ConMeZO: Adaptive Descent-Direction Sampling for Gradient-Free Finetuning of Large Language Models

Lejs Deen Behric **Liang Zhang** **Bingcong Li** **Kiran Koshy Thekumparampil**
Department of Computer Science, ETH Zurich
Amazon AGI Labs
{lejs.behric, liang.zhang, bingcong.li}@inf.ethz.ch
kkt@amazon.com

Abstract

Zeroth-order or derivative-free optimization (MeZO) is an attractive strategy for finetuning large language models (LLMs) because it eliminates the memory overhead of backpropagation. However, it converges slowly due to the inherent curse of dimensionality when searching for descent directions in the high-dimensional parameter space of billion-scale LLMs. We propose ConMeZO, a novel zeroth-order optimizer that accelerates convergence by adaptive directional sampling. Instead of drawing the direction uniformly at random, ConMeZO restricts the sampling to a cone centered around a momentum estimate. This concentrates the search in directions where the true gradient is more likely to lie and thus reduces the effect of high dimensions. We prove that ConMeZO achieves the same worst-case convergence rate as MeZO. Empirically, when finetuning LLMs on natural language tasks, ConMeZO is up to 2× faster than MeZO while retaining the low-memory footprint of zeroth-order methods.

1 INTRODUCTION

Finetuning LLMs enables pre-trained models such as LLaMA (Touvron et al., 2023a,b; Grattafiori et al., 2024) and Gemma (Team et al., 2024a,b, 2025) to excel in diverse tasks. By adapting to specific applications, finetuning enhances model capabilities without requiring training from scratch, making state-of-the-art solutions more accessible. However, traditional finetuning methods face significant challenges due to their high computational and memory demands. These backpropagation-based approaches require substantial GPU resources for storing activation values and computing gradients, often exceeding the budgets when only consumer-grade GPUs are available.

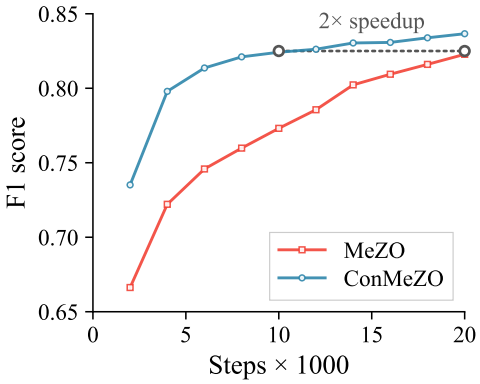


Figure 1: ConMeZO requires 2× fewer training iterations than MeZO to reach the same accuracy when finetuning OPT-1.3B on the SQuAD dataset.

Zeroth-order optimization (ZO) methods, such as those employed by MeZO (Malladi et al., 2023), offer a promising alternative. By relying only on forward passes to estimate gradients, ZO methods bypass the memory-intensive backward pass, facilitating finetuning in resource-constrained scenarios. Despite their advantages, ZO methods suffer from high variance in gradient estimates, leading to slower convergence compared to first-order methods. As shown in (Malladi et al., 2023, Table 15), while it takes approximately 1K iterations with Adam to finetune a RoBERTa-large model to desirable accuracy, MeZO requires significantly more steps, specifically 100K, to achieve comparable performance. As a result, the overall runtime of MeZO can be significantly longer than that of Adam.

This work aims to address the runtime inefficiency of ZO methods while preserving their memory benefits. Traditional ZO methods typically rely on random

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

search directions sampled from either a sphere or Gaussian distribution. Such random strategies, especially in the high-dimensional regime, result in high variance in gradient estimations and thereby slow convergence. We propose reducing gradient variance by constraining random search directions within a cone centered on a promising search direction, defined by a momentum vector. This strategy improves convergence by narrowing the search space while maintaining the flexibility of ZO optimization. The proposed approach, coined ConMeZO, significantly reduces iteration counts while retaining the memory efficiency, and matches MeZO in wall-clock runtime. Combining theoretical analysis and empirical validation, we contribute to advancing efficient and accessible finetuning methods for LLMs.

Contributions: This work presents a new ZO algorithm, built on an innovative geometrical concept. The algorithm reduces the high variance in gradient estimation by constraining perturbations to a cone centered around a momentum direction. This novel approach balances exploration and exploitation, leading to faster convergence and more reliable ZO optimization. Our contributions can be summarized as:

1. **Algorithm design and implementation:** A distinctive cone-sampling strategy inspired by geometrical principles focuses search directions toward areas more likely to yield productive updates. This approach not only reduces noise but also preserves the simplicity of ZO optimization, making it both efficient and theoretically sound. ConMeZO further introduces a vectorized implementation that performs perturbations and updates in fused in-place operations over a flattened parameter buffer, avoiding costly Python loops and random tensor generation. This design yields significant wall-clock speedups without altering the underlying algorithmic logic.
2. **Theoretical analysis:** Unlike traditional ZO optimizers whose convergence rates suffer from the curse of dimensionality, we show that moderately aligning the momentum to the true gradient can provide up to $O(d)$ speedup over MeZO.
3. **Improved practical performance:** Experiments on finetuning LLMs demonstrate faster convergence of ConMeZO, especially in early iterations. ConMeZO ultimately achieves up to $2\times$ speedup over MeZO.

1.1 Related Work

Zeroth-order (ZO) optimization. The work of Nesterov and Spokoiny (2017) marks a foundational step in formally analyzing the convergence rate of zeroth-order methods, such as zeroth-order (stochastic) gradient descent (ZO-SGD) that substitutes gradients in SGD with their zeroth-order estimators. Building on this foundation, Shamir (2017) refine the analysis for nonsmooth convex functions, while Lin et al. (2022) extend these insights to nonsmooth nonconvex functions. Contributions by Ghadimi and Lan (2013) further tackle smooth functions in stochastic settings. These works have shown that for smooth problems, the squared norm of the gradient converges with a worst-case rate of $O(d/T)$ where d is the number of dimensions (Nesterov and Spokoiny, 2017). In stark contrast, standard gradient descent has a rate of $O(1/T)$ (Nesterov, 2003). Further, there exist lower complexity results that prove such dimension dependence is unavoidable (Jamieson et al., 2012; Wibisono et al., 2012; Duchi et al., 2015; Golovin et al., 2020; Alabdulkareem and Honorio, 2021) unless there are additional structural assumptions such as sparsity or low-rank Hessian (Wang et al., 2018b; Yue et al., 2023). More recently, Zhang et al. (2025) prove that zeroth-order methods converge to flat minima for convex and sufficiently smooth functions.

These studies are motivated by the growing interest in zeroth-order methods, driven by practical challenges including the memory limitations imposed by fast differentiation techniques (Wang et al., 2018b; Liu et al., 2020). ZO has been enriched with various enhancements such as conditional gradient methods (Balasubramanian and Ghadimi, 2018) and variance reduction techniques (Liu et al., 2018; Fang et al., 2018; Ji et al., 2019). Other notable adaptations include the integration of SignSGD (Liu et al., 2019a) and applications to minimax optimization (Wang et al., 2022). Beyond algorithmic development, these methods have demonstrated utility across diverse domains, including black-box machine learning (Grill et al., 2015; Chen et al., 2017, 2019), bandit optimization (Flaxman et al., 2005; Shamir, 2017), reinforcement learning (Salimans et al., 2017; Choromanski et al., 2018; Mania et al., 2018), and distributed learning, where they mitigate communication overhead (Fang et al., 2022; Zelikman et al., 2023; Xu et al., 2024).

ZO for LLM finetuning. In the realm of ZO optimization for LLMs, various approaches have emerged, emphasizing memory efficiency and computational effectiveness. MeZO (Malladi et al., 2023) offers a breakthrough by eliminating backpropagation and significantly reducing memory requirements, but it suffers from slower convergence rates and sensitivity to high-dimensional noise. Zhang et al. (2024b) provide a more comprehensive benchmark for evaluating the performance of ZO for LLM finetuning, where they observe that directly combining ZO with momentum methods

does not lead to significant performance gain. Liu et al. (2024) introduce sparse MeZO, where only a carefully chosen subset of parameters is updated. LeZO (Wang et al., 2024) introduces a layer-wise sparse strategy to reduce computational overhead. Gautam et al. (2024) integrate variance reduction to ZO optimizers and propose MeZO-SVRG. Zhao et al. (2025) also use the estimation of second-order information with ZO oracles to improve the performance of MeZO. Similarly, LOZO (Chen et al., 2025) incorporates low-rank gradient estimations, capturing the inherent low-dimensional structure of LLM gradients. The work of Park et al. (2025) further develops a theoretical framework to characterize effectiveness of structural perturbations, such as sparsity and low rankness, in ZO approaches. It is also pointed out in Ma and Huang (2025) that effective perturbations in ZO should account for the (estimated) gradient directions, and they propose an approach that requires halving the minibatch data. DPZero (Zhang et al., 2024a) extends ZO optimization into the realm of differential privacy, addressing the dual challenges of memory efficiency and data privacy in finetuning LLMs. Addax (Li et al., 2025) strategically combines first-order and ZO steps to improve overall efficiency.

Notation We use $\|\cdot\|$ for the Euclidean norm and $\langle\cdot,\cdot\rangle$ for the standard inner product in $\mathbb{R}^d$. Let $\mathbb{S}^{d-1} = \{x \in \mathbb{R}^d \mid \|x\| = 1\}$ be the unit sphere in $\mathbb{R}^d$ and $r\,\mathbb{S}^{d-1}$ the sphere of radius $r > 0$. A function $f : \mathbb{R}^d \to \mathbb{R}$ is ℓ-smooth iff it is differentiable and $\|\nabla f(x_1) - \nabla f(x_2)\| \le \ell\|x_1 - x_2\|$, $\forall x_1, x_2 \in \mathbb{R}^d$, where $\nabla f(x)$ is the gradient at x. The orthogonal complement of a vector $x \in \mathbb{R}^d$, denoted $(x)^\perp$, is the maximal subspace of $\mathbb{R}^d$ orthogonal to x, i.e., $(x)^\perp = \{v \in \mathbb{R}^d \mid \langle x, v\rangle = 0\}$. $\mathcal{N}(0, I_d)$ and $\mathcal{U}(\mathcal{S})$ denote the standard d-dimensional Gaussian and the uniform distribution over a set $\mathcal{S}$, respectively.

2 ZEROTH-ORDER OPTIMIZATION

We consider solving the following problem with *zeroth-order optimization* (ZO),

$$\min_{x\in\mathbb{R}^d} f(x)\,, \tag{1}$$

where a direct access to the gradient is unavailable. Instead, we assume access to only a ZO oracle which can compute the objective value at any given point (Nesterov and Spokoiny, 2017). This setting arises when gradients are challenging or costly to compute (Liu et al., 2020). The ZO problem is usually solved by applying gradient descent (GD) using a ZO gradient estimator, like the Simultaneous Perturbation Stochastic Approximation (SPSA) (Spall, 1992), computed via function value evaluations at perturbed points. We use the following popular stochastic ZO estimator which perturbs the point along randomly sampled directions (Nesterov and Spokoiny, 2017; Duchi et al., 2015).

Definition 1. *The stochastic ZO gradient estimate (ZOGE) of f at x using z sampled randomly from an isotropic distribution like $\mathcal{N}(0, I_d)$ or $\mathcal{U}(\sqrt{d}\,\mathbb{S}^{d-1})$ and a smoothing parameter $\lambda > 0$ is given by*

$$g_\lambda(x, z) = \frac{f(x + \lambda z) - f(x - \lambda z)}{2\lambda} z. \tag{2}$$

The ZOGE is highly efficient because it requires only two evaluations of $f(x)$ and avoids explicit gradient computation, thus reducing memory and compute usage for non-trivial f. Further, it is known that $\lim_{\lambda\to 0} g_\lambda(x, z)$ is an unbiased estimator of the gradient.

