mathbf{0} \in \mathbb{R}^d$.
3: for $t = 0, 1, 2, \ldots, T$ do
4:     Sample $i_t \sim$ Uniform($\overline{1, d}$)
5:     Set one-hot vector $e^t$ with 1 in the $i_t$ coordinate: $e^t_{i_t} = 1$ and $e^t_{i \neq i_t} = 0$ for all $i \in \overline{1, d}$
6:     Sample stochastic variable $\xi^t \sim \mathcal{D}$
7:     Compute $\widetilde{\nabla}_{i_t} f(x^t, \xi^t) := \frac{\hat{f}(x^t + \tau e^t, \xi^t) - \hat{f}(x^t - \tau e^t, \xi^t)}{2\tau} \in \mathbb{R}$
8:     Set $m^t_{i_t} = \beta m^{t-1}_{i_t} + (1 - \beta)\widetilde{\nabla}_{i_t} f(x^t, \xi^t)$ and $m^t_{i \neq i_t} = m^{t-1}_{i \neq i_t}$ for all $i \in \overline{1, d}$
9:     Set $x^{t+1} = x^t - \gamma \cdot \text{sign}(m^t)$
10: end for
11: Return: $x^{N(T)}$, where $N(T) \sim$ Uniform($\overline{1, T}$).
```

The gradient approximation employed in Algorithm 1 deviates from that of the original JAGUAR method [60], as we introduce a momentum variable β. The estimator from the original work can be recovered by setting $\beta = 0$.

We now present a lemma characterizing the closeness between the momentum variable m^t from line 8 and the true gradient $\nabla f(x^t)$.

Lemma 3.4. *Consider m^t from line 8 of Algorithm 1. Under Assumptions 3.1, 3.2 and 3.3 it holds that:*

$$\mathbb{E}\left[\left\|m^t - \nabla f(x^t)\right\|_2^2\right] = \mathcal{O}\left[\frac{d^3 L^2 \gamma^2}{(1-\beta)^2} + (1-\beta) d\sigma^2 + dL^2\tau^2 + \frac{2d\Delta^2}{\tau^2} + \left(\frac{1-\beta}{d}\right)^t \left\|\nabla f(x^0)\right\|_2^2\right].$$

Discussion. This lemma closely parallels Lemma 1 from [60], with the key distinction that our analysis incorporates the momentum parameter β, which was not present in [60]. The introduction of momentum is essential for proving convergence of algorithms such as signSGD (Algorithm 1) and Muon (Algorithm 2) in the stochastic zero-order setting [59], as it enables more careful handling of variance σ in the gradient estimates (2). Another important difference from prior works is that our result does not involve the term $\|\nabla f(x^t)\|_2^2$, which typically appears in analyses where the zero-order gradient estimator (2) is constructed using random uniform or Gaussian vectors e [6, 31, 20, 47]. With the presence of the term $\|\nabla f(x^t)\|_2^2$, it is not possible to achieve convergence guarantees for signSGD (Algorithm 1) and Muon (Algorithm 2) in the stochastic zero-order setting. It is worth noting that a similar result can be obtained when using a full coordinate estimator [34], however, this approach requires $\mathcal{O}(d)$ calls to the zero-order oracle per iteration, which can be computationally expensive. In contrast, the JAGUAR method achieves the same result with only $\mathcal{O}(1)$ oracle calls and with the same number of parameters, offering significant improvements in efficiency. This makes our approach particularly attractive for large-scale optimization tasks, where reducing oracle complexity is critical.

Now with the help of Lemma 3.4 we provide convergence analysis of the `JAGUAR SignSGD` Algorithm 1.

Theorem 3.5. *Consider Assumptions 3.1, 3.2 and 3.3. Then the `JAGUAR SignSGD` Algorithm 1 has the following convergence rate:*

$$\frac{1}{T}\sum_{t=0}^{T} \mathbb{E}\left[\left\|\nabla f(x^t)\right\|_1\right] = \mathcal{O}\left[\frac{\delta_0}{\gamma T} + \frac{d\left\|\nabla f(x^0)\right\|_2}{T\sqrt{1-\beta}} + \frac{d^2 L\gamma}{1-\beta} + \sqrt{1-\beta}\, d\sigma + dL\tau + \frac{d\Delta}{\tau}\right],$$

where we used a notation $\delta_0 := f(x^0) - f^*$.

Discussion. The convergence rate established in this theorem is similar to what is known for first-order methods [5, 27, 53, 30], however our bounds include an additional factor of d, which is typical for all coordinate-based methods [43, 51], not just zero-order ones. This dependence on the dimension arises because coordinate methods process one direction at a time, accumulating complexity proportional to d. It is also important to note that without momentum ($\beta = 0$), the algorithm can only guarantee convergence to a neighbourhood of the optimum of size proportional to σ, as shown in previous works on zero-order signSGD [37, 30]. Let us also point out that we can not choose an arbitrary ε in Corollary **??**, since there exists an irreducible [14, 60] error Δ in the zero-order oracle (see Assumption 3.3). However, since Δ is very small, we can still achieve an acceptable accuracy ε.

3.3 Zero-Order Muon with JAGUAR Gradient Approximation

In this section, we address the matrix optimization setting, where the optimization variables X_t are elements of the matrix space $\mathbb{R}^{m\times n}$, rather than the standard vector space $\mathbb{R}^d$. For the first time in the literature, we introduce a zero-order version of the Muon algorithm (Algorithm 2), broadening the applicability to matrix-structured optimization tasks where only function evaluations are available.

Algorithm 2 Zero-Order Muon with JAGUAR [60] (`JAGUAR Muon`)

```
 1: Parameters: stepsize (learning rate) $\gamma$, momentum $\beta$, gradient approximation parameter $\tau$,
    number of Newton-Schulz steps ns_steps, number of iterations $T$.
 2: Initialization: choose $X^0 \in \mathbb{R}^{m\times n}$ and $M^{-1} = \mathbf{0} \in \mathbb{R}^{m\times n}$.
