Les Cahiers du GERAD ISSN: 0711–2440

Zeroth-order Kronecker optimization for pretraining language models

N. Allaire, S. Le Digabel, D. Orban, V. Partovi Nia

G–2025–44

July 2025
Revised: September 2025

Zeroth-order Kronecker optimization for pretraining language models

Nathan Allaire [a, b]

Sébastien Le Digabel [a, b]

Dominique Orban [a, b]

Vahid Partovi Nia [a, b, c]

[a] *Département de génie électrique, Polytechnique Montréal, Montréal, (Qc), Canada, H3T 1J4*

[b] *GERAD, Montréal (Qc), Canada, H3T 1J4*

[c] *Huawei Noah's Ark Lab, Montréal (Qc), Canada, H3N 1X9*

nathan.allaire@polymtl.ca
sebastien.le-digabel@polymtl.ca
dominique.orban@gerad.ca
vahid.partovi-nia@polymtl.ca

July 2025
Revised: September 2025
Les Cahiers du GERAD
G–2025–44

Abstract : Training language models (LMs) under tight GPU memory budgets rules out standard back-propagation and motivates *zeroth-order* (ZO) optimization. While ZO methods have proven effective for fine-tuning, their potential during the more memory-intensive pretraining stage has received little attention. We first revisit the singular-value spectra of layer gradients during pretraining and show that the gradient information is spread across many directions; low-rank ZO methods therefore potentially discard some informative components. Building on this insight, we introduce **KronZO**, a Kronecker-structured ZO optimizer that (i) explores a full-rank search subspace with state-of-the-art storage compression and (ii) employs a criterion-driven *directional* update that selectively keeps only informative steps. When pretraining GPT-2 Small from scratch on OpenWebText, KronZO achieves a markedly lower training loss than all previous ZO baselines while consuming less GPU memory. Although still trailing first-order methods in final loss, KronZO substantially narrows the gap at a fraction of their memory footprint, extending ZO optimization to larger models and longer runs and paving the way for memory-efficient pretraining on commodity hardware.

Keywords : Language models; pretraining; Zeroth-Order optimization; Kronecker factorization

Acknowledgements: The authors would like to thank Théo Denorme for his valuable contribution to the convergence analysis presented in this work.

1 Introduction

Recent scaling laws demonstrate that, when trained with the same optimization method and data, a larger LM (with more parameters) achieves a lower training loss and better downstream accuracy than a smaller LM [18].

Consequently, the number of parameters exploded (from 340M in BERT-base [30] to 70B in modern LMs) while commodity GPU memory has risen from 16GB to only 80GB over the same period. The dominant memory bottleneck during training is *not* due to the model weights themselves, but to the activation tensors retained for back-propagation, which can increase the overall memory footprint of training by up to 12 times on modern LMs compared to inference [24]. Even *parameter-efficient fine-tuning* (PEFT) techniques, such as *Low-Rank Adapters* like LoRA [16], leave this activation cost unchanged because they still rely on first-order optimization and gradient computation in the backward pass (SGD [2], Adam [20]).

Hardware-level approaches (model compression, quantization, pruning, knowledge distillation) alleviate inference cost but do not eliminate the back-propagation memory overhead during training. ZO algorithms provide an alternate solution: they approximate gradients from forward evaluations alone, bypass back-propagation entirely and eliminate activation storage for gradient computation [27]. Classical variants such as ZO-SGD [11] have matured theoretically [9, 25] and were recently adapted to LM fine-tuning through the *Memory-Efficient Zeroth-Order* (MeZO) algorithm [24]. Follow-up work introduced forward-gradient refinements [33] and the *Low-Rank Zeroth-Order* (LoZO) method [7], which exploits gradient low-rankness in fine-tuning to improve performance and reduce memory.

In contrast, ZO *pretraining* remains largely unexplored. Allaire et al. [1] provided first evidence that ZO *can* pretrain small LMs, but plain ZO-SGD falls short at larger scales.