Lemma 1. *(Zhang et al., 2024a) When f is differentiable and λ is sufficiently small, $g_\lambda(x, z) \approx (z^\top \nabla f(x))z$. Further, the first two moments of this term satisfy*

$$\mathbb{E}_z[(z^\top \nabla f(x))z] = \nabla f(x) \quad \textit{and} \tag{3}$$

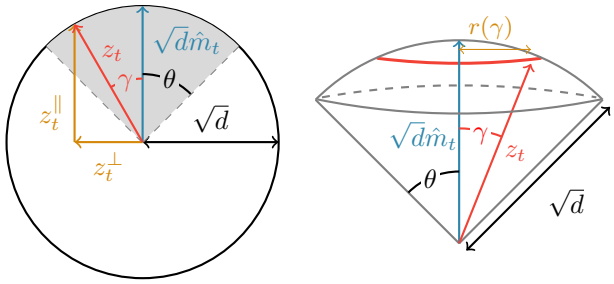
$$\mathbb{E}_z[\|(z^\top \nabla f(x))z\|^2] \le 2d\|\nabla f(x)\|^2. \tag{4}$$

Therefore, despite the benefits mentioned above, this gradient estimator suffers from high $O(d)$ variance, especially in high-dimensional settings of LLM finetuning. It can be shown that the worst case per-step objective function decrease of GD using this estimator is upper-bounded by $O(\|\nabla f(x)\|^2/d)$ (Nesterov and Spokoiny, 2017). This leads to $O(d)$ slower convergence speed than first-order methods which have $\Omega(\|\nabla f(x)\|^2)$ objective decrease. Addressing this limitation is crucial for making ZO optimization competitive in practical scenarios (Malladi et al., 2023). In the next sections, we mitigate this high variance and slow convergence of ZO methods by constraining the search direction z around the true gradient direction estimated via a momentum, while still retaining the memory advantage of ZO methods over first-order approaches.

3 CONMEZO: ZO WITH CONE SAMPLING

To address the slow convergence of ZO methods due to the high variance of the ZOGE (Definition 1 and Lemma 1), we propose a novel cone-based sampling strategy for the search direction z that builds upon the following two components:

1. **Promising Search Direction.** A momentum vector m accumulates past gradient estimates, serving as a coarse predictor of future productive search directions; and



(a) Gray search angle in 2D

(b) Sampling a z_t in the red ring in the cone.

Figure 2: 2D- and 3D-representation of the cone-sampling approach. (a) Sphere with radius $\sqrt{d}$ and (gray) search space cone of half-angle θ around promising search direction $\hat{m}_t$. We can set random direction $z_t = z_t^{\parallel} + z_t^{\perp}$ with angle γ to $\hat{m}_t$. (b) 3D representation of cone sampling in red area.

2. **Cone Restriction.** The perturbation direction z is then constrained to a cone with an apex at the origin, central axis along m, and a half-angle θ, reducing the variance of the ZOGE.

The proposed approach can be seamlessly incorporated into ZO methods which use the vanilla ZOGE. In Section 4, we show that cone sampling reduces the variance of the gradient estimate by focusing the search in regions more likely to yield productive parameter updates, thereby striking a more effective balance between exploration and exploitation. The next sections formalize the cone-sampling approach and provide an approximate implementation for high-dimensional problems.

3.1 Momentum-based Search Direction

We iteratively construct the promising search direction using a momentum m_t defined as the exponentially moving average of past gradient estimates:

$$m_{t+1} \leftarrow \beta \cdot m_t + (1-\beta) \cdot g(x_t, z_t) \tag{5}$$

where $\beta \in [0,1]$. At each new step, the current gradient estimate $g(x_t, z_t)$ is given a weight $(1-\beta)$, which serves as a tunable hyperparameter. We adopt a momentum mechanism since it is well known to reduce variance of stochastic GD, particularly in training nonconvex neural networks (Tieleman and Hinton, 2012; Kingma and Ba, 2014; Cutkosky and Orabona, 2019).

3.2 Sampling from a Cone

In this subsection, we discuss how the perturbation direction z_t is sampled uniformly from the intersection

Algorithm 1 Cone-based memory-efficient zeroth-order (ConMeZO) optimization algorithm.

Require: Parameters $x \in \mathbb{R}^d$, function $f : \mathbb{R}^d \to \mathbb{R}$, cone angle $\theta \in [0, \frac{\pi}{2}]$, momentum parameter $\beta \in [0,1]$, iterations T, learning rate η, smoothing parameter $\lambda > 0$.
for $t = 0, \dots, T-1$ **do**
  $u_t \sim \mathcal{U}(\mathbb{S}^{d-1})$ ▷ Sample u_t
  $[m_0 \leftarrow u_0]_{t=0}$
  $z_t \leftarrow \sqrt{d}(\cos(\theta) \cdot m_t/\|m_t\| + \sin(\theta) \cdot u_t)$
  $x \leftarrow x - \eta \cdot g_\lambda(x, z_t)$
  $m_{t+1} \leftarrow \beta \cdot m_t + (1-\beta) \cdot g_\lambda(x, z_t)$
end for

of the sphere $\sqrt{d}\,\mathbb{S}^{d-1}$ and the cone with a central axis along the direction m_t and half-angle θ. Using the 2D Figure 2a as a geometrical reference, we decompose z_t into two additive components:

1. $z_t^{\parallel}$, the component of z_t parallel to m_t; and,
2. $z_t^{\perp}$, the component of z_t orthogonal to m_t,

so that $z_t = z_t^{\parallel} + z_t^{\perp}$. For a fixed angle γ between z_t and m_t, it follows that

$$z_t^{\parallel} = \sqrt{d}\cos(\gamma) \cdot \hat{m}_t \quad \text{and} \quad z_t^{\perp} = \sqrt{d}\sin(\gamma) \cdot u_t^{\perp}, \tag{6}$$

where $\hat{m}_t = m_t/\|m_t\|$ and $u_t^{\perp}$ is uniformly sampled from the intersection of the unit sphere $\mathbb{S}^{d-1}$ and the subspace $(m_t)^{\perp}$ orthogonal to m_t. Since $u_t^{\perp} \perp m_t$, we can directly verify that $\|z_t\|^2 = d$, and thus $z_t \in \sqrt{d}\,\mathbb{S}^{d-1}$. Next, we discuss how to sample $u_t^{\perp}$ and introduce randomness in the angle γ. To this end, we adopt two practical simplifications that we justify in the high-dimensional regime.

Sampling of orthogonal $u_t^{\perp}$. Instead of sampling $u_t^{\perp}$ from $\mathbb{S}^{d-1} \cap (m_t)^{\perp}$, we sample it uniformly from $\mathbb{S}^{d-1}$. This simplification is justified as in high dimensions ($d \gg 1$), a random vector is nearly orthogonal to any fixed direction. We provide a formal result below whose proof is in Appendix A.1.

Proposition 1. *The cosine similarity between a randomly sampled vector $u_t \sim \mathcal{U}(\mathbb{S}^{d-1})$ and a fixed unit vector $\hat{m}_t$ becomes negligible in high dimensions, i.e., $\langle \hat{m}_t, u_t \rangle \to 0$ as $d \to \infty$.*

Sampling of angle γ. To ensure that z_t is uniformly distributed in the intersection of the sphere and the cone, one must sample γ appropriately. However, due to concentration phenomena in high dimensions, most of the probability mass of the distribution of γ lies sharply near the edge of the cone. Thus, in practice,

it is suitable to set $\gamma = \theta$. We formalize this intuition below, with the proof given in Appendix A.2.

Proposition 2. *Consider a cone $\mathcal{C}$ with apex at the origin, central axis aligned with a unit vector $\hat{m}_t$, and half-angle θ. If a random vector z_t is sampled uniformly from $\mathcal{C} \cap \sqrt{d}\,\mathbb{S}^{d-1}$, then the angle γ between z_t and $\hat{m}_t$ converges in distribution to a Dirac delta at θ, i.e., $\Pr(\gamma \le \theta') \to 0$ as $d \to \infty$, $\forall\ \theta' < \theta$.*

The construction of the random direction z_t can be then summarized as follows:

$$z_t \leftarrow \sqrt{d}\Big(\cos(\theta)\cdot\hat{m}_t + \sin(\theta)\cdot u_t\Big), \text{ where } u_t \sim \mathcal{U}(\mathbb{S}^{d-1}).$$

Note that even though the originally proposed sampling method is conceptually complicated, we have designed a simple and practical approximate implementation for it. We use the ZOGE (Definition 1) with this newly constructed search direction z_t as our gradient estimate. The resultant ConMeZO method is given in Algorithm 1. The impact of hyperparameters θ, β and the learning rate η will be analyzed theoretically in Section 4 and empirically in Section 5.

3.3 Speedup Due to Extra Memory Buffer

Notice that Algorithm 1 maintains an extra optimizer state to store the momentum m_t. In our PyTorch implementation, we utilize this extra memory buffer to make ConMeZO faster than MeZO even though the pseudocode of ConMeZO in Algorithm 1 has 2× the number of update steps. To keep the peak GPU VRAM usage small, the MeZO implementation perturbs the iterate one parameter at a time across the neural network and regenerates the random perturbation z_t four times per iteration. However, our implementation perturbs the iterate in a single vectorized operation and regenerates u_t only twice because it can temporarily store the perturbation value in the momentum buffer. Note that it is impossible to speed up MeZO similarly unless we increase its memory usage to match that of ConMeZO. Table 3 confirms that our implementation of ConMeZO is faster than that of MeZO. More details and sample code are provided in Appendix B.

3.4 Momentum Warm-up

Since the initial perturbation directions are random and not necessarily informative, they can bias the trajectory of the ConMeZO optimizer for many future steps. To address this, we gradually increase the momentum parameter β during the early phase of training. This allows the momentum to be shaped by multiple directions before it becomes dominant, leading to more stable early iterations and more reliable progress later on. In practice, we use the following warm-up schedule for a training run of 20K steps, which we find to work well:

$$\beta_t = \begin{cases} 0.1, & 0 \le t \le 200, \\ \beta_{\text{final}} - \dfrac{\beta_{\text{final}} - 0.1}{\left(1 + 8 \cdot \left(\frac{t-200}{1800}\right)^{1.8}\right)^3}, & 200 < t \le 2000, \\ \beta_{\text{final}}, & t > 2000, \end{cases}$$

For shorter training runs of 10K steps, we simply halve the interval lengths (i.e., 0–100, 100–1000, and beyond). This schedule smoothly transitions from a small initial value to the final momentum parameter, stabilizing early optimization without additional overhead. The schedule is designed to have three phases: (i) a short flat start to avoid immediate momentum bias, (ii) a smooth ease-in ramp to gradually build momentum, and (iii) a quick saturation to the final value. This shape is chosen empirically and gives stable early optimization and consistent performance across tasks. Detailed intuition, schedule visualization (Figure 8), and ablation results (Table 14) are provided in Appendix C.5.

4 THEORETICAL ANALYSIS

In this section, we formally analyze ConMeZO (Algorithm 1) for optimization problems with ℓ-smooth and potentially nonconvex objectives. Our analysis shows that at each iteration, ConMeZO can decrease the objective value faster than the standard ZO method, MeZO (Malladi et al., 2023), provided that the momentum is well aligned with the true gradient. Furthermore, the convergence rate of ConMeZO is no worse than that of MeZO. We begin our analysis by characterizing the first and second moments of the cone-based gradient estimator used in ConMeZO (Algorithm 1). For ease of exposition, we assume that the "smoothing" parameter λ of the ZOGE (Definition 1) estimator is infinitesimally small, i.e., $\lambda \to 0$.