 3: for $t = 0, 1, 2, \dots, T$ do
 4:     Sample $i_t \sim$ Uniform($\overline{1, m}$) and $j_t \sim$ Uniform($\overline{1, n}$)
 5:     Set one-hot matrix $E^t$ with 1 in the $(i_t, j_t)$ coordinate
 6:     Sample stochastic variable $\xi^t \sim \mathcal{D}$
 7:     Compute $\widetilde{\nabla}_{i_t} f(X^t, \xi^t) := \frac{\hat{f}(X^t+\tau E^t,\xi^t)-\hat{f}(X^t-\tau E^t,\xi^t)}{2\tau} \in \mathbb{R}$
 8:     Set $M^t_{i_t,j_t} = \beta M^{t-1}_{i_t,j_t} + (1-\beta)\widetilde{\nabla}_{i_t} f(x^t, \xi^t)$ and $M^t_{i\neq i_t, j\neq j_t} = M^{t-1}_{i\neq i_t, j\neq j_t}$
 9:     Set $X^{t+1} = X^t - \gamma \cdot$ Newton_Schulz($M^t, K =$ ns_steps)
10: end for
11: Return: $X^{N(T)}$, where $N(T) \sim$ Uniform($\overline{1, T}$).
 1: Subroutine Newton_Schulz($A \in \mathbb{R}^{m\times n}, K = 10$) [4]:
 2:     Set $A^0 = A/\|A\|_F$
 3:     for $k = 0, 1, 2, \dots, K$ do
 4:         $A^{k+1} = 3/2 \cdot A^k - 1/2 \cdot A^k (A^k)^T A^k$
 5:     end for
 6:     Return: $A^K \approx U_A \cdot V_A^T$.
```

Algorithm 2 is similar to the first-order Muon algorithm [28], the only difference is that we use zero-order gradient approximation JAGUAR [60] in line 8.

Let us note that when extending to matrix-valued parameters, it is necessary to slightly modify Assumptions 3.1 and 3.2: all occurrences of the ℓ_2 norm $\|\cdot\|_2$ should be replaced with the Frobenius norm $\|\cdot\|_F$. This modification is justified, as the following property holds for all matrixes $A \in \mathbb{R}^{m\times n}$: $\|A\|_F = \|\overline{\text{vec}}(A)\|_2$. We now provide convergence analysis of the `JAGUAR Muon` Algorithm 2.

Theorem 3.6. *Consider Assumptions 3.1, 3.2 (with Frobenius norm) and 3.3. Then the `JAGUAR Muon` Algorithm 2 has the following convergence rate:*

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\left[\left\|\nabla f(X^t)\right\|_{\mathcal{S}_1}\right] = \mathcal{O}\left[\frac{\delta_0}{\gamma T} + \frac{m^{1/2} n \left\|\nabla f(x^0)\right\|_2}{T\sqrt{1-\beta}} + \frac{m^{3/2} n^2 \gamma}{1-\beta} + \sqrt{1-\beta}\, m^{1/2} n \sigma + m^{1/2} n L \tau + \frac{m^{1/2} n \Delta}{\tau}\right],$$

where we used a notation $\delta_0 := f(x^0) - f^*$. We also assume that $n \le m$.

Discussion. The convergence rate established in Theorem 3.6 is consistent with the first-order case. However, there remain zero-order terms depending on τ and Δ, as for Algorithm 1 (see Theorem 3.5 and Discussion part after it). From a proof perspective, Theorems 3.5 and 3.6 are very similar, since the orthogonalization operation (`Newton_Schulz`) in Algorithm 2 can be interpreted as taking the sign of the gradient matrix eigenvalues. Accordingly, both the form and the convergence rate criterion are analogous (the ℓ_1 norm for Algorithm 1 and the $\mathcal{S}_1$ norm for Algorithm 2). Nevertheless, the convergence rates of the two algorithms differ slightly. We examine the two boundary cases in the following remark.

Accordingly, comparing these convergence rates with that obtained in Corollary 1, we observe an improvement by factors of $d^{1/2}$ and $d^{1/4}$, respectively.

In addition to `JAGUAR Muon`, we introduce `ZO-MUON` Algorithm 3, a direct zero-order adaptation of the Muon, utilizing standard zero-order gradient estimation for comparison. We also explore various matrix sampling techniques 4 to enhance this approach, as detailed in Appendix A.

4 Experiments

In this section, we present a comprehensive empirical evaluation to validate the theoretical contributions of our proposed ZO optimization methods for fine-tuning large language models. Our study aims to assess both the accuracy and memory efficiency of these methods, comparing them against established ZO and FO baselines. We build upon the experimental framework proposed in [66], extending it to incorporate our novel algorithms: `JAGUAR SignSGD`, `ZO-Muon`, and `JAGUAR Muon`. The primary objective is to achieve competitive test accuracy on downstream tasks while maintaining memory efficiency comparable to the baseline methods.

4.1 Experimental Setup

Fine-Tuning Task and Schemes. Fine-tuning LLMs is a pivotal process in adapting pre-trained models to specific downstream tasks, enabling high performance with limited task-specific data. This process typically involves adjusting model parameters to minimize a task-specific loss function, a task that becomes computationally intensive as model sizes scale to billions of parameters. To explore the efficacy of our ZO methods, we focus on the SST2 dataset [58], a widely-used benchmark for binary sentiment classification. We evaluate two fine-tuning schemes:

- **Full Fine-Tuning (FT):** Updates all parameters of the pre-trained model, offering maximum flexibility at the cost of higher computational resources.
- **Low-Rank Adaptation (LoRA):** Introduces a small set of trainable parameters while keeping the original model parameters frozen, enhancing memory efficiency [24].

Models. We conduct experiments using two prominent LLMs: OPT-1.3B [65], a 1.3 billion parameter model from the OPT family, and RoBERTa-Large [38], a 355 million parameter model known for its robust performance in natural language processing tasks. These models represent a range of sizes and architectures, allowing us to assess the scalability of our methods.

Methods. We evaluate the following ZO optimization methods proposed in this work:

- `JAGUAR SignSGD`**:** Combines the JAGUAR gradient approximation [60] with SignSGD and momentum for efficient updates (Algorithm 1).
- `ZO-Muon`**:** A novel ZO adaptation of the Muon optimizer, leveraging matrix-based optimization principles (Algorithm 3).