We present **KronZO**, a new ZO method that

- exploits a Kronecker factorization to craft memory-efficient perturbations, markedly reducing the storage budget relative to state-of-the-art methods;
- employs high-dimensional perturbations to ensure broad exploration of the complex objective landscape;
- introduces a novel, criterion-driven *directional update* that stabilizes and improves convergence speed in highly stochastic regimes.

Related work

ZO optimization, or derivative-free optimization, addresses settings where gradients are unavailable, unreliable, or prohibitively expensive [3]. In machine learning, ZO algorithms estimate gradients from finite-difference queries [6, 27], and subsequently employ optimization methods based on these estimates. They have proved valuable for adversarial attacks [17, 29, 34], model interpretability [8], hyper-parameter search [21, 28], and more recently, LM training [1, 7, 24]. This field is currently under active research [31, 33].

Fine-tuning has become the de facto approach to adapt LMs for specific downstream applications [13]. Full fine-tuning has the same memory cost as regular pretraining since the number of trainable parameters is the same. Although PEFT reduces the number of trainable parameters, full back-propagation still dominates memory use. MeZO [24] showed that ZO methods can finetune LMs with comparable accuracy and lower memory footprint; LoZO [7] further leverages gradient structure in fine-tuning to surpass both MeZO and LoRA in terms of memory efficiency and convergence performance on fine-tuning tasks.

Pretraining poses additional challenges: extreme dimensionality, sharper non-convexity, batch-dependent noise. To the best of our knowledge, beyond the first study of Allaire et al. [1], ZO pretraining has not been explored in the literature. KronZO fills this gap, coupling Kronecker perturbations with innovative ZO updates to push the frontier of scalable, memory-efficient LM pretraining.

2 Preliminaries

In this section, we first introduce our notation, then give a brief overview of first order and ZO optimization for LMs before presenting our method, KronZO.

2.1 Notation

Let L be the total number of layers in an LM. Matrices in layer l have m^l rows and n^l columns, $l \in \{1, 2, \ldots, L\}$. The optimization proceeds over K iterations, indexed by k. Matrices are denoted $A^l, B^l, U^l, V^l, Z^l, \theta^l$, while sets of such matrices are written in bold, e.g., $\mathbf{Z} = \{Z^l\}_{l=1}^L$. The set of matrices $\boldsymbol{\theta} = \{\theta^l\}_{l=1}^L$ where $\theta^l \in \mathbb{R}^{m^l \times n^l}$ corresponds to the model parameters, or weights in layer l. The total number of parameters is $d = \sum_{l=1}^L m^l n^l$. For convenience, we occasionally treat matrices or sets of matrices as column vectors, e.g., $\mathbf{Z} \in \mathbb{R}^d$. The loss function to be minimized is denoted $\mathcal{L}$. Let ξ be a random variable representing the stochasticity in batch selection, and let ξ_k denote its realization at iteration k, i.e., the mini-batch sampled at iteration k. Finally, the cardinality of a set Ω is denoted $|\Omega|$.

We consider the optimization problem

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \mathcal{L}(\boldsymbol{\theta}) \quad \text{where} \quad \mathcal{L}(\boldsymbol{\theta}) := \mathbb{E}_\xi[\mathcal{L}(\boldsymbol{\theta}; \xi)]\,. \tag{1}$$

In an LM context, we assume that $\mathcal{L}$ is differentiable and has Lipchitz-continuous gradient.

2.2 First-order optimization

The classic way to solve (1) is to use the stochastic gradient method

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha \nabla \mathcal{L}(\boldsymbol{\theta}; \xi) \tag{2}$$

where $\nabla \mathcal{L}(\boldsymbol{\theta}; \xi)$ is the stochastic gradient of the loss with respect to $\boldsymbol{\theta}$ and $\alpha > 0$ is the step size, often referred to as the *learning rate*. At iteration k, a mini batch ξ_k is sampled from ξ, a forward pass gives the value $\mathcal{L}(\boldsymbol{\theta}_k; \xi_k)$, and the back-propagation computes $\nabla \mathcal{L}(\boldsymbol{\theta}_k; \xi_k)$ using the chain rule.