Lemma 2. *Let $a_t = \nabla f(x_t)$, ρ_t be the angle between a_t and m_t, and $z_t = \sqrt{d}(\cos(\theta)\cdot\hat{m}_t + \sin(\theta)\cdot u_t)$, where $u_t \sim \mathcal{U}(\mathbb{S}^{d-1})$. When $\lambda \to 0$, the ZOGE becomes $(z_t^\top a_t)z_t$, and its first and second moments satisfy:*

$$\mathbb{E}_{u_t}[(z_t^\top a_t)z_t] = d\cos^2\theta\cdot(\hat{m}_t^\top a_t)\hat{m}_t + \sin^2\theta\cdot a_t\,, \text{ and}$$
$$\mathbb{E}_{u_t}\left[\|(z_t^\top a_t)z_t\|^2\right] \le d\|a_t\|^2\big((d+4)\cos^2\theta\cos^2\rho_t + \sin^2\theta\big)\,.$$

A proof can be found in Appendix A.3. Comparing these results with the moments of the vanilla ZOGE in Lemma 1, we see that the cone-sampled gradient estimator is biased toward the momentum direction $\hat{m}_t$, and the second moment has an extra $O(\cos^2\rho_t)$ term. The bias vanishes and the second moment becomes similar to that in Lemma 1 when $\hat{m}_t$ aligns

with the true gradient a_t, i.e., $\rho_t \approx 0$. The parameter θ modulates the tradeoff between the momentum-aligned component and the true gradient component.

4.1 Convergence Guarantee

Next, we characterize the expected improvement per iteration and the global convergence of ConMeZO. Using Lemma 2, we get the following "descent lemma" whose proof is in Appendix A.4.

Theorem 1 (Descent Lemma)**.** *Assume f is ℓ-smooth and let $\lambda \to 0$. The expected value of $f(x_{t+1})$ satisfies*

$$\begin{aligned}\mathbb{E}_{u_t}[f(x_{t+1})] &\le f(x_t) \\ &- \eta\left(d\cos^2(\theta)\cos^2(\rho_t) + \sin^2(\theta)\right)\|a_t\|^2 \\ &+ \frac{\eta^2\ell}{2} d\left((d+4)\cos^2(\theta)\cos^2(\rho_t) + \sin^2(\theta)\right)\|a_t\|^2.\end{aligned}$$

Further, with the choice of $\eta = (d\cos^2(\theta)\cos^2(\rho_t) + \sin^2(\theta))/(\ell d\left((d+4)\cos^2(\theta)\cos^2(\rho_t) + \sin^2(\theta)\right))$, we can get that

$$\begin{aligned}&\mathbb{E}_{u_t}[f(x_{t+1})] - f(x_t) \\ &\le -\frac{\left(d\cos^2(\theta)\cos^2(\rho_t) + \sin^2(\theta)\right)^2}{2\ell d\left((d+4)\cos^2(\theta)\cos^2(\rho_t) + \sin^2(\theta)\right)}\|a_t\|^2 \\ &\approx -\frac{\left(d\cos^2(\theta)\cos^2(\rho_t) + \sin^2(\theta)\right)}{2\ell d}\|a_t\|^2.\end{aligned}$$

The above theorem shows that ConMeZO achieves a per-step decrease on the order of $O(\|a_t\|^2)$, provided that the momentum $\hat{m}_t$ is effectively aligned with the gradient a_t. Specifically, when the alignment is strong (i.e., $\cos^2(\rho_t)$ is large) and $\sin(\theta)$ is small, ConMeZO can significantly outperform standard MeZO, which is restricted to a $O(\|a_t\|^2/d)$ descent (Nesterov and Spokoiny, 2017). Thus, the progress of ConMeZO aligns directly with how well the momentum tracks the gradient direction. We empirically verify this alignment between the momentum vector and the true gradient in Appendix C.4 (Figure 6).

Trade-off in θ. The cone angle θ balances the exploitation of the promising momentum direction and the exploration of alternative directions through random sampling. At step t, the maximally exploitative choice for θ is

$$\theta^\star = \begin{cases} 0, & \text{if } \cos^2(\rho_t) > (d+4)/d^2, \\ \frac{\pi}{2}, & \text{otherwise.} \end{cases} \tag{7}$$

Since the expected squared cosine similarity between two random vectors is $1/d \approx (d+4)/d^2$, this result indicates that momentum should be exploited precisely when it provides a better-than-random direction, i.e., when $\cos^2(\rho_t) > 1/d$. However, this binary prescription is extreme and does not account for the global convergence as θ affects the quality of future momentum vectors. A small θ reduces exploration, while a large θ sacrifices exploitation. In practice, it is better to use a balanced choice of θ, which maintains some directional bias while still allowing sufficient exploration. Empirical observations in Section C.4 confirm this trade-off. We now establish an overall convergence guarantee of our algorithm over T iterations.

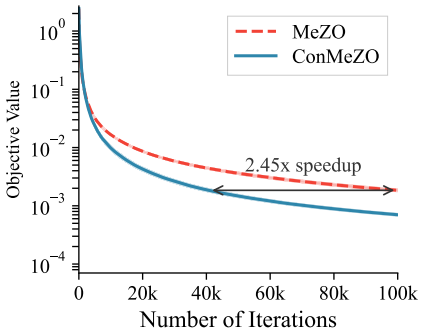


Figure 3: **Synthetic Optimization Problem:** ConMeZO achieves a 2.45× speedup over MeZO on the synthetic quadratic problem.

Corollary 1. *Assume f is ℓ-smooth and bounded from below, i.e., $f^\star = \min_{x\in\mathbb{R}^d} f(x) > -\infty$ and let $\lambda \to 0$. There exists a hyperparameter setting for ConMeZO such that after $T \ge 1$ steps, the average expected squared gradient norm over T iterations satisfies*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[\|\nabla f(x_t)\|^2\right] \le \frac{2\ell d(f(x_0) - f^\star)}{T}. \tag{8}$$

A proof directly follows from Theorem 1 by setting $\theta = \pi/2$ and telescoping across T iterations. This establishes that ConMeZO matches the worst-case convergence rate of MeZO, while Theorem 1 shows it can be significantly faster whenever momentum aligns better than chance with the gradient ($\cos^2(\rho_t) \gg 1/d$).

5 EXPERIMENTAL RESULTS

In this section, we study the empirical performance and efficiency of ConMeZO on a synthetic problem and LLM finetuning tasks with RoBERTa (Liu et al., 2019b) and OPT (Zhang et al., 2022) backbones. Our code can be found at `https://github.com/LejsDeen/ConMeZO`.

5.1 Synthetic Optimization Problem

First, we compare MeZO and ConMeZO on a synthetic strongly convex quadratic problem in $d = 1000$

Table 1: **RoBERTa-Large:** ConMeZO achieves better test metrics (%) than MeZO, and MeZO+Momentum (Mom.) when finetuning RoBERTa-large.

		ZO Methods:		
Task	AdamW	MeZO	Mom.	ConMeZO
SST-2	93.1	92.8	92.2	**93.5**
SST-5	56.6	49.3	**51.9**	48.9
SNLI	86.4	81.0	80.9	**81.9**
MNLI	81.4	69.7	70.7	**73.2**
RTE	83.6	73.9	74.0	**75.1**
TREC	95.9	88.4	89.2	**90.0**
Average	82.8	75.8	76.5	**77.1**

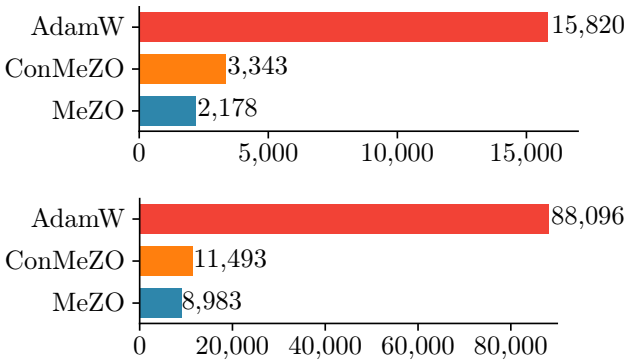


Figure 4: Peak GPU memory usage (MiB) increase of ConMeZO over MeZO is negligible when compared to the memory usage of first-order methods like AdamW: **Top:** RoBERTa-Large on SST2 (batch size 64). **Bottom:** OPT-1.3B on BoolQ (batch size 16).

dimensions with a condition number of value d. We tune the hyperparameters of both the methods over a simple grid with the final average objective value over 5 trials serving as selection criteria. In Figure 3, we plot the mean objective value (and standard error) of the best hyperparameter settings for each method. We observe that even in this simple synthetic problem, ConMeZO achieves a 2.45× speedup over MeZO. See Appendix C.1 for more details.

5.2 Finetuning of RoBERTa-large

We finetune RoBERTa-large (Liu et al., 2019b), a language model with 355 million parameters. Following Malladi et al. (2023), the finetuning process tackles a few-shot setting on a suite of six standard NLP tasks from the GLUE benchmark (Wang et al., 2018a). We compare ConMeZO with first-order AdamW, MeZO, and the MeZO+Momentum baseline. MeZO+Momentum is a novel baseline that we design, and it maintains a momentum m_t similar to ConMeZO, but instead of using it to bias the ZO perturbation z_t, its momentum replaces $g(x_t, z_t)$ as the iterate update direction. We set the smoothing parameter as $\lambda = 10^{-3}$ for all ZO methods and report the test metrics after tuning the hyperparameters for 10K iterations. See Appendix C.2 for more details. In Table 1, ConMeZO achieves the best average performance across all tasks except SST-5, where MeZO+Momentum performs best. While MeZO+Momentum outperforms vanilla MeZO on average, it does not consistently match ConMeZO, suggesting that naively incorporating momentum into ZO methods is suboptimal. We report standard errors (Table 10) along with intermediate metrics (Table 11) and a comparison to first-order SGD (Table 9) in Appendix C.2. Additional analyses, including test metric curves (Figure 7) and an ablation study of parameters β and θ (Figure 5), can be found in Appendix C.4.

5.3 Finetuning of OPT models

Here, we finetune OPT-1.3B and OPT-13B (Zhang et al., 2022) on eight standard benchmarks using ConMeZO and MeZO. We omit AdamW due to GPU VRAM limitations, and MeZO+Momentum since it did not achieve the best performance on RoBERTa. We set the smoothing parameter as $\lambda = 10^{-3}$ for both methods and report the test metrics after tuning the hyperparameters for 20K iterations. See Appendix C.3 for more details. In Table 2, we see that ConMeZO delivers the highest final accuracy/F1 on almost every task and achieves the best average across the tasks. The advantages of ConMeZO we observed in RoBERTa carry over seamlessly to the larger OPT-1.3B and OPT-13B models. Note that we omit the DROP dataset for OPT-13B because we obtained an out-of-memory error when running both MeZO and ConMeZO on our GPU. We also plot learning curves of ConMeZO and MeZO when finetuning OPT-1.3B on the SQuAD dataset in Figure 1. The steep slope highlights ConMeZO's accelerated convergence: it reaches MeZO's 20K step performance in less than 10K steps, which yields a 2× speedup. Standard errors and full task breakdowns are reported in Appendix C.3 (Table 12 for OPT-1.3B and Table 13 for OPT-13B).

5.4 Compute and Memory Efficiency

Compared to MeZO, ConMeZO achieves a slightly lower runtime per iteration while incurring a moderate increase in memory consumption. For instance, on RoBERTa-large with SST-2, ConMeZO requires `3343 MiB` versus MeZO's `2178 MiB`, a relative increase of +54%. Importantly, this overhead becomes much less

Table 2: **OPT-1.3B and OPT-13B**: ConMeZO achieves better average test accuracy / F1 score (%) than MeZO when averaged over 3 seeds. OOM means out of memory.