- `JAGUAR Muon`**:** Integrates JAGUAR with the Muon optimizer, incorporating momentum and orthogonalization (Algorithm 2).

Comparison procedure. For comparison, we include baseline methods from [66]: ZO-SGD, ZO-SGD-MMT, ZO-SGD-Cons, ZO-SGD-Sign, ZO-Adam, Forward-Grad, and the FO method FO-SGD. The results for which are given in the benchmark paper. We perform experiments for our methods in accordance with similar experiments from [66]. For details of our parameter selection and model training procedure, see Appendix C.

4.2 Results

Accuracy Comparison. Table 1 presents the test accuracy results for SST2 across both models and fine-tuning schemes. Our proposed methods demonstrate strong performance, often outperforming baseline ZO methods.

Table 1: Test accuracy on SST2 for OPT-1.3B and RoBERTa-Large with FT and LoRA. Best performance among ZO methods is in **bold**. Blue indicates outperformance of all baselines, red indicates matching or exceeding FO-SGD.

Method	OPT-1.3B				RoBERTa-Large			
	FT	FT (std)	LoRA	LoRA (std)	FT	FT (std)	LoRA	LoRA (std)
FO-SGD	91.1	-	93.6	-	91.4	-	91.2	-
Forward-Grad	90.3	-	90.3	-	90.1	-	89.7	-
ZO-SGD	90.8	-	90.1	-	89.4	-	90.8	-
ZO-SGD-MMT	85.2	-	91.3	-	89.6	-	90.9	-
ZO-SGD-Cons	88.3	-	90.5	-	89.6	-	**91.6**	-
ZO-SGD-Sign	87.2	-	91.5	-	52.5	-	90.2	-
ZO-Adam	84.4	-	92.3	-	89.8	-	89.5	-
`JAGUAR SignSGD`	**94.0**	0.1	92.5	0.5	**93.5**	0.2	**92.0**	0.4
`ZO-Muon`	86.5	0.1	93.5	0.1	85.0	0.1	91.0	0.2
`JAGUAR Muon`	72.5	0.1	**94.0**	0.1	72.0	0.1	91.5	0.2

Discussion. For OPT-1.3B with LoRA, `ZO-Muon` achieves the highest accuracy of $93.5\% \pm 0.1$, surpassing ZO-Adam's 92.3%, while `JAGUAR SignSGD` and `JAGUAR Muon` reach $92.5\% \pm 0.5$ and $92.5\% \pm 0.001$, respectively. In the FT setting, `JAGUAR SignSGD` excels with $94.0\% \pm 0.1$, significantly improving over ZO-SGD's 90.8%. However, `ZO-Muon` and `JAGUAR Muon` exhibit lower FT performance at $84.0\% \pm 0.1$ and $70.0\% \pm 0.1$, suggesting challenges in scaling to full parameter updates. For RoBERTa-Large, `JAGUAR SignSGD` achieves $93.5\% \pm 0.2$ in FT and $92.0\% \pm 0.4$ in LoRA, consistently outperforming most baselines, while `ZO-Muon` and `JAGUAR Muon` maintain competitive results.

4.3 Memory Efficiency

Table 2 compares peak memory usage for OPT-1.3B, highlighting the efficiency of our methods.

Table 2: Peak memory usage (GB) for OPT-1.3B on SST2 with FT and LoRA.

Method	FT Memory	LoRA Memory
FO-SGD	148	24.29
ZO-SGD	64	13.22
ZO-Adam	158	15.43
`JAGUAR SignSGD`	65	3.94
`ZO-Muon`	66	3.98
`JAGUAR Muon`	65	3.99

Our methods consume approximately 65-66 GB in FT and 13.5-13.6 GB in LoRA, closely aligning with ZO-SGD's 64 GB and 13.22 GB, respectively, while FO-SGD requires 148 GB and 24.29 GB. This demonstrates that our approaches effectively balance accuracy gains with memory efficiency.

5 Discussion

TODO

6 Conclusion

TODO

References

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

[2] Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning (ICML)*, 2017.

[3] Jeremy Bernstein and Laker Newhouse. Modular duality in deep learning. *arXiv preprint arXiv:2410.21265*, 2024.

[4] Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

[5] Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd with majority vote is communication efficient and fault tolerant. *arXiv preprint arXiv:1810.05291*, 2018.

[6] HanQin Cai, Yuchen Lou, Daniel McKenzie, and Wotao Yin. A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization. In *ICML*, pages 1182–1191, 2021.

[7] Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David Cox. Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization. *Advances in neural information processing systems*, 32, 2019.

[8] Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David Cox. Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization. *arXiv preprint arXiv:1910.06513*, 2019.

[9] Yiming Chen, Yuan Zhang, Liyuan Cao, Kun Yuan, and Zaiwen Wen. Enhancing zeroth-order fine-tuning for language models with low-rank structures. *ArXiv*, abs/2410.07698, 2024.

[10] George E Dahl, Frank Schneider, Zachary Nado, Naman Agarwal, Chandramouli Shama Sastry, Philipp Hennig, Sourabh Medapati, Runa Eschenhagen, Priya Kasimbeg, Daniel Suo, et al. Benchmarking neural network training algorithms. *arXiv preprint arXiv:2306.07179*, 2023.

[11] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

[12] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

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

[14] Pavel Dvurechensky, Eduard Gorbunov, and Alexander Gasnikov. An accelerated directional derivative method for smooth stochastic convex optimization. *European Journal of Operational Research*, 290(2):601–621, 2021.

[15] Anonymous Author et al. Mezo-a^{3}dam: Memory-efficient zeroth-order adam with adaptivity adjustments. *OpenReview, ICLR 2025*, 2024. Under review.

[16] Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in neural information processing systems*, 31, 2018.

[17] Abraham D. Flaxman, Adam Tauman Kalai, and H. Brendan McMahan. Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 385–394, 2005.

[18] Lei Gao, Amir Ziashahabi, Yue Niu, Salman Avestimehr, and Murali Annavaram. Enabling efficient on-device fine-tuning of llms using only inference engines. *arXiv preprint arXiv:2409.15520*, 2024.

[19] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM journal on optimization*, 23(4):2341–2368, 2013.