Back-propagation consumes more GPU memory than the forward pass because it must retain every layer output (activation) and store a gradient tensor for each weight matrix. For a batch of size $|\xi_k|$ and L layers, the principal memory terms are

$$\underbrace{\mathcal{M}_{\text{act}} = |\xi_k| \sum_{l=1}^L n^l,}_{\text{activations}} \qquad \underbrace{\mathcal{M}_{\text{grad}} = \sum_{l=1}^L m^l n^l}_{\text{weight gradients}}.$$

Most first-order optimizers keep $s_{\text{FO}} \in \{0, 1, 2\}$ auxiliary buffers per parameter (e.g., momentum for SGD, first and second moments for AdamW). Including the weights themselves, the memory consumed by the optimizer state is

$$\mathcal{M}_{\text{opt,FO}} = (1 + s_{\text{FO}}) \sum_{l=1}^L m^l n^l.$$

Hence the *peak first-order footprint* is

$$\mathcal{M}_{\text{FO}} = |\xi_k| \sum_{l=1}^L n^l + (2 + s_{\text{FO}}) \sum_{l=1}^L m^l n^l. \tag{3}$$

2.3 ZO optimization

ZO methods do not back-propagate, so they require neither $\mathcal{M}_{\text{act}}$ nor $\mathcal{M}_{\text{grad}}$. They store only the weights and a method-specific state (see sets of matrices $\mathbf{A}$ and $\mathbf{B}$ for KronZO in section 3) whose size is at most a fraction $s_{\text{ZO}} \leq 1$ of the parameter count:

$$\mathcal{M}_{\text{ZO}} = (1 + s_{\text{ZO}}) \sum_{l=1}^{L} m^l n^l, \qquad s_{\text{ZO}} \leq 1. \tag{4}$$

Malladi et al. [24] report that training OPT-13B requires 12 times more memory than inference, a ratio consistent with (3) and with our own measurements (see Figure 6 in Section 5).

To sidestep back-propagation and its significant memory overhead, we turn to a ZO paradigm that relies exclusively on loss evaluations.

Finite differences

The most straightforward ZO technique is to estimate directional derivatives by *finite differences* [11], namely the Random Gradient Estimator (RGE)

$$\widehat{\nabla}\mathcal{L}(\boldsymbol{\theta}; \xi) := \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \mathbf{Z}; \xi) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \mathbf{Z}; \xi)}{2\epsilon} \mathbf{Z}, \tag{5}$$

where $\mathbf{Z} \sim \mathcal{N}(0, \mathbf{I}_d) \in \mathbb{R}^d$, and $\epsilon > 0$ is a small perturbation step size called the *smoothing parameter*. In order to reduce the variance inherent to RGE (5), it can be interesting to average the estimator over q randomly sampled directions $\mathbf{Z}_i$ with q-RGE

$$\widehat{\nabla}_q\mathcal{L}(\boldsymbol{\theta}; \xi) := \frac{1}{q} \sum_{i=1}^{q} \frac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \mathbf{Z}_i; \xi) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \mathbf{Z}_i; \xi)}{2\epsilon} \mathbf{Z}_i, \tag{6}$$

where each $\mathbf{Z}_i \sim \mathcal{N}(0, \mathbf{I}_d) \in \mathbb{R}^d$ and q is the *query budget*.