Model	Method	SQuAD	SST2	WIC	BoolQ	DROP	ReCoRD	RTE	MultiRC	Avg.
OPT-1.3B	MeZO	72.76	88.49	56.53	63.50	25.90	70.67	**56.92**	**55.90**	61.33
	ConMeZO	**75.34**	**90.56**	**58.15**	**64.20**	**26.53**	70.67	55.48	53.50	**61.80**
OPT-13B	MeZO	82.28	91.25	**58.93**	67.73	OOM	**81.17**	63.66	57.53	71.79
	ConMeZO	**83.66**	**92.39**	58.31	**69.33**	OOM	80.87	**64.50**	**58.30**	**72.48**

Table 3: ConMeZO achieves better average wall-clock time (s) per step across all the tasks than MeZO when finetuning RoBERTa-large and OPT-1.3B.

	RoBERTa-Large							OPT-1.3B				
Method	SST2	SST5	SNLI	MNLI	RTE	TREC	Avg.	SST2	BoolQ	DROP	SQuAD	Avg.
MeZO	0.193	0.202	0.198	0.335	0.548	0.146	0.270	0.096	0.270	0.426	0.194	0.246
ConMeZO	**0.180**	**0.195**	**0.189**	**0.325**	**0.538**	**0.144**	**0.262**	**0.083**	**0.257**	**0.413**	**0.181**	**0.233**
% Speedup	6.95%	3.59%	4.89%	3.15%	1.89%	1.10%	3.60%	15.73%	5.10%	3.33%	7.29%	7.86%

pronounced at larger context lengths, which is consistent with current trends in LLM training, where the relative gap shrinks significantly. As shown in Figure 4 and Table 8, Appendix B, the increase is only around +10% on OPT-1.3B with DROP and +28% with BoolQ, while still staying within the same order of magnitude as MeZO. In contrast, first-order optimizers such as AdamW demand dramatically more resources (e.g., `15820 MiB` for finetuning RoBERTa-Large on SST-2; see Figure 4). The runtime comparison in Table 3 further highlights that ConMeZO not only avoids extra cost but is in fact faster per iteration, despite its seemingly higher algorithmic complexity. On average, ConMeZO improves iteration speed by 3.6% on RoBERTa-Large and 7.9% on OPT-1.3B. This speedup comes from the extra memory buffer as discussed in Section 3.

6 COMPARISON WITH RECENT ZO METHODS

We compare ConMeZO with recent ZO finetuning methods to clarify the relationship and demonstrate empirical performance. These methods target different aspects of ZO optimization and are largely orthogonal to ConMeZO's cone-guided directional sampling.

6.1 HiZOO: Second-Order Information

HiZOO (Zhao et al., 2025) augments MeZO with second-order curvature information via three function evaluations per step, estimating local Hessian structure to refine update directions. In principle, ConMeZO's cone-guided sampling could be combined with HiZOO's curvature-informed estimator.

Test accuracy. For RoBERTa-Large, we evaluated both methods for 10k steps (yielding ∼2× longer wall-clock time for HiZOO), while for OPT-1.3B we used equal wall-clock budgets (HiZOO 10K steps, ConMeZO 20K steps). For HiZOO, we use settings recommended in their paper; ConMeZO uses our default configuration (Appendices C.2 and C.3). See results in Table 4.

Table 4: ConMeZO maintains a clear advantage across the majority of tasks and both models. For HiZOO, we report the best performance across learning rates $\{10^{-5}, 10^{-6}, 10^{-7}\}$ selected individually for each task.

RoBERTa-Large (5 seeds)			OPT-1.3B (3 seeds)			
Method	SST-2	RTE	SST-2	BoolQ	WiC	Avg.
HiZOO	83.4	54.2	88.5	63.0	**59.0**	69.6
ConMeZO	**93.5**	**75.1**	**90.6**	**64.2**	58.2	**76.3**

HiZOO performs competitively on OPT-1.3B but exhibits sensitivity to hyperparameters on RoBERTa-Large. ConMeZO maintains robust performance across tasks with fixed hyperparameters.

Wall-clock time and VRAM. We evaluate both methods on OPT-1.3B with identical hardware and batch size for 2K steps. ConMeZO is 2–2.25× faster than HiZOO across tasks (∼4 vs ∼9 minutes on SST-2, ∼18 vs ∼36 minutes on BoolQ), due to HiZOO's higher per-step overhead. VRAM usage is comparable: ConMeZO peaks at `11493` MiB while HiZOO uses `10681` MiB on OPT-1.3B/SST-2.

6.2 LOZO: Low-Rank Perturbations

LOZO (Chen et al., 2025) incorporates low-rank structure into gradient estimations, restricting updates to a low-dimensional adapter subspace. In principle, one could apply ConMeZO's cone-guided ZO estimator within LOZO's low-rank adapter space.

We compare LOZO, its momentum variant LOZO-M, and ConMeZO on RoBERTa-Large under equal wall-clock time. LOZO has ∼20% higher per-step runtime than ConMeZO (0.022s vs 0.018s per step on SST-2). Following the authors' recommendations, we sweep learning rates {1e-6, 1e-7}, rank $\{1, 2\}$, and update interval $\nu \in \{50, 100\}$. ConMeZO uses default settings. We provide a comparison in Table 5.

Table 5: Accuracy comparison on RoBERTa-Large. Under equal wall-clock time, ConMeZO achieves superior average performance compared to LOZO & LOZO-M.

Method	SST2	SST5	SNLI	MNLI	RTE	TREC	Avg
LOZO	92.3	49.4	**82.5**	70.5	**78.7**	89.4	77.1
LOZO-M	92.8	**50.4**	81.3	69.5	75.1	89.8	76.5
ConMeZO	**93.2**	50.0	82.0	**73.3**	76.2	**90.4**	**77.5**

6.3 MeZO-SVRG: Variance Reduction

MeZO-SVRG (Gautam et al., 2024) reduces *data-induced* variance by periodically computing large-batch reference gradients. This targets a different variance source than ConMeZO's *directional* variance reduction, making the methods conceptually orthogonal and potentially combinable.

Importantly, the experiments in (Gautam et al., 2024) are conducted in the non-prompted fine-tuning setting, whereas our work focuses on the prompt-conditioned regime. To ensure a direct comparison under our target regime, we evaluate MeZO-SVRG in the same prompt-conditioned setting used throughout this paper.

We evaluate on RoBERTa-Large in the prompt-conditioned setting (3 seeds) in Table 6.

Table 6: Accuracy (%). MeZO-SVRG was run for 24K steps on both tasks, ConMeZO for 10K (SST-2) and 20K (MNLI) steps. ConMeZO matches or exceeds MeZO-SVRG's accuracy with fewer training steps.

Method	SST-2	MNLI
MeZO-SVRG	92	75
ConMeZO	**93.5**	**76.4**

Wall-clock time and VRAM. MeZO-SVRG computes a full-batch ZO gradient every other iteration, requiring ∼16 minutes per 100 steps versus ConMeZO's ∼1 minute. VRAM usage is similar: `3719` MiB for MeZO-SVRG vs `3343` MiB for ConMeZO (RoBERTa-Large, SST-2).

6.4 ZO-AdaMM: Adaptive Moments

Table 7: Accuracy (%) comparison on SST-2. ConMeZO consistently outperforms ZO-AdaMM across model architectures. ZO-AdaMM results are as reported by Zhang et al. (2024b) and always use 20K steps; ConMeZO results use 10K steps for RoBERTa-Large and 20K steps for OPT-1.3B.

Method	RoBERTa-Large	OPT-1.3B
ZO-AdaMM	89.8	84.4
ConMeZO	**93.5**	**90.6**

ZO-AdaMM stores a second-moment buffer, increasing memory usage beyond ConMeZO. Conceptually, one could apply ZO-AdaMM's adaptive scaling on top of ConMeZO's cone-guided estimates, potentially improving robustness at the cost of additional memory.

7 CONCLUSION

This work explores the challenges and opportunities in finetuning LLMs using ZO optimization. By introducing a novel cone-sampling strategy, we propose ConMeZO that mitigates the high variance of traditional random-direction estimators and leverages momentum to guide updates more effectively. Empirical evaluations on RoBERTa-large and larger OPT-1.3B and OPT-13B models show that our method consistently outperforms state-of-the-art ZO optimizers like MeZO, achieving up to a 2× speedup in early convergence and absolute accuracy gains across benchmarks. While ConMeZO incurs a modest memory overhead compared to MeZO, this trade-off is practical for settings where gradient access is unavailable or expensive.

ConMeZO's directional sampling strategy is largely orthogonal to recent ZO methods which target complementary aspects. This suggests potential for hybrid approaches that combine ConMeZO's cone-guided sampling with these techniques. Future work can explore such combinations, as well as lightweight, self-adaptive mechanisms for adjusting the cone angle θ and momentum β dynamically during optimization. Finally, establishing theoretical guarantees that ConMeZO achieves strictly better convergence rates than MeZO under realistic assumptions remains a valuable direction.

Acknowledgements

L.Z. gratefully acknowledges funding by the Max Planck ETH Center for Learning Systems (CLS). B.L. is supported by SNSF Project Funding No. 200021-207343. This work does not relate to the current position of K.T. at Amazon.

References

Alabdulkareem, A. and Honorio, J. (2021). Information-theoretic lower bounds for zero-order stochastic gradient estimation. In *IEEE International Symposium on Information Theory (ISIT)*, pages 2316–2321. IEEE.

Balasubramanian, K. and Ghadimi, S. (2018). Zeroth-order (non)-convex stochastic optimization via conditional gradient and gradient updates. *Advances in Neural Information Processing Systems*, 31.

Bentivogli, L., Clark, P., Dagan, I., and Giampiccolo, D. (2009). The fifth PASCAL recognizing textual entailment challenge.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017). ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security*, pages 15–26.

Chen, X., Liu, S., Xu, K., Li, X., Lin, X., Hong, M., and Cox, D. (2019). ZO-AdaMM: Zeroth-order adaptive momentum method for black-box optimization. *Advances in Neural Information Processing Systems*, 32.

Chen, Y., Zhang, Y., Cao, L., Yuan, K., and Wen, Z. (2025). Enhancing zeroth-order fine-tuning for language models with low-rank structures. In *International Conference on Learning Representations*.

Choromanski, K., Rowland, M., Sindhwani, V., Turner, R., and Weller, A. (2018). Structured evolution with compact architectures for scalable policy optimization. In *International Conference on Machine Learning*, pages 970–978. PMLR.

Clark, C., Lee, K., Chang, M.-W., Kwiatkowski, T., Collins, M., and Toutanova, K. (2019). BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2924–2936.

Cutkosky, A. and Orabona, F. (2019). Momentum-based variance reduction in non-convex SGD. *Advances in Neural Information Processing Systems*, 32.

Dagan, I., Glickman, O., and Magnini, B. (2005). The PASCAL recognising textual entailment challenge.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. (2019). DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2368–2378.

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

Fang, C., Li, C. J., Lin, Z., and Zhang, T. (2018). SPIDER: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in Neural Information Processing Systems*, 31.

Fang, W., Yu, Z., Jiang, Y., Shi, Y., Jones, C. N., and Zhou, Y. (2022). Communication-efficient stochastic zeroth-order optimization for federated learning. *IEEE Transactions on Signal Processing*, 70:5058–5073.

Flaxman, A. D., Kalai, A. T., and McMahan, H. B. (2005). Online convex optimization in the bandit setting: Gradient descent without a gradient. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 385–394.

Gautam, T., Park, Y., Zhou, H., Raman, P., and Ha, W. (2024). Variance-reduced zeroth-order methods for fine-tuning language models. In *International Conference on Machine Learning*, pages 15180–15208. PMLR.

Ghadimi, S. and Lan, G. (2013). Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, W. B. (2007). The third PASCAL recognizing textual entailment challenge.

Golovin, D., Karro, J., Kochanski, G., Lee, C., Song, X., and Zhang, Q. (2020). Gradientless descent: High-dimensional zeroth-order optimization. In *International Conference on Learning Representations*.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. (2024). The LLaMA 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Grill, J.-B., Valko, M., and Munos, R. (2015). Black-box optimization of noisy functions with unknown smoothness. *Advances in Neural Information Processing Systems*, 28.