[20] Eduard Gorbunov, Pavel Dvurechensky, and Alexander Gasnikov. An accelerated method for derivative-free smooth stochastic convex optimization. *SIAM Journal on Optimization*, 32(2):1210–1238, 2022.

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

[22] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[23] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint*, abs/1510.00149, 2015.

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

[25] Feihu Huang, Shangqian Gao, Jian Pei, and Heng Huang. Accelerated zeroth-order and first-order momentum methods from mini to minimax optimization. *Journal of Machine Learning Research*, 23(36):1–70, 2022.

[26] Shuoran Jiang, Qingcai Chen, Youcheng Pan, Yang Xiang, Yukang Lin, Xiangping Wu, Chuanyi Liu, and Xiaobao Song. Zo-adamu optimizer: Adapting perturbation by the momentum and uncertainty in zeroth-order optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18363–18371, 2024.

[27] Richeng Jin, Yufan Huang, Xiaofan He, Huaiyu Dai, and Tianfu Wu. Stochastic-sign sgd for federated learning with theoretical guarantees. *arXiv preprint arXiv:2002.10940*, 2020.

[28] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Nikita Kornilov, Philip Zmushko, Andrei Semenov, Alexander Gasnikov, and Alexander Beznosikov. Sign operator for coping with heavy-tailed noise: High probability convergence bounds with extensions to distributed optimization and comparison oracle. *arXiv preprint arXiv:2502.07923*, 2025.

[31] David Kozak, Cesare Molinari, Lorenzo Rosasco, Luis Tenorio, and Silvia Villa. Zeroth order optimization with orthogonal random directions. *arXiv preprint*, abs/2107.03941, 2021.

[32] Bingrui Li, Jianfei Chen, and Jun Zhu. Memory efficient optimizers with 4-bit states. *Advances in Neural Information Processing Systems*, 36:15136–15171, 2023.

[33] Jiaxiang Li and Mingyi Hong. A note on the convergence of muon and further, 2025.

[34] Xiangru Lian, Huan Zhang, Cho-Jui Hsieh, Yijun Huang, and Ji Liu. A comprehensive linear speedup analysis for asynchronous stochastic parallel optimization from zeroth-order to first-order. *Advances in neural information processing systems*, 29, 2016.

[35] Changyue Liao, Mo Sun, Zihan Yang, Jun Xie, Kaiqi Chen, Binhang Yuan, Fei Wu, and Zeke Wang. Lohan: Low-cost high-performance framework to fine-tune 100b model on a consumer gpu. *arXiv preprint arXiv:2403.06504*, 2024.

[36] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025.

[37] Sijia Liu, Pin-Yu Chen, Xiangyi Chen, and Mingyi Hong. signsgd via zeroth-order oracle. In *International conference on learning representations*, 2019.

[38] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

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

[40] Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources. *arXiv preprint arXiv:2306.09782*, 2023.

[41] Bharath Malladi, Amir Aghazadeh, Haotian Tang, et al. Mezo: Memory-efficient zeroth-order optimization of large language models. *arXiv preprint*, abs/2305.18660, 2023.

[42] 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.

[43] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[44] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

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

[46] Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025.

[47] Yuxiang Qian and Yong Zhao. Zeroth-order proximal stochastic recursive momentum algorithm for nonconvex nonsmooth optimization. In *2023 International Conference on New Trends in Computational Intelligence (NTCI)*, volume 1, pages 419–423. IEEE, 2023.

[48] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[49] Marco Rando, Cesare Molinari, Silvia Villa, and Lorenzo Rosasco. Stochastic zeroth order descent with structured directions. *Computational Optimization and Applications*, pages 1–37, 2024.

[50] Tadipatri Uday Kiran Reddy and Mathukumalli Vidyasagar. Convergence of momentum-based heavy ball method with batch updating and/or approximate gradients. In *2023 Ninth Indian Control Conference (ICC)*, pages 182–187. IEEE, 2023.

[51] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *Journal of Machine Learning Research*, 17(75):1–25, 2016.

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

[53] Mher Safaryan and Peter Richtárik. Stochastic sign descent methods: New algorithms and better theory. In *International Conference on Machine Learning*, pages 9224–9234. PMLR, 2021.

[54] Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

[55] Ohad Shamir. On the complexity of bandit and derivative-free stochastic convex optimization. In *ICML*, pages 1001–1009, 2013.

[56] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[57] Anshumali Shrivastava, Ping Li, et al. Communication-efficient distributed optimization using compressed gradients. *arXiv preprint*, abs/1503.06889, 2015.

[58] 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. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[59] Tao Sun, Qingsong Wang, Dongsheng Li, and Bao Wang. Momentum ensures convergence of signsgd under weaker assumptions. In *International Conference on Machine Learning*, pages 33077–33099. PMLR, 2023.

[60] Andrey Veprikov, Alexander Bogdanov, Vladislav Minashkin, and Aleksandr Beznosikov. New aspects of black box conditional gradient: Variance reduction and one point feedback. *Chaos, Solitons & Fractals*, 189:115654, 2024.

[61] Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

[62] Fei Wang, Li Shen, Liang Ding, Chao Xue, Ye Liu, and Changxing Ding. Simultaneous computation and memory efficient zeroth-order optimizer for fine-tuning large language models. *arXiv preprint arXiv:2410.09823*, 2024.

[63] Junjie Yin, Jiahao Dong, Yingheng Wang, Christopher De Sa, and Volodymyr Kuleshov. Modulora: finetuning 2-bit llms on consumer gpus by integrating with modular quantizers. *arXiv preprint arXiv:2309.16119*, 2023.

[64] Ziming Yu, Pan Zhou, Sike Wang, Jia Li, and Hua Huang. Zeroth-order fine-tuning of llms in random subspaces, 2024.

[65] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, et al. Opt: Open pre-trained transformer language models, 2022.

[66] Yihua Zhang, Pingzhi Li, Junyuan Hong, Jiaxiang Li, Yimeng Zhang, Wenqing Zheng, Pin-Yu Chen, Jason D. Lee, Wotao Yin, Mingyi Hong, Zhangyang Wang, Sijia Liu, and Tianlong Chen. Revisiting zeroth-order optimization for memory-efficient llm fine-tuning: A benchmark, 2024.