When $q = 1$, (6) reduces to (5). Since $\lim_{\epsilon \to 0} \frac{\mathcal{L}(\boldsymbol{\theta}+\epsilon\mathbf{Z};\xi) - \mathcal{L}(\boldsymbol{\theta}-\epsilon\mathbf{Z};\xi)}{2\epsilon} = \mathbf{Z}^\top \nabla\mathcal{L}(\boldsymbol{\theta}; \xi)$ in vectorized notation, (5) can be seen as the projection of the stochastic gradient on $\mathbf{Z}$. Since $\mathbf{Z}$ comes from a standard normal distribution, (5) and (6) are non-biased estimators of the stochastic gradient:

$$\mathbb{E}[\widehat{\nabla}\mathcal{L}(\boldsymbol{\theta}; \xi)] = \mathbb{E}[\widehat{\nabla}_q\mathcal{L}(\boldsymbol{\theta}; \xi)] = \nabla\mathcal{L}(\boldsymbol{\theta}; \xi).$$

At each iteration, the parameters are updated similarly to (2):

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \widehat{\nabla}\mathcal{L}(\boldsymbol{\theta}_k; \xi_k). \tag{7}$$

Malladi et al. [24] use (5) and (6) to finetune LMs with MeZO.

For convenience, we also introduce the layer-wise notation of (5) and (6)

$$\widehat{\nabla}\mathcal{L}(\theta^l; \xi) := \frac{\mathcal{L}(\theta^l + \epsilon Z^l; \xi) - \mathcal{L}(\theta^l - \epsilon Z^l; \xi)}{2\epsilon} Z^l, \qquad l = 1, 2, \ldots, L \tag{8}$$

$$\widehat{\nabla}_q\mathcal{L}(\theta^l; \xi) := \frac{1}{q} \sum_{i=1}^{q} \frac{\mathcal{L}(\theta^l + \epsilon Z_i^l; \xi) - \mathcal{L}(\theta^l - \epsilon Z_i^l; \xi)}{2\epsilon} Z_i^l \qquad l = 1, 2, \ldots, L, \tag{9}$$

where $\theta^l \in \mathbb{R}^{m^l \times n^l}$ are the layer parameters. Thus Z^l and $Z_i^l \in \mathbb{R}^{m^l \times n^l}$. Without additional effort for storing Z^l efficiently, the memory required to run RGE or q-RGE is $\mathcal{O}(m^l n^l)$ for each layer l, which corresponds to $s_{\text{ZO}} = 1$ in (4). In MeZO, only the seed to generate Z^l is stored, and it can be regenerated from this seed *on-the-fly* during computations (thus $s_{\text{ZO}} \ll 1$). Despite reducing the memory costs, the regeneration process creates additional floating points operations.

Low-rank ZO

In MeZO, $Z^l \in \mathbb{R}^{m^l \times n^l}$ has very likely full rank, i.e., $\operatorname{rank}(Z^l) \approx \min\{m^l, n^l\}$. Based on the observation that gradients exhibit a low-rank structure during fine-tuning, Chen et al. [7] introduced LoZO to align the structure of the stochastic gradient estimator with that of the real stochastic gradient. In their work, $Z_i^l = U_i^l (V^l)^\top$, where $U^l \in \mathbb{R}^{m^l \times r}$, $V^l \in \mathbb{R}^{n^l \times r}$ with $0 < r \ll \min\{m^l, n^l\}$, both sampled independently from a standard Gaussian distribution. The layer-wise gradient estimator is

$$\widehat{\nabla}_q^{\mathrm{LoZO}} \mathcal{L}(\theta^l;\xi) := \sum_{i=1}^{q} \frac{\mathcal{L}(\theta^l + \epsilon U_i^l (V^l)^\top;\xi) - \mathcal{L}(\theta^l - \epsilon U_i^l (V^l)^\top;\xi)}{2\epsilon} \frac{U_i^l (V^l)^\top}{r}. \tag{10}$$

The scaling $1/r$ in (10) makes the gradient estimator unbiased. The matrix U_i^l is sampled q times at every iteration for each layer, while V^l is refreshed once every ν iteration ($\nu \in \mathbb{N}_{>0}$) for each layer. This way, (10) explores the subspace defined by V^l for ν steps. Since $\operatorname{rank}(U^l) \approx r$ and $\operatorname{rank}(V^l) \approx r$, $\operatorname{rank}(U^l V^{l\top}) \approx r$, and the perturbation has low rank.