Haim, R. B., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. (2006). The second PASCAL recognising textual entailment challenge.

Jamieson, K. G., Nowak, R., and Recht, B. (2012). Query complexity of derivative-free optimization. *Advances in Neural Information Processing Systems*, 25.

Ji, K., Wang, Z., Zhou, Y., and Liang, Y. (2019). Improved zeroth-order variance reduced algorithms and analysis for nonconvex optimization. In *International Conference on Machine Learning*, pages 3100–3109. PMLR.

Khashabi, D., Chaturvedi, S., Roth, M., Upadhyay, S., and Roth, D. (2018). Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–262.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, Z., Zhang, X., Zhong, P., Deng, Y., Razaviyayn, M., and Mirrokni, V. (2025). Addax: Utilizing zeroth-order gradients to improve memory efficiency and performance of SGD for fine-tuning language models. In *International Conference on Learning Representations*.

Lin, T., Zheng, Z., and Jordan, M. (2022). Gradient-free methods for deterministic and stochastic nonsmooth nonconvex optimization. *Advances in Neural Information Processing Systems*, 35:26160–26175.

Liu, S., Chen, P.-Y., Chen, X., and Hong, M. (2019a). SignSGD via zeroth-order oracle. In *International Conference on Learning Representations*.

Liu, S., Chen, P.-Y., Kailkhura, B., Zhang, G., Hero III, A. O., and Varshney, P. K. (2020). A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Processing Magazine*, 37(5):43–54.

Liu, S., Kailkhura, B., Chen, P.-Y., Ting, P., Chang, S., and Amini, L. (2018). Zeroth-order stochastic variance reduction for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019b). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Liu, Y., Zhu, Z., Gong, C., Cheng, M., Hsieh, C.-J., and You, Y. (2024). Sparse MeZO: Less parameters for better performance in zeroth-order LLM fine-tuning. *arXiv preprint arXiv:2402.15751*.

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

Malladi, S., Gao, T., Nichani, E., Damian, A., Lee, J. D., Chen, D., and Arora, S. (2023). Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075.

Mania, H., Guy, A., and Recht, B. (2018). Simple random search of static linear policies is competitive for reinforcement learning. *Advances in Neural Information Processing Systems*, 31.

Nesterov, Y. (2003). *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

Nesterov, Y. and Spokoiny, V. (2017). Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527–566.

Park, S., Yun, J., Kim, S., Kundu, S., and Yang, E. (2025). Unraveling zeroth-order optimization through the lens of low-dimensional structured perturbations. *arXiv preprint arXiv:2501.19099*.

Pilehvar, M. T. and Camacho-Collados, J. (2019). WiC: The word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1267–1273.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Shamir, O. (2017). An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *Journal of Machine Learning Research*, 18(1):1703–1713.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recur-

sive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37(3):332–341.

Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., et al. (2025). Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. (2024a). Gemma: Open models based on Gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. (2024b). Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5 – RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023a). LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023b). LLAMA 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Voorhees, E. M. and Tice, D. M. (2000). Building a question answering test collection. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 200–207.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018a). GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Wang, F., Shen, L., Ding, L., Xue, C., Liu, Y., and Ding, C. (2024). Simultaneous computation and memory efficient zeroth-order optimizer for fine-tuning large language models. *arXiv preprint arXiv:2410.09823*.

Wang, Y., Du, S., Balakrishnan, S., and Singh, A. (2018b). Stochastic zeroth-order optimization in high dimensions. In *International Conference on Artificial Intelligence and Statistics*, pages 1356–1365. PMLR.

Wang, Z., Balasubramanian, K., Ma, S., and Razaviyayn, M. (2022). Zeroth-order algorithms for nonconvex–strongly-concave minimax problems with improved complexities. *Journal of Global Optimization*, pages 1–32.

Wibisono, A., Wainwright, M. J., Jordan, M., and Duchi, J. C. (2012). Finite sample convergence rates of zero-order stochastic optimization methods. *Advances in Neural Information Processing Systems*, 25.

Williams, A., Nangia, N., and Bowman, S. (2018). A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1112–1122.

Xu, M., Cai, D., Wu, Y., Li, X., and Wang, S. (2024). FwdLLM: Efficient federated finetuning of large language models with perturbed inferences. In *USENIX Annual Technical Conference*, pages 579–596.

Yue, P., Yang, L., Fang, C., and Lin, Z. (2023). Zeroth-order optimization with weak dimension dependency. In *Conference on Learning Theory*, pages 4429–4472. PMLR.

Zelikman, E., Huang, Q., Liang, P., Haber, N., and Goodman, N. D. (2023). Just one byte (per gradient): A note on low-bandwidth decentralized language model finetuning using shared randomness. *arXiv preprint arXiv:2306.10015*.

Zhang, L., Li, B., Thekumparampil, K. K., Oh, S., and He, N. (2024a). DPZero: Private fine-tuning of language models without backpropagation. In *International Conference on Machine Learning*, pages 59210–59246. PMLR.

Zhang, L., Li, B., Thekumparampil, K. K., Oh, S., Muehlebach, M., and He, N. (2025). Zeroth-order optimization finds flat minima. *arXiv preprint arXiv:2506.05454*.

Zhang, S., Liu, X., Liu, J., Gao, J., Duh, K., and Van Durme, B. (2018). ReCoRD: Bridging the gap between human and machine commonsense reading comprehension. *arXiv preprint arXiv:1810.12885*.

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. (2022). OPT: Open pre-trained Transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhang, Y., Li, P., Hong, J., Li, J., Zhang, Y., Zheng, W., Chen, P.-Y., Lee, J. D., Yin, W., Hong, M.,

et al. (2024b). Revisiting zeroth-order optimization for memory-efficient LLM fine-tuning: A benchmark. In *International Conference on Machine Learning*, pages 59173–59190. PMLR.

Zhao, Y., Dang, S., Ye, H., Dai, G., Qian, Y., and Tsang, I. (2025). Second-order fine-tuning without pain for LLMs: A Hessian informed zeroth-order optimizer. In *International Conference on Learning Representations*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Yes]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Yes]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

ConMeZO: Adaptive Descent-Direction Sampling for Gradient-Free Finetuning of Large Language Models: Supplementary Material

A PROOFS

A.1 Proof of Proposition 1

Let $u_t \sim \mathcal{U}(\mathbb{S}^{d-1})$ and $\hat{m}_t \in \mathbb{R}^d$ with $\|\hat{m}_t\| = 1$ be a promising search direction. Instead of ensuring that the sampled random direction is orthogonal to $\hat{m}_t$, we show that it suffices to sample any random direction $u_t \sim \mathcal{U}(\mathbb{S}^{d-1})$. We show that the relative magnitude of the projection $\langle \hat{m}_t, u_t \rangle$ becomes negligible as $d \to \infty$.

Proof. *Notice that u_t can be sampled as $u_t = \frac{X}{\|X\|}$, where $X \sim \mathcal{N}(0, I_d)$. We have that*

$$\langle \hat{m}_t, u_t \rangle = \frac{\langle \hat{m}_t, X \rangle}{\|X\|}.$$

$\langle \hat{m}_t, X \rangle$ is a $\mathcal{N}(0,1)$ random variable since $\|\hat{m}_t\| = 1$, and $\|X\|^2$ is a χ^2-distributed random variable with d degrees of freedom. Therefore, for large d, the ratio $\langle \hat{m}_t, X \rangle / \|X\|$ is on the order of $\mathcal{N}(0,1)/\sqrt{d}$, which converges to 0 in probability as $d \to \infty$.

A.2 Proof of Proposition 2

Consider a cone $\mathcal{C}$ in $\mathbb{R}^d$ with apex at the origin, central axis aligned with a unit vector $\hat{m}_t$, and half-angle $\theta \in [0, \pi/2]$. We are interested in the distribution of the angle γ between a random vector z_t, sampled uniformly from the intersection of $\mathcal{C}$ with $\sqrt{d}\,\mathbb{S}^{d-1}$, and the axis $\hat{m}_t$. The following proof demonstrates that as the dimension $d \to \infty$, the angle γ becomes concentrated at θ.

Proof. *We consider the case where $d \to \infty$:*

$$\begin{aligned} p(\gamma \le \theta') &= \frac{\int_0^{\theta'} \textit{ Surface area of hypercircle with radius } r(\alpha)\ d\alpha}{\int_0^{\theta} \textit{ Surface area of hypercircle with radius } r(\beta)\ d\beta} \\ &= \frac{\int_0^{\theta'} C_d \cdot r(\alpha)^{d-1}\ d\alpha}{\int_0^{\theta} C_d \cdot r(\beta)^{d-1}\ d\beta} \end{aligned}$$

where C_d is a constant dependent on d and independent of radius. When inspecting Figure 2b, it is simple to see that $r(\gamma) = \sqrt{d}\sin(\gamma)$. Now assume that $\theta' < \theta$. Further calculation yields

$$\begin{aligned} p(\gamma \le \theta') &= \frac{\int_0^{\theta'} (\sqrt{d}\sin(\alpha))^{d-1}\ d\alpha}{\int_0^{\theta} (\sqrt{d}\sin(\beta))^{d-1}\ d\beta} \\ &\le \frac{\theta'(\sin(\theta'))^{d-1}}{\int_0^{\theta} (\sin(\beta))^{d-1}\ d\beta} \end{aligned}$$

because $(\sin\alpha)^{d-1} \le (\sin\theta')^{d-1}$ for $0 \le \alpha \le \theta'$, and the interval length is θ'. Let $s = \frac{\theta+\theta'}{2} \in (\theta', \theta)$. Now because $\sin(\beta)$ is increasing on $[0, \theta]$, on the sub-interval $[s, \theta]$ we have $(\sin(\beta))^{d-1} \ge (\sin(s))^{d-1}$. Hence

$$\int_0^{\theta} (\sin(\beta))^{d-1}\, d\beta \ge \int_s^{\theta} (\sin(\beta))^{d-1}\, d\beta \ge (\theta - s)\,(\sin(s))^{d-1}.$$

We have that

$$\sin(s) > \sin(\theta'),$$

since $\theta > s > \theta'$. *Putting these together, we get*

$$\begin{aligned} p(\gamma \le \theta') &= \frac{\int_0^{\theta'} (\sin(\alpha))^{d-1}\, d\alpha}{\int_0^{\theta} (\sin(\beta))^{d-1}\, d\beta} \\ &\le \frac{\theta'}{\theta - s} \left(\frac{\sin(\theta')}{\sin(s)}\right)^{d-1}. \end{aligned}$$

Since $\sin(s) > \sin(\theta')$, *we have that*

$$p(\gamma \le \theta') \to 0 \text{ for } d \to \infty.$$

So instead of sampling γ, *in practice we can set* $\gamma = \theta$.