[67] Ligeng Zhu, Lanxiang Hu, Ji Lin, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. Pockengine: Sparse and efficient fine-tuning in a pocket. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 1381–1394, 2023.

[68] Yujia Zhu, Yujing Zhang, Ziwei Zhang, et al. Efficient fine-tuning of language models via zeroth-order optimization. *arXiv preprint*, abs/2305.14395, 2023.

A Classical ZO Muon

A.1 Zero-Order Muon

The modification presented here reduces memory usage by eliminating the need to store or compute gradients, instead enabling efficient parameter updates through directional finite differences. By substituting the gradient with its zero-order estimate $G_t = \frac{\mathcal{L}(W_t+\tau E)-\mathcal{L}(W_t-\tau E)}{2\tau}E$, where E is sampled from the standard Normal distribution $\mathcal{N}_{st}$ (each value $E_{i,j} \sim \mathcal{N}(0,1)$), we adapt the Muon algorithm [28] into its zero-order form:

Algorithm 3 Zero-Order Muon (`ZO-Muon`)

```
1: Parameters: stepsize (learning rate) $\gamma$, gradient approximation parameter $\tau$, number of iterations $T$.
2: Initialization: choose $W_0 \in \mathbb{R}^{n\times d}$, set $\mathbf{B}_{-1} = \mathbf{0}$
3: for $t = 0, 1, 2, \ldots, T$ do
4:     Sample $E$ from $\mathcal{N}_{st}$
5:     Compute $\mathbf{G}_t = \frac{\mathcal{L}(W_t+\tau E)-\mathcal{L}(W_t-\tau E)}{2\tau}E$
6:     Compute $\mathbf{O}_t = \text{NewtonSchulz}(\mathbf{G}_t)$
7:     Set $W_{t+1} = W_t - \gamma \mathbf{O}_t$
8: end for
```

A.2 Memory-Efficient Zeroth-Order Muon

In this section, we consider a zero-order gradient estimate:

$$G = \frac{\mathcal{L}(W+\tau E)-\mathcal{L}(W-\tau E)}{2\tau}E, \tag{3}$$

where $E \in \mathbb{R}^{n\times d}$ is typically sampled from $RS_2(1)$.

Using the Newton-Schulz process [28], we approximate the gradient estimate with its SVD components, where $G = U\Sigma V^T$ yields:

$$\text{NewtonSchulz}(G) \approx UV^T. \tag{4}$$

From (3), we express E as:

$$E = \text{sign}(\mathcal{L}(W+\tau E)-\mathcal{L}(W-\tau E))U\left|\frac{2\tau}{\mathcal{L}(W+\tau E)-\mathcal{L}(W-\tau E)}\right|\Sigma V^T. \tag{5}$$

Given the SVD of $E = U_E\Sigma_E V_E^T$, equating with (5) gives:

$$\begin{aligned} U &= \text{sign}(\mathcal{L}(W+\tau E)-\mathcal{L}(W-\tau E))U_E, \\ \Sigma &= \left|\frac{2\tau}{\mathcal{L}(W+\tau E)-\mathcal{L}(W-\tau E)}\right|\Sigma_E, \\ V &= V_E. \end{aligned} \tag{6}$$

Substituting into (4), we obtain:

$$\text{NewtonSchulz}(G) \approx \text{sign}(\mathcal{L}(W+\tau E)-\mathcal{L}(W-\tau E))U_E V_E^T. \tag{7}$$

This approximation allows orthogonalization of G using the SVD of E without directly applying Newton-Schulz.

The resulting Matrix-Efficient Zero-Order Muon algorithm is:

Algorithm 4 Matrix-Efficient Zero-Order Muon

```
1: Parameters: stepsize (learning rate) $\gamma$, gradient approximation parameter $\tau$, number of iterations $T$.
2: Initialization: choose $W^0 \in \mathbb{R}^{n\times d}$
3: for $t = 0, 1, 2, \dots, T$ do
4:     Sample orthogonal $U_E \in \mathbb{R}^{n\times n}$, $V_E \in \mathbb{R}^{d\times d}$, $\Sigma_E \sim \mathcal{D} \in \mathbb{R}^{n\times d}$
5:     Set $E = U_E \Sigma_E V_E^T$
6:     Compute $\mathbf{O}_t = \text{sign}(\mathcal{L}(W^t + \tau E) - \mathcal{L}(W^t - \tau E)) U_E V_E^T$
7:     Set $W^{t+1} = W^t - \gamma \cdot \mathbf{O}_t$
8: end for
```

B Proofs for ZO Momentum SignSGD with JAGUAR (Algorithm 1)

B.1 Proof of Lemma 3.4

Proof. We start with applying one step recursion to the momentum form the Algorithm 1:

$$\begin{aligned}
\mathbb{E}\left[\left\|m^t - \nabla f(x^t)\right\|_2^2\right] &= \mathbb{E}\left[\left\|m^{t-1} - (1-\beta)\left\langle m^{t-1}, e^t\right\rangle e^t + (1-\beta)\widetilde{\nabla}_{i_t} f(x^t, \xi^t) - \nabla f(x^t)\right\|_2^2\right] \\
&= \mathbb{E}\Big[\Big\| \left\{I - (1-\beta)e^t(e^t)^T\right\} \underbrace{\left\{m^{t-1} - \nabla f(x^{t-1})\right\}}_{=:a^t} \\
&\quad + (1-\beta)e^t(e^t)^T \underbrace{\left\{\widetilde{\nabla} f(x^t, \xi^t) - \nabla f(x^t)\right\}}_{=:b^t} \\
&\quad - \left\{I - (1-\beta)e^t(e^t)^T\right\} \underbrace{\left\{\nabla f(x^t) - \nabla f(x^{t-1})\right\}}_{=:c^t} \Big\|_2^2\Big],
\end{aligned} \tag{8}$$

where we used a notation $\widetilde{\nabla} f(x, \xi) := \sum_{i=1}^d \frac{\hat{f}(x+\tau e^i, \xi) - \hat{f}(x - \tau e^i, \xi)}{2\tau} e^i$, and e^i is the one-hot vector with 1 in the i-th coordinate. In equation (8) we also used the classical notation of the identity matrix $I \in \mathbb{R}^{d\times d}$.