LoZO requires storing all U_i^l and V^l for each layer, for a memory cost of $\mathcal{O}(m^l r + n^l r)$. With a typical attention layer size of 784 × 2,304 (GPT-2 Small, BeRT-base), MeZO needs to store roughly 2×10^6 additional floats per layer, while LoZO only needs to store $784 \times r + 2{,}304 \times r$ floats, which is about 10^4 with the typical value $r = 4$. LoZO showed state-of-the-art fine-tuning efficiency on a large family of tasks and LMs.

Studies on *intrinsic dimension* in LMs [22, 23] show that stochastic gradients collapse into a low-dimensional subspace during fine-tuning, making low-rank ZO optimizers such as LoZO highly effective. These findings rely on analyzing the *singular-value spectrum* (the ordered set of singular values) of each layer's gradient matrix, which in fine-tuning is sharply peaked and therefore concentrates most of the information in only a few directions. Figure 1 confirms this behavior on GPT-2: as fine-tuning proceeds on the Shakespeare dataset, more singular values shrink while a handful remain large, signaling an increasingly low-rank structure.

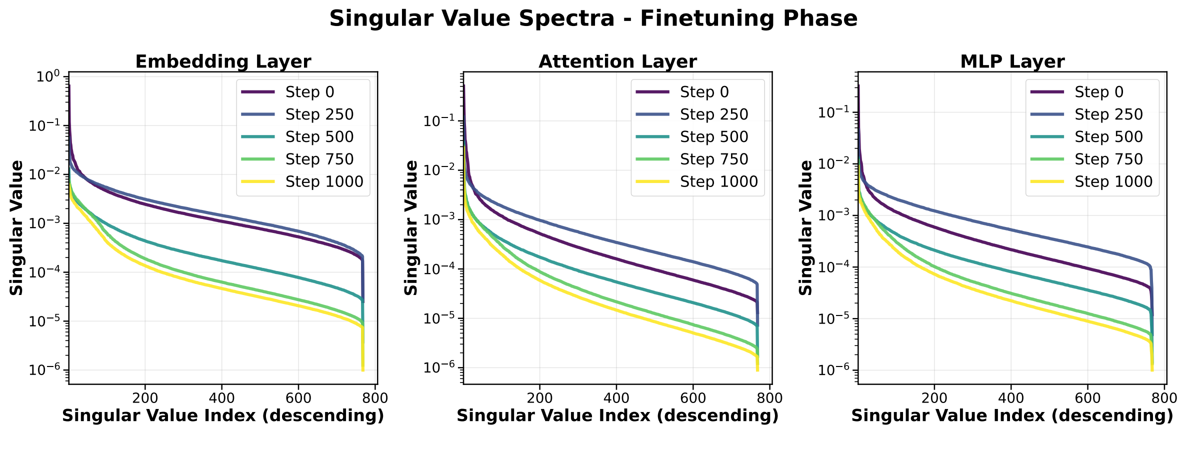

Figure 1: Evolution of stochastic gradients spectra during GPT-2 fine-tuning on the Shakespeare dataset. At early steps, the purple and dark-blue curves are relatively flat, with several high singular values. At step 1,000, the yellow curve still retains a handful of high singular values but drops off sharply: information concentrates rapidly in a few directions, evidencing low-rank collapse.

By contrast, Figure 2 shows that during *pretraining* the spectrum remains comparatively flat: although a handful of directions still carry the largest singular values, many others exhibit mid-range magnitudes, indicating that information is spread across a wide set of modes.