A.3 Proof of Lemma 2

Proof. *We have that* $z_t = \sqrt{d}\cos(\theta)\cdot \hat{m}_t + \sin(\theta)\cdot u_t$, *where* $u_t \sim \mathcal{U}(\sqrt{d}\,\mathbb{S}^{d-1})$.

$$\begin{aligned} \mathbb{E}_{u_t}\left[(z_t^\top a_t) z_t\right] &= \mathbb{E}_{u_t}\left[\left(\left(\cos(\theta)\sqrt{d}\cdot \hat{m}_t + \sin(\theta)\cdot u_t\right)^\top a_t\right)\left(\cos(\theta)\sqrt{d}\cdot \hat{m}_t + \sin(\theta)\cdot u_t\right)\right] \\ &= d\cos^2(\theta)(\hat{m}_t^\top a_t)\cdot \hat{m}_t + \sin^2(\theta)\cdot \mathbb{E}\left[(u_t^\top a_t) u_t\right] \\ &= d\cos^2(\theta)(\hat{m}_t^\top a_t)\cdot \hat{m}_t + \sin^2(\theta)\cdot a_t. \end{aligned}$$

Let $z_t = \alpha\hat{m}_t + \beta u_t$, *where* $u_t \sim \mathcal{U}(\sqrt{d}\ \mathbb{S}^{d-1})$, $\|\hat{m}_t\| = 1$, $\alpha = \cos(\theta)\cdot\sqrt{d}$ *and* $\beta = \sin(\theta)$. *We now derive the second moment of* $(z_t^\top a_t) z_t$.

$$\begin{aligned} & \mathbb{E}_{u_t}\left[\left\|(z_t^\top a_t) z_t\right\|^2\right] \\ = \ & \mathbb{E}_{u_t}\left[\left\|\left(\alpha(\hat{m}_t^\top a_t) + \beta(u_t^\top a_t)\right)(\alpha\hat{m}_t + \beta u_t)\right\|^2\right] \\ = \ & \mathbb{E}_{u_t}\left[\left(\alpha^2(\hat{m}_t^\top a_t)^2 + \beta^2(u_t^\top a_t)^2 + 2\alpha\beta(\hat{m}_t^\top a_t)(u_t^\top a_t)\right)\left(\alpha^2\|\hat{m}_t\|^2 + \beta^2\|u_t\|^2 + 2\alpha\beta(\hat{m}_t^\top u_t)\right)\right] \\ = \ & \alpha^2\left(\alpha^2(\hat{m}_t^\top a_t)^2 + \beta^2\|a_t\|^2 + 0\right) \\ & +\beta^2\left(\alpha^2(\hat{m}_t^\top a_t)^2 d + \beta^2 d\|a_t\|^2 + 0\right) \\ & +2\alpha\beta\left(0 + 0 + 2\alpha\beta(\hat{m}_t^\top a_t)^2\right) \\ = \ & d^2\cos^2(\theta)(\hat{m}_t^\top a_t)^2 + 4d\sin^2(\theta)\cos^2(\theta)(\hat{m}_t^\top a_t)^2 + d\sin^2(\theta)\|a_t\|^2 \\ = \ & d\cos^2(\theta)\left(d + 4\sin^2(\theta)\right)(\hat{m}_t^\top a_t)^2 + d\sin^2(\theta)\|a_t\|^2 \\ \le \ & d(d+4)\cos^2(\theta)(\hat{m}_t^\top a_t)^2 + d\sin^2(\theta)\|a_t\|^2. \end{aligned}$$

Using $(\hat{m}_t^\top a_t)^2 = (\cos(\rho)\|\hat{m}_t\|\|a_t\|)^2 = \cos^2(\rho)\cdot\|a_t\|^2$, *we have:*

$$\mathbb{E}_{u_t}\left[\left\|(z_t^\top a_t) z_t\right\|^2\right] \le d\left((d+4)\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)\|a_t\|^2.$$

A.4 Proof of Theorem 1

We assume that the function f is ℓ-smooth, meaning:

$$f(x_{t+1}) \le f(x_t) + \nabla f(x_t)^\top (x_{t+1} - x_t) + \frac{\ell}{2}\|x_{t+1} - x_t\|^2.$$

Additionally, we use the update rule $x_{t+1} = x_t - \eta g_\lambda(x_t, z_t)$, where $g_\lambda(x_t, z_t)$ is the gradient estimate at x_t and $z_t = \cos(\theta)\sqrt{d} \cdot \hat{m}_t + \sin(\theta) \cdot u_t$, where $u_t \sim \mathcal{U}(\sqrt{d}\,\mathbb{S}^{d-1})$. Let $a_t = \nabla f(x_t)$. The proof derives the expected improvement in $f(x)$ per iteration under these assumptions.

Proof. *Substituting the update rule $x_{t+1} = x_t - \eta g_\lambda(x_t, z_t)$ into the smoothness assumption, we have that*

$$f(x_{t+1}) \leq f(x_t) - \eta \nabla f(x_t)^\top g_\lambda(x_t, z_t) + \frac{\eta^2 \ell}{2} \|g_\lambda(x_t, z_t)\|^2.$$

Taking expectations with respect to u_t, the random search direction, we obtain

$$\mathbb{E}_{u_t}[f(x_{t+1})] \leq f(x_t) - \eta a_t^\top \mathbb{E}_{u_t}[g_\lambda(x_t, z_t)] + \frac{\eta^2 \ell}{2} \mathbb{E}_{u_t}[\|g_\lambda(x_t, z_t)\|^2].$$

Using the moments of $g_\lambda(x_t, z_t)$ with $\lambda \to 0$:

$$\mathbb{E}_{u_t}[(z_t^\top a_t) z_t] = d\cos^2(\theta)(\hat{m}_t^\top a_t) \cdot \hat{m}_t + \sin^2(\theta) \cdot a_t,$$

and

$$\mathbb{E}_{u_t}\left[\|(z_t^\top a_t) z_t\|^2\right] \leq d\left((d+4)\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)\|a_t\|^2,$$

we substitute these into the inequality and get that

$$\begin{aligned}\mathbb{E}_{u_t}[f(x_{t+1})] \leq f(x_t) &- \eta\left(d\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)\|a_t\|^2 \\ &+ \frac{\eta^2 \ell}{2} d\left((d+4)\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)\|a_t\|^2.\end{aligned}$$

This proves the first part of the theorem.

Next, rearranging the inequality to isolate $\|a_t\|^2$, we have that

$$\begin{aligned}\mathbb{E}_{u_t}[f(x_t) - f(x_{t+1})] \geq \Big(&\eta\left(d\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right) \\ &- \frac{\eta^2 \ell}{2} d\left((d+4)\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)\Big)\|a_t\|^2.\end{aligned}$$

Thus, it holds that

$$\|a_t\|^2 \leq \frac{\mathbb{E}_{u_t}[f(x_t) - f(x_{t+1})]}{\eta\left(d\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right) - \frac{\eta^2 \ell}{2} d\left((d+4)\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)}.$$

Observe that the denominator is a concave quadratic function of η and achieves its maximum at:

$$\eta^* = \frac{d\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)}{\ell d\left((d+4)\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)}.$$

Substituting $\eta = \eta^*$ and rearranging for $\mathbb{E}_{u_t}[f(x_{t+1})] - f(x_t)$ yields:

$$\begin{aligned}\mathbb{E}_{u_t}[f(x_{t+1})] - f(x_t) &\leq -\left(\eta^*\left(d\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right) - \frac{\eta^{*2}\ell}{2} d\left((d+4)\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)\right)\|a_t\|^2 \\ &= -\frac{\left(d\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)^2}{2\ell d\left((d+4)\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)}\|a_t\|^2 \\ &\leq -\frac{\left(d\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)\right)^2}{2\ell d\left((d+4)\cos^2(\theta)\cos^2(\rho) + \frac{d+4}{d}\sin^2(\theta)\right)}\|a_t\|^2 \\ &= -\frac{d\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)}{2\ell d(1+4/d)}\|a_t\|^2 \\ &\approx -\frac{d\cos^2(\theta)\cos^2(\rho) + \sin^2(\theta)}{2\ell d}\|a_t\|^2.\end{aligned}$$

This completes the proof of the theorem.

Table 8: ConMeZO increases peak memory usage (MiB) after 100 steps by a constant amount for each model across all the tasks

	RoBERTa-Large							OPT-1.3B				
Method	SST2	SST5	SNLI	MNLI	RTE	TREC	Avg.	SST2	BoolQ	DROP	SQuAD	Avg.
MeZO	**2178**	**2178**	**2177**	**2549**	**2542**	**2178**	**2300**	**3510**	**8983**	**24915**	**8621**	**11507**
ConMeZO	3343	3343	3343	4105	4099	3343	3596	6020	11493	27425	11132	14017
Δ Increase	1165	1165	1165	1556	1557	1165	1295	2510	2510	2511	2510	2510
% Increase	53.5%	53.5%	53.5%	61.0%	61.2%	53.5%	56.0%	71.5%	27.9%	10.1%	29.1%	34.6%

B IMPLEMENTATION AND PRACTICAL SPEEDUPS

As an additional contribution, ConMeZO introduces an efficient implementation framework that reduces wall-clock training time. This speedup arises from two complementary effects: (1) the optimizer converges in fewer iterations due to more informative, low-variance search directions, and (2) each iteration itself is faster thanks to a fully vectorized implementation.

Most MeZO variants retain similar algorithmic structure and perform perturbations and updates inside Python loops, generating random tensors for each parameter separately. In contrast, ConMeZO maintains a single flattened parameter buffer and applies perturbations or updates through fused in-place vector operations. This avoids repeated kernel launches, reduces RNG overhead, and minimizes Python-side iteration costs.

Beyond ConMeZO itself, this implementation framework is broadly applicable to other momentum-based zeroth-order optimizers, offering a general recipe for improving their computational efficiency without modifying the underlying algorithmic logic.

MeZO (loop-based perturbation):

```
def efficient_perturb_parameters(self, model, random_seed, scaling_factor=1):
    torch.manual_seed(random_seed)
    for name, param in self.named_parameters_to_optim:
        u = torch.normal(mean=0, std=1, size=param.shape, device=param.device)
        param.data += scaling_factor * u * self.args.zero_order_eps
    return model
```

ConMeZO (vectorized perturbation):

```
def efficient_perturb_parameters(self, model, random_seed, scaling_factor=1):
    fac = np.sqrt(self.d) / self.momentum_norm
    alpha = fac * np.cos(self.args.cone_theta)
    self.params_flat.add_(self.momentum_flat,
        alpha=scaling_factor * alpha * self.args.zero_order_eps)
    return model
```

For reference, ConMeZO samples the random direction once and stores the full perturbation in the momentum buffer. This design allows all three perturbations and the model update to be performed in a single vectorized pass, whereas MeZO re-samples random directions four times.

While ConMeZO improves computational efficiency, it introduces a modest and constant memory overhead due to maintaining the momentum vector. As shown in Table 8, this results in a consistent increase in peak memory usage across models and tasks.

C EXPERIMENTAL RESULTS AND DETAILS

C.1 Synthetic Experiments

This section provides details and additional results for the experiments in Section 5.1. Our objective is the quadratic problem $f(x) = \sum_{i=1}^{d} \sigma_i x_i^2$, where (σ_i) is a geometric series with initial value $1.0/d$ and final value

Table 9: RoBERTa-Large test performance. We include standard SGD as an additional FO baseline; remarkably, ConMeZO can outperform SGD on tasks like RTE.

	FO Methods		ZO Methods		
Task	AdamW	SGD	MeZO	Mom.	ConMeZO
SST-2	93.1	91.6	92.8	92.2	**93.5**
RTE	83.6	70.9	73.9	74.0	**75.1**

Table 10: Final test accuracy (%) of RoBERTa Large after 10,000 iterations with MeZO and ConMeZO, averaged over 5 seeds. Entries are mean ± std across seeds.

	SST-2	SST-5	SNLI	MNLI	RTE	TREC	Avg.
MeZO	92.8 ± 0.6	**49.3** ± 1.1	81.0 ± 0.4	69.7 ± 0.9	73.9 ± 0.9	88.4 ± 1.8	75.85
ConMeZO	**93.5** ± 0.7	48.9 ± 0.9	**81.9** ± 0.8	**73.2** ± 1.2	**75.1** ± 1.5	**90.0** ± 0.7	**77.11**

1.0. Hence the problem is strongly convex with condition number d. The initial iterate x_0 is randomly sampled from the vectors with norm 10.0. We set $\lambda = 0.01$ and tune the rest of the hyperparameters on the grid with choices $\eta = \{10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$, $\beta = \{0.8, 0.9, 0.95, 0.99\}$, $\theta = \{1.2, 1.3, 1.4, 1.5\}$. Tuning is done over $100,000$ iterations and the final plot is shown for $100,000$ iterations. Note that we do not apply any momentum parameter warm-up for synthetic experiments.