Now using axillary notations a^t, b^t, c^t from equation (8) we divide it into six parts:

$$\begin{aligned}
\mathbb{E}\left[\left\|a^{t+1}\right\|_2^2\right] &= \underbrace{\mathbb{E}\left[\left\|\left\{I - (1-\beta)e^t(e^t)^T\right\} a^t\right\|_2^2\right]}_{①} \\
&+ \underbrace{\mathbb{E}\left[\left\|(1-\beta)e^t(e^t)^T b^t\right\|_2^2\right]}_{②} \\
&+ \underbrace{\mathbb{E}\left[\left\|\left\{I - (1-\beta)e^t(e^t)^T\right\} c^t\right\|_2^2\right]}_{③} \\
&+ \underbrace{\mathbb{E}\left[2\left\langle\left\{I - (1-\beta)e^t(e^t)^T\right\} a^t, (1-\beta)e^t(e^t)^T b^t\right\rangle\right]}_{④} \\
&+ \underbrace{\mathbb{E}\left[2\left\langle\left\{I - (1-\beta)e^t(e^t)^T\right\} a^t, \left\{I - (1-\beta)e^t(e^t)^T\right\} c^t\right\rangle\right]}_{⑤} \\
&+ \underbrace{\mathbb{E}\left[2\left\langle(1-\beta)e^t(e^t)^T b^t, \left\{I - (1-\beta)e^t(e^t)^T\right\} c^t\right\rangle\right]}_{⑥}.
\end{aligned} \tag{9}$$

Consider ①. Since i_t from Algorithm 1 is generated independent and uniform and $\{m^{s-1}, x^s\}_{s=0}^t$ do not depend on i_t, we can apply tower property:

$$① = \mathbb{E}\left[\left\|\left\{I - (1-\beta)e^t(e^t)^T\right\} a^t\right\|_2^2\right]$$

$$= \mathbb{E}\left[(a^t)^T \left\{I - (1-\beta)e^t(e^t)^T\right\}^T \left\{I - (1-\beta)e^t(e^t)^T\right\} a^t\right]$$
$$= \mathbb{E}\left[(a^t)^T \left\{I - (1-\beta)(2-(1-\beta))e^t(e^t)^T\right\} a^t\right]$$
$$= \mathbb{E}\left[(a^t)^T \cdot \mathbb{E}_{i_t\sim U[1;d]}\left[I - (1-\beta^2)e^t(e^t)^T\right] \cdot a^t\right]$$
$$= \mathbb{E}\left[(a^t)^T \cdot \left(1 - \frac{1-\beta^2}{d}\right) I \cdot a^t\right] = \left(1 - \frac{1-\beta^2}{d}\right)\mathbb{E}\left[\left\|a^t\right\|_2^2\right]. \tag{10}$$

Here we used the fact that $\left(e^t(e^t)^T\right)^T e^t(e^t)^T = e^t(e^t)^T$ and $\mathbb{E}_{i_t\sim U[1;d]}\left[e^t(e^t)^T\right] = \frac{1}{d}I$.

Similarly to equation (10), we can estimate ② and ③:

$$② = \mathbb{E}\left[\left\|(1-\beta)e^t(e^t)^T b^t\right\|_2^2\right] = \frac{(1-\beta)^2}{d}\mathbb{E}\left[\left\|b^t\right\|^2\right],$$
$$③ = \mathbb{E}\left[\left\|\left\{I - (1-\beta)e^t(e^t)^T\right\} c^t\right\|_2^2\right] = \left(1 - \frac{1-\beta^2}{d}\right)\mathbb{E}\left[\left\|c^t\right\|^2\right].$$

Since $b^t = \widetilde{\nabla} f(x^t, \xi^t) - \nabla f(x^t)$, we can use Lemma 4 from [60] with $\sigma_f = 0, \sigma_\nabla = \sigma$ and obtain the result of the form:

$$② \le \frac{(1-\beta)^2}{d} \cdot \left(dL^2\tau^2 + 2d\sigma^2 + \frac{2d\Delta^2}{\tau^2}\right), \tag{11}$$

where L, σ and Δ come from Assumptions 3.1, 3.2 and 3.3.