This qualitative gap suggests that low-rank ZO methods, while effective for fine-tuning, may not be suitable for pretraining. In such settings, high-rank ZO methods are more appropriate, as supported

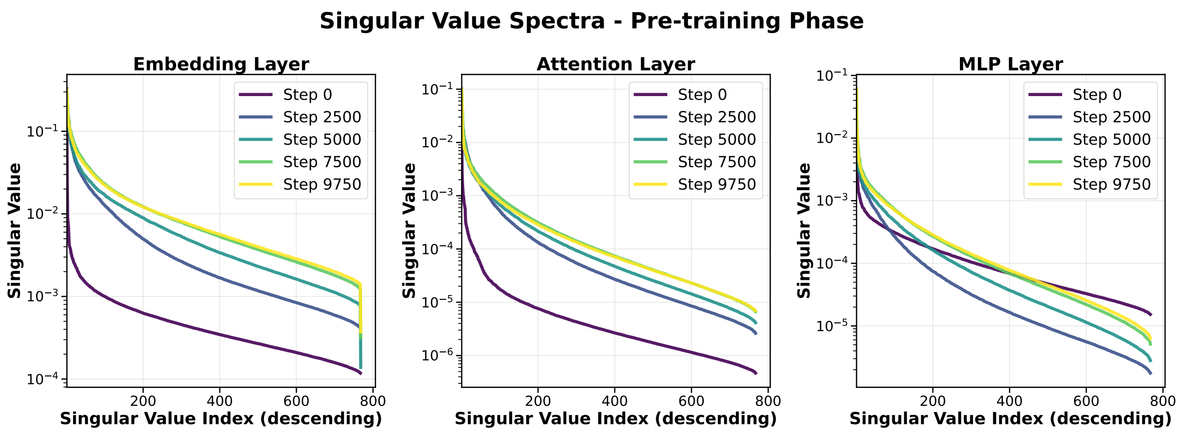

Figure 2: Evolution of stochastic gradients spectra in GPT-2 pretraining. During early steps, the purple and dark-blue curves show a very sharp descent. At final steps, the yellow curve is flatter, indicating a broader spectrum.

by our results in Section 5. Since almost no singular value is exactly zero, the algebraic rank remains nearly full at each step. To gauge the true dimensionality of pretraining gradients, we therefore turn to *effective-rank* measures instead.

Spectral structure of pretraining gradients

In this section, we compare the widely used *stable rank* with the *participation ratio* (PR) and show that the former can underestimate dimensionality, whereas the latter more accurately captures the richness of LM stochastic gradients.

Stable rank

For a matrix $A \in \mathbb{R}^{m\times n}$ with singular values $\{\sigma_i\}_{i=1}^{\mu}$, where $\mu = \min(m,n)$ the stable rank is

$$\operatorname{srank}(A) \;=\; \frac{\|A\|_F^2}{\|A\|_2^2} \;=\; \frac{\sum_{i=1}^{\mu}\sigma_i^2}{\max_{i=1}^{\mu}\sigma_i^2}. \tag{11}$$

Throughout this section, we call $\lambda_i = \sigma_i^2$ the *energy* of singular direction i, and $(\lambda_1,\dots,\lambda_\mu)$ the energy vector with total energy $\sum_{i=1}^{\mu}\lambda_i$.

Although continuous, the stable rank is driven by the *largest* singular value and can mask numerous medium-energy directions. Take $D = \operatorname{diag}(100, 60, \dots, 60)$ where 60 is repeated 19 times; thus $\operatorname{rank}(D) = 20$. D has a structure similar to pretraining gradients: one high singular value, and many medium-sized. Its energy vector is $\boldsymbol{\lambda} = (100^2, 19 \times 60^2)$ and $\sum_{i=1}^{\mu}\lambda_i = 78\,400$. Nineteen medium directions therefore carry $68\,400/78\,400 \approx 87\%$ of the energy, yet

$$\operatorname{srank}(D) = \frac{10\,000 + 19\times 3\,600}{10\,000} \approx 7.8,$$

i.e., only 40% of the true rank. Figure 3 echoes this behavior: the stable-