C.2 RoBERTa

This section provides details and additional results for the experiments in Section 5.2.

The finetuning process involved six standard NLP tasks from the GLUE benchmark (Wang et al., 2018a): SST-2 (binary sentiment classification), SST-5 (fine-grained sentiment classification) (Socher et al., 2013), SNLI (natural language inference) (Bowman et al., 2015), MNLI (multi-genre natural language inference) (Williams et al., 2018), RTE (recognizing textual entailment) (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009; Wang et al., 2018a), and TREC (question classification) (Voorhees and Tice, 2000).

The experiments were conducted by finetuning RoBERTa-large in a few-shot setting, using 512 samples per class, to evaluate the optimizer's performance. All experiments are executed on a single NVIDIA H100 GPU with ∼95 GiB of memory. Our implementation samples random directions from a standard normal distribution instead of from the sphere. This common practice simplifies implementation without hurting performance Zhang et al. (2024a). We fixed hyperparameters for our optimizer to $\theta = 1.35$ and $\beta = 0.99$ for reporting all results, while the learning rate $\eta = 10^{-6}$ and smoothing parameter $\lambda = 10^{-3}$ were fixed for both MeZO and ConMeZO to ensure a fair comparison. We tuned hyperparameters θ and β on the grid with choices $\theta = \{1.35, 1.4\}$, $\beta = \{0.95, 0.99\}$.

Our implementation builds on the framework provided by DPZero paper (Zhang et al., 2024a) and can be found at `https://github.com/LejsDeen/ConMeZO`. The optimizer's performance is analyzed under varying configurations of its hyperparameters: θ, β, and the learning rate η. These parameters are systematically adjusted to evaluate their sensitivity and impact on model performance. The smoothing parameter λ is fixed to 10^{-3} for all experiments. We use the seeds $13, 21, 42, 87, 100$ to calculate values provided in Tables 1, 10 and 11. In Figure 7 we provide test accuracy curves for a direct comparison between MeZO and ConMeZO.

Comparison to First-Order SGD. For completeness, we additionally report results for standard first-order SGD (FO-SGD) on two representative tasks, SST-2 and RTE in Table 9. We use the test accuracies reported in Zhang et al. (2024b) for FO-SGD.

Parameter Sensitivity & Ablation Study. Understanding the sensitivity of the optimizer to its hyperparameters, particularly momentum (β) and cone angle (θ), provides critical insights into its performance across different phases of optimization. Section C.4 explores their roles in convergence acceleration and alignment with the true gradient, highlighting key patterns observed in the experiments.

Table 11: Accuracy (%) with standard deviation of ConMeZO vs. MeZO on RoBERTa at different steps averaged over 5 seeds.

	SST-2		SST-5		SNLI	
Steps	MeZO	ConMeZO	MeZO	ConMeZO	MeZO	ConMeZO
1500	87.8 ±0.4	**90.2 ±1.1**	43.5 ±1.0	**44.4 ±1.4**	64.2 ±1.2	**68.1 ±1.3**
3000	91.4 ±0.5	**92.2 ±0.7**	46.1 ±1.0	**47.5 ±1.3**	73.0 ±1.3	**76.3 ±1.4**
6000	92.6 ±0.7	**92.9 ±0.7**	48.2 ±0.9	**48.5 ±1.1**	78.3 ±0.8	**79.6 ±0.3**

	MNLI		RTE		TREC	
Steps	MeZO	ConMeZO	MeZO	ConMeZO	MeZO	ConMeZO
1500	55.2 ±0.2	**57.3 ±0.4**	62.1 ±0.6	**65.6 ±1.5**	46.6 ±2.1	**49.8 ±3.4**
3000	60.5 ±0.6	**64.6 ±1.3**	68.3 ±0.5	**71.0 ±0.7**	66.2 ±2.5	**81.4 ±3.3**
6000	66.1 ±0.5	**69.7 ±1.7**	71.5 ±0.5	**73.6 ±0.6**	80.3 ±2.0	**88.8 ±0.9**

Table 12: OPT 1.3B Test Accuracy or F1 score in %. Mean ± std over 3 seeds.

	SQuAD	SST2	WIC	BoolQ
MeZO	72.76 ± 0.97	88.49 ± 0.29	56.53 ± 0.72	63.50 ± 0.78
ConMeZO	**75.34** ± 1.41	**90.56** ± 0.83	**58.15** ± 0.57	**64.20** ± 1.90
	DROP	ReCoRD	RTE	MultiRC
MeZO	25.90 ± 2.34	70.67 ± 0.75	**56.92** ± 1.16	**55.90** ± 1.23
ConMeZO	**26.53** ± 1.92	70.67 ± 0.45	55.48 ± 3.62	53.50 ± 1.41
Average	MeZO: 61.33		ConMeZO: **61.80**	

C.3 OPT

This section provides details and additional results for the experiments in Section 5.3.

All OPT-1.3B and 13B experiments are conducted for 20K iterations using both ConMeZO and MeZO with a fixed learning rate of $\eta = 10^{-7}$. Due to the high cost of finetuning a 1.3 and 13B-parameter model and rerunning multiple random seeds, additional learning-rate searches are not performed. Consequently, every OPT result reported use three seeds and $\eta = 10^{-7}$ is adopted for both optimizers. These findings align well with our RoBERTa results, demonstrating that the observed performance gains under fixed hyperparameter settings extend across different model scales. We use seeds 0, 29, and 83 for our reported results.

Our implementation builds on the DPZero framework (Zhang et al., 2024a). We fix the smoothing parameter to $\lambda = 10^{-3}$ and, instead of sampling uniformly from a hypersphere, draw random directions from $\mathcal{N}(0, I_d)$, which is a valid simplification in high dimensions.

Fine-tuning is conducted on decoder-only Transformers OPT-1.3B and OPT-13B (Zhang et al., 2022), respectively. We evaluate ConMeZO on eight benchmarks spanning diverse reasoning and understanding skills: SST-2 (Socher et al., 2013) (binary sentiment classification), BoolQ (Clark et al., 2019) (boolean question answering), SQuAD v1.1 (Rajpurkar et al., 2016) (span-based QA), DROP (Dua et al., 2019) (discrete reasoning QA), WiC (Pilehvar and Camacho-Collados, 2019) (word sense disambiguation), ReCoRD (Zhang et al., 2018) (reading comprehension with commonsense reasoning), RTE (Dagan et al., 2005; Bentivogli et al., 2009) (textual entailment), and MultiRC (Khashabi et al., 2018) (multi-sentence reasoning and multiple-choice comprehension).

Each task is fine-tuned for 20K iterations to capture performance across early, mid, and late training phases. For reporting final results of our optimizer, we fix hyperparameters to $\theta = 1.4, \beta = 0.99$. For ensuring a fair comparison, we fix the learning rate to 10^{-7} and the smoothing parameter to $\lambda = 10^{-3}$, which is the same for MeZO.

Table 13: OPT 13B Test Accuracy or F1 score in %. Mean ± std over 3 seeds.

	SQuAD	SST2	WIC	BoolQ
MeZO	82.28 ± 1.02	91.25 ± 0.18	**58.93** ± 0.57	67.73 ± 2.16
ConMeZO	**83.66** ± 0.71	**92.39** ± 0.58	58.31 ± 1.66	**69.33** ± 2.31
	DROP	ReCoRD	RTE	MultiRC
MeZO	OOM	**81.17** ± 1.04	63.66 ± 1.27	57.53 ± 2.27
ConMeZO	OOM	80.87 ± 1.00	**64.50** ± 1.63	**58.30** ± 0.56
Average	MeZO: 71.79		ConMeZO: **72.48**	

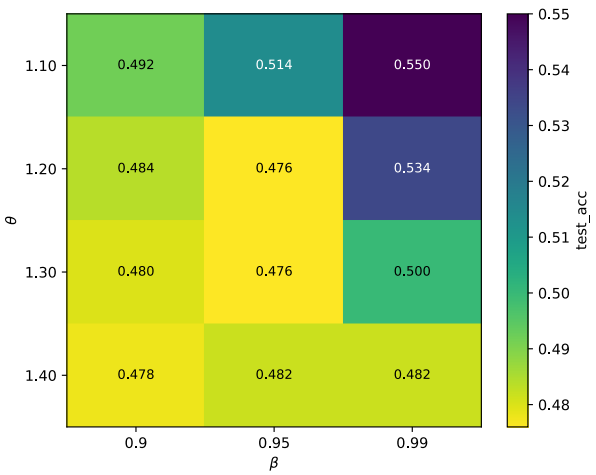
C.4 Additional Experimental Results

The results in Figure 5a, Figure 5b, and Figure 6b serve as an ablation study, illustrating how different hyperparameter choices shape the optimizer's behavior and convergence dynamics.

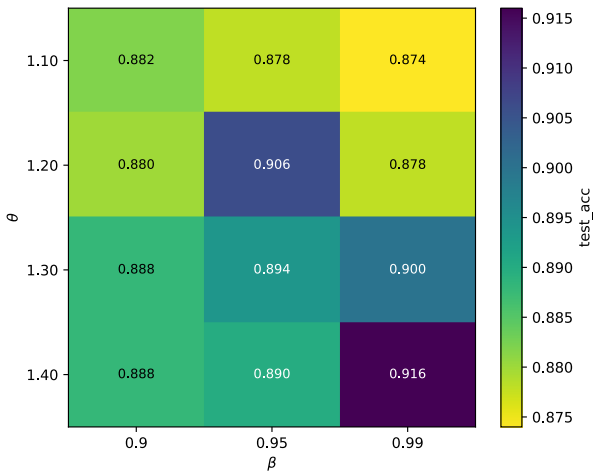
Early-Phase Convergence and Momentum Alignment. Momentum plays a critical role in maintaining consistent update directions during optimization. High momentum values, such as $\beta = 0.99$, help align updates with the true gradient direction, reducing variance and leading to faster and more stable convergence. As shown in the heatmaps (Figure 5a and Figure 5b), this effect becomes even more pronounced when combined with a small cone angle (θ), while more balanced configurations yield the strongest final performance.

Our analysis of the squared cosine similarity between the momentum vector and the true gradients (Figure 6) confirms this behavior. High momentum ($\beta = 0.99$) substantially improves directional alignment, achieving up to twice the accuracy of random directions during the first 2,000 iterations. The alignment remains consistently higher throughout training, although its relative gain decreases as convergence is approached, which indicates that high momentum primarily stabilizes updates rather than dominating progress in later stages.

Impact of Cone Angle: Exploration vs. Exploitation. The cone angle (θ) balances building an accurate gradient approximation against effectively exploiting it. Larger θ values sample broader directions, increasing cosine similarity with the true gradient but reducing reliance on the estimated gradient for updates. Figure 6b shows how varying θ affects cosine similarity over iterations.