Since $c^t = \nabla f(x^t) - \nabla f(x^{t-1})$, we can use Assumption 3.1 and obtain:

$$③ \le \left(1 - \frac{1-\beta^2}{d}\right) L^2 \left\|x^t - x^{t-1}\right\|_2^2 = \left(1 - \frac{1-\beta^2}{d}\right) L^2 \left\|\text{sign}(m^t)\right\|_2^2$$
$$= \left(1 - \frac{1-\beta^2}{d}\right) dL^2\gamma^2 \le dL^2\gamma^2. \tag{12}$$

Consider ④. Let us move all matrixes to the left side of the dot product:

$$④ = \mathbb{E}\left[2\left\langle(1-\beta)\left\{I - (1-\beta)e^t(e^t)^T\right\} e^t(e^t)^T \cdot a^t, b^t\right\rangle\right]$$
$$= \mathbb{E}\left[2\left\langle(1-\beta)\beta e^t(e^t)^T \cdot a^t, b^t\right\rangle\right].$$

Now we use tower property for i_t as we did for ① − ③ and use the definitions of a^t and b^t:

$$④ = \frac{(1-\beta)\beta}{d} \cdot \mathbb{E}\left[2\left\langle a^t, b^t\right\rangle\right]$$
$$= \frac{(1-\beta)\beta}{d} \cdot \mathbb{E}\left[2\left\langle m^{t-1} - \nabla f(x^{t-1}), \widetilde{\nabla} f(x^t, \xi^t) - \nabla f(x^t)\right\rangle\right].$$

We now again use tower property, but with stochastic variable ξ^t. Since $\{m^{s-1}, x^s\}_{s=0}^t$ do not depend on ξ^t, we can obtain that:

$$④ = \frac{(1-\beta)\beta}{d} \cdot \mathbb{E}\left[2\left\langle m^{t-1} - \nabla f(x^{t-1}), \mathbb{E}_{\xi^t}\left[\widetilde{\nabla} f(x^t, \xi^t)\right] - \nabla f(x^t)\right\rangle\right]$$
$$\le \frac{(1-\beta)\beta}{2d} \cdot \mathbb{E}\left[\left\|m^{t-1} - \nabla f(x^{t-1})\right\|_2^2\right] \tag{13}$$
$$+ \frac{2(1-\beta)\beta}{d} \cdot \mathbb{E}\left[\left\|\mathbb{E}_{\xi^t}\left[\widetilde{\nabla} f(x^t, \xi^t)\right] - \nabla f(x^t)\right\|_2^2\right].$$

For the inequality (13) we use Fenchel-Young inequality. For estimating $\left\|\mathbb{E}_{\xi^t}\left[\widetilde{\nabla} f(x^t, \xi^t)\right] - \nabla f(x^t)\right\|_2^2$ we again can use Lemma 4 from [60] but now with $\sigma_\nabla = \sigma_f = 0$ since we have no randomness in $\mathbb{E}_{\xi^t}\left[\widetilde{\nabla} f(x^t, \xi^t)\right]$. Therefore ④ is bounded as:

$$④ \le \frac{(1-\beta)\beta}{2d} \cdot \mathbb{E}\left[\left\|a^t\right\|_2^2\right] + \frac{2(1-\beta)\beta}{d} \cdot \left(dL^2\tau^2 + \frac{2d\Delta^2}{\tau^2}\right). \tag{14}$$

Consider ⑤. Similar to ④ we can obtain:

$$\begin{aligned}
\textcircled{5} &= \mathbb{E}\left[2\left\langle \left\{I-(1-\beta)e^t(e^t)^T\right\}a^t, \left\{I-(1-\beta)e^t(e^t)^T\right\}c^t\right\rangle\right] \\
&= \mathbb{E}\left[2\left\langle \left\{I-(1-\beta^2)e^t(e^t)^T\right\}a^t, c^t\right\rangle\right] \\
&= \left(1-\frac{1-\beta^2}{d}\right)\cdot \mathbb{E}\left[2\left\langle a^t, c^t\right\rangle\right] \\
&\leq \left(1-\frac{1-\beta^2}{d}\right)\cdot \frac{1-\beta}{2d}\cdot \mathbb{E}\left[\left\|a^t\right\|_2^2\right] + \left(1-\frac{1-\beta^2}{d}\right)\cdot\frac{2d}{1-\beta}\cdot\mathbb{E}\left[\left\|c^t\right\|_2^2\right] \\
&\leq \frac{1-\beta}{2d}\cdot\mathbb{E}\left[\left\|a^t\right\|_2^2\right] + \frac{2d}{1-\beta}\cdot dL^2\gamma^2. \qquad (15)
\end{aligned}$$

Finally, we estimate ⑥ in the same way:

$$\begin{aligned}
\textcircled{6} &= \mathbb{E}\left[2\left\langle (1-\beta)e^t(e^t)^Tb^t, \left\{I-(1-\beta)e^t(e^t)^T\right\}c^t\right\rangle\right] \\
&= \mathbb{E}\left[2\left\langle (1-\beta)\beta e^t(e^t)^Tb^t, c^t\right\rangle\right] \\
&= \frac{(1-\beta)\beta}{d}\cdot\mathbb{E}\left[2\left\langle b^t, c^t\right\rangle\right] \\
&\leq \frac{(1-\beta)\beta}{d}\cdot\mathbb{E}\left[\left\|\mathbb{E}_{\xi^t}\left[\widetilde{\nabla} f(x^t,\xi^t)\right]-\nabla f(x^t)\right\|_2^2\right] + \frac{(1-\beta)\beta}{d}\cdot\mathbb{E}\left[\left\|c^t\right\|_2^2\right] \\
&\leq \frac{(1-\beta)\beta}{d}\cdot\left(dL^2\tau^2+\frac{2d\Delta^2}{\tau^2}\right) + \frac{(1-\beta)\beta}{d}\cdot dL^2\gamma^2. \qquad (16)
\end{aligned}$$

We made it! Now let us combine equations (10), (11), (12), (14), (15) and (16) to bound $\mathbb{E}\left[\left\|a^{t+1}\right\|_2^2\right]$ from equation (9):

$$\begin{aligned}
\mathbb{E}\left[\left\|a^{t+1}\right\|_2^2\right] \leq& \left(1-\frac{1-\beta}{d}\left[\underbrace{1+\beta}_{(10)}-\underbrace{\frac{\beta}{2}}_{(14)}-\underbrace{\frac{1}{2}}_{(15)}\right]\right)\cdot\mathbb{E}\left[\left\|a^t\right\|_2^2\right] \\
&+\frac{1-\beta}{d}\left(\underbrace{1-\beta}_{(11)}+\underbrace{2\beta}_{(14)}+\underbrace{\beta}_{(16)}\right)\cdot\left(dL^2\tau^2+\frac{2d\Delta^2}{\tau^2}\right)+\underbrace{\frac{(1-\beta)^2}{d}}_{(11)}\cdot 2d\sigma^2 \\
&+\left(\underbrace{1}_{(12)}+\underbrace{\frac{2d}{1-\beta}}_{(15)}+\underbrace{\frac{(1-\beta)\beta}{d}}_{(16)}\right)\cdot dL^2\gamma^2 \\
\leq& \left(1-\frac{1-\beta^2}{2d}\right)\cdot\mathbb{E}\left[\left\|a^t\right\|_2^2\right] \\
&+3\frac{1-\beta}{d}\cdot\left(dL^2\tau^2+\frac{2d\Delta^2}{\tau^2}\right)+2\frac{(1-\beta)^2}{d}\cdot d\sigma^2+\frac{4d}{1-\beta}\cdot dL^2\gamma^2.
\end{aligned}$$

By unrolling the recursion in the last inequality we obtain:

$$\begin{aligned}
\mathbb{E}\left[\left\|m^t-\nabla f(x^t)\right\|_2^2\right] \leq& 8\frac{d^2}{(1-\beta)(1-\beta^2)}\cdot dL^2\gamma^2+4\frac{(1-\beta)^2}{1-\beta^2}\cdot d\sigma^2 \\
&+6\frac{1-\beta}{1-\beta^2}\cdot\left(dL^2\tau^2+\frac{2d\Delta^2}{\tau^2}\right)+\left(\frac{1-\beta^2}{2d}\right)^t\left\|\nabla f(x^0)\right\|_2^2 \\
=& \mathcal{O}\left[\frac{d^3}{(1-\beta)^2}L^2\gamma^2+(1-\beta)d\sigma^2+dL^2\tau^2+\frac{2d\Delta^2}{\tau^2}\right. \\
&\left.+\left(1-\frac{1-\beta}{2d}\right)^t\left\|\nabla f(x^0)\right\|_2^2\right].
\end{aligned}$$

This finishes the proof. □

B.2 Proof of Theorem 3.5

Proof. We start from using Lemma 1 from [59]. For the points x^t, generated by Algorithm 1 it holds that:

$$f(x^{t+1}) - f(x^t) \leq -\gamma \left\|\nabla f(x^t)\right\|_1 + 2\sqrt{d}\gamma \left\|m^t - \nabla f(x^t)\right\|_2 + \frac{dL\gamma^2}{2}. \tag{17}$$

Now we take mathematical expectation of the both sides of the inequality (17) and use the results from Lemma 3.4:

$$\begin{aligned}
\mathbb{E}\left[f(x^{t+1})\right] - \mathbb{E}\left[f(x^t)\right] &\leq -\gamma\mathbb{E}\left[\left\|\nabla f(x^t)\right\|_1\right] + 2\sqrt{d}\gamma\mathbb{E}\left[\left\|m^t - \nabla f(x^t)\right\|_2\right] + \frac{dL\gamma^2}{2} \\
&= -\gamma\mathbb{E}\left[\left\|\nabla f(x^t)\right\|_1\right] + \mathcal{O}\Bigg[\frac{d^2}{1-\beta}\cdot L\gamma^2 + \sqrt{1-\beta}d\gamma\sigma + d\gamma L\tau \\
&\quad + \frac{d\gamma\Delta}{\tau} + \sqrt{d}\gamma\left(1 - \frac{1-\beta}{d}\right)^{t/2}\left\|\nabla f(x^0)\right\|_2\Bigg] + \frac{dL\gamma^2}{2}.
\end{aligned}$$

Consequently, after summing all T steps, we obtain:

$$\begin{aligned}
\gamma\sum_{t=0}^{T}\mathbb{E}\left[\left\|\nabla f(x^t)\right\|_1\right] &= \mathcal{O}\Bigg[f(x^0) - f(x^T) + T\cdot\left(\frac{d^2}{1-\beta}\cdot L\gamma^2 + \sqrt{1-\beta}d\gamma\sigma + d\gamma L\tau\right) \\
&\quad + T\cdot\frac{d\gamma\Delta}{\tau} + \sqrt{d}\gamma\sum_{t=0}^{T}\left(1 - \frac{1-\beta}{d}\right)^{t/2}\left\|\nabla f(x^0)\right\|_2\Bigg].
\end{aligned} \tag{18}$$

Now, we divide equation (18) by γT from both sides and obtain:

$$\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\left[\left\|\nabla f(x^t)\right\|_1\right] = \mathcal{O}\left[\frac{\delta_0}{\gamma T} + \frac{d\left\|\nabla f(x^0)\right\|_2}{T\sqrt{1-\beta}} + \frac{d^2L\gamma}{1-\beta} + \sqrt{1-\beta}d\sigma + dL\tau + \frac{d\Delta}{\tau}\right],$$

where we used a notation $\delta_0 := f(x^0) - f^*$. This finishes the proof. □

C Experimental setup

C.1 Evaluation procedure

Hyperparameter Tuning. To ensure optimal performance, we conducted a grid search over key hyperparameters for each method:

- Momentum parameter: $\beta \in \{10^{-3}, 10^{-2}\}$,
- Learning rate: $\gamma \in [10^{-6}, 10^{-1}]$,
- Smoothing parameter: $\tau \in \{10^{-1}, 10^{-2}, 10^{-3}\}$.

Additional fixed parameters include a momentum of 0.9 and an epsilon of 10^{-3} for numerical stability. The best-performing hyperparameters for each algorithm are detailed in the supplementary material, with notable settings including $\tau = 10^{-3}$, $\beta = 10^{-2}$ across all methods, and method-specific learning rates (e.g., $\gamma = 10^{-3}$ for `JAGUAR SignSGD` in LoRA). HERE MIGHT BE A GITHUB LINK

Evaluation Metrics. We assess performance using:

- **Test Accuracy:** Measured as the percentage of correct predictions on the SST2 test set, reflecting model effectiveness.
- **Peak Memory Usage:** Quantified in gigabytes (GB) during training, indicating memory efficiency.

Implementation Details. Experiments were conducted with three independent runs per configuration, each with a randomly selected seed fixed at the start to ensure reproducibility. We report the mean and standard deviation of test accuracy. Following [42], we employed half-precision (F16) training for ZO methods and mixed-precision (FP16) training for FO methods to optimize memory usage. Training was performed on a single NVIDIA A100 GPU, with memory profiling conducted using standard PyTorch utilities.

C.2 Experimental Methodology

Our experimental procedure was designed to rigorously evaluate the proposed methods under controlled conditions. For each combination of dataset (SST2), model (OPT-1.3B, RoBERTa-Large), fine-tuning scheme (FT, LoRA), and optimization method, we executed the following steps:

1. **Initialization:** Loaded the pre-trained model and initialized trainable parameters (all for FT, LoRA-specific for LoRA).
2. **Hyperparameter Selection:** Performed a preliminary parameter search to identify the best hyperparameters per method, iterating over the specified ranges and selecting based on validation accuracy.
3. **Evaluation:** Computed test accuracy on the SST2 test set after each run, averaging results across three runs with different seeds.
4. **Memory Profiling:** Recorded peak memory usage during training, ensuring consistency by maintaining identical hardware settings.

This methodology ensures a fair comparison across methods, capturing both performance and resource utilization comprehensively.