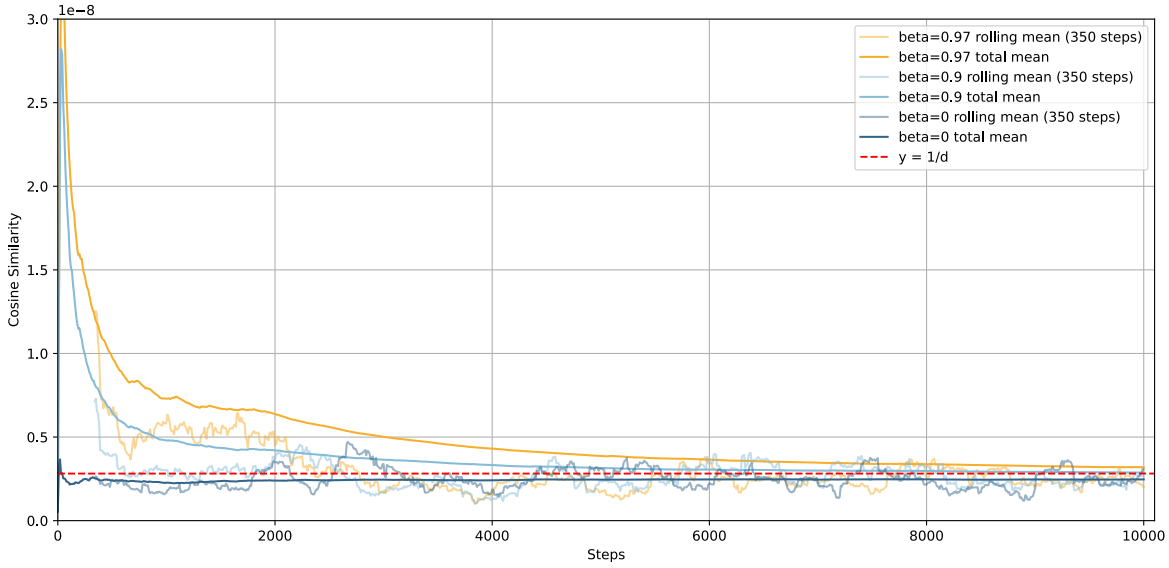


(a) After 1,000 iterations. MeZO's test accuracy after 1,000 iterations is 0.474.

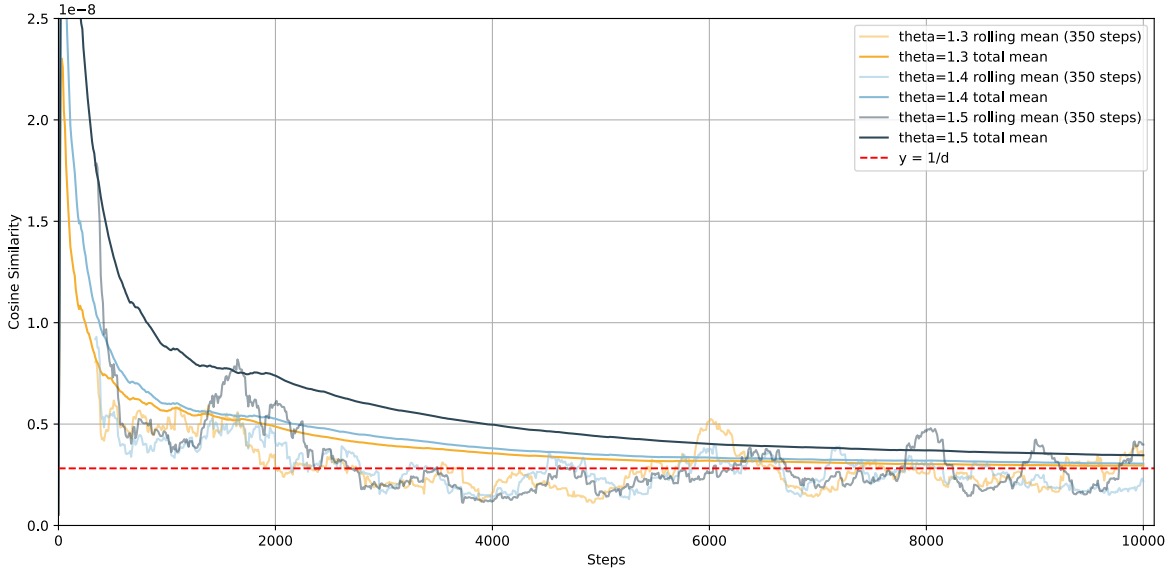


(b) After 10,000 iterations. MeZO's test accuracy after 10,000 iterations is 0.89.

Figure 5: Heatmaps of Test Accuracy of ConMeZO on TREC dataset for different θ and β values and fixed learning rate $\eta = 10^{-6}$.



(a) TREC dataset for $\theta = 1.3$ and varying β.



(b) TREC dataset for $\beta = 0.95$ and varying θ.

Figure 6: Squared cosine similarity between the true gradient and the momentum vector during training. $\frac{1}{d}$ in red highlights the expected squared cosine similarity for a random direction.

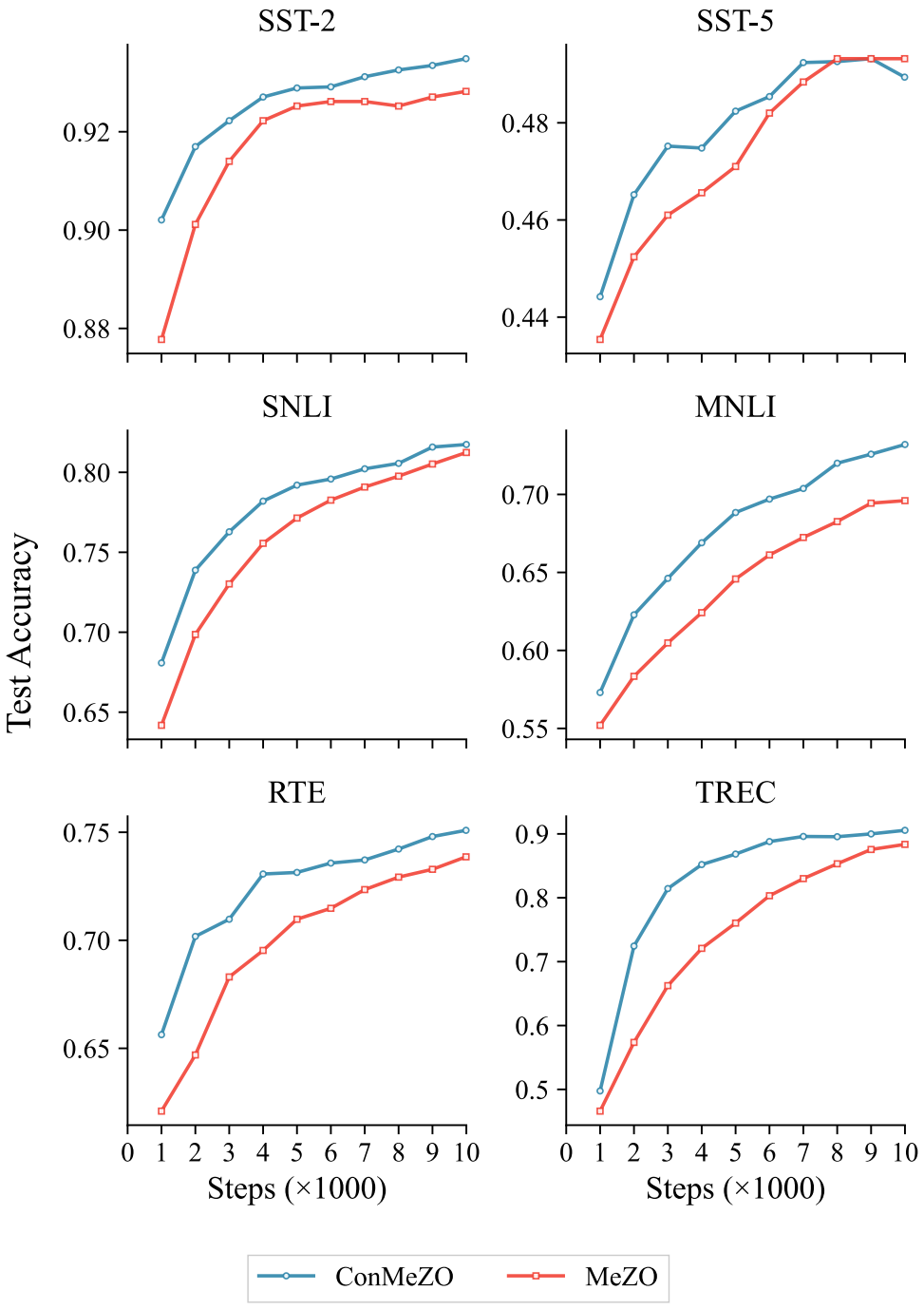


Figure 7: Test Accuracy of ConMeZO with settings mentioned in C.2 compared to MeZO over 10,000 iterations.

C.5 Momentum Warm-up Details

Intuition. With high momentum ($\beta = 0.99$, our default), the exponential moving average retains long memory: after 100 steps, a gradient still contributes $0.99^{100} \approx 37\%$ of the current momentum. Since early ZO estimates are noisy, assigning such large weight to initial estimates is undesirable. The warm-up starts with short memory ($\beta = 0.1$) to quickly forget early noise, then gradually increases β so that many recent gradients contribute comparably.

For an EMA, effective memory length scales as $1/(1-\beta)$, making the difference between $\beta = 0.95$ and $\beta = 0.99$

significant. Small changes near $\beta = 0.99$ cause large changes in effective memory:

- At $\beta = 0.95$: A 100-step-old gradient contributes $0.95^{100} \approx 0.6\%$
- At $\beta = 0.99$: A 100-step-old gradient contributes $0.99^{100} \approx 36.6\%$

Our schedule therefore increases quickly during early iterations and very slowly near the final value to match this exponential behavior. Figure 8 visualizes the warm-up curve.

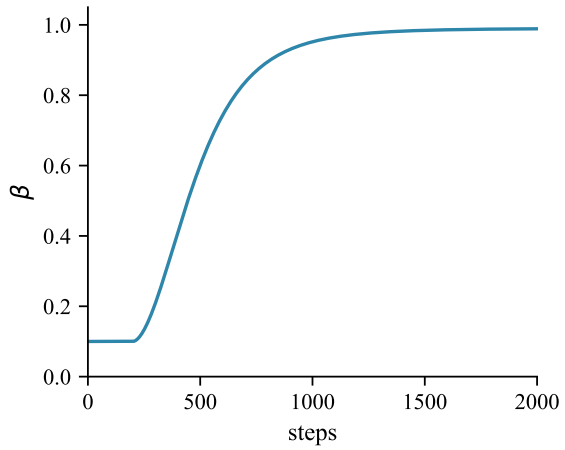


Figure 8: Momentum warm-up schedule for a 20K step training run with $\beta = 0.99$.

Ablation study. We compare ConMeZO with and without warm-up on RoBERTa-Large (5 seeds, 10K iterations).

Table 14: Momentum warm-up Ablation study. Incorporating a warm-up phase leads to the highest average accuracy and superior performance on the majority of tasks.

Method	SST-2	SST-5	MNLI	SNLI	RTE	TREC	Avg.
MeZO	0.928	**0.493**	0.697	0.810	0.739	0.884	0.759
ConMeZO (no warmup)	0.930	0.490	0.715	0.800	0.742	**0.908**	0.764
ConMeZO (with warmup)	**0.935**	0.489	**0.732**	**0.819**	**0.751**	0.900	**0.771**

The warm-up yields consistent improvements across all tasks, validating its effectiveness. Note that the warm-up is a practical heuristic rather than a core algorithmic component: all theoretical results assume fixed β, and synthetic experiments (Section 5.1) use ConMeZO without warm-up.

C.6 Licenses

Our evaluations are carried out on commonly-used datasets and models in the literature.

Datasets. GLUE (Wang et al., 2018a) is designed to provide a general-purpose evaluation of language understanding. Those adopted in our work include MNLI (inference, Williams et al. (2018)), SST-2/5 (sentiment analysis, Socher et al. (2013)), SNLI (natural language inference) (Bowman et al., 2015), RTE[1] (inference), and TREC (question classification, Voorhees and Tice (2000)). These datasets are released under different permissive licenses.

RoBERTa-large. This is a 355M parameter model. The model checkpoint[2] is released under the MIT license.

OPT-1.3B and OPT-13B. The model checkpoints[3] are released under a non-commercial license[4].

[1] https://paperswithcode.com/dataset/rte
[2] https://huggingface.co/FacebookAI/roberta-large
[3] https://huggingface.co/facebook/opt-1.3b, https://huggingface.co/facebook/opt-13b
[4] https://github.com/facebookresearch/metaseq/blob/main/projects/OPT/MODEL_LICENSE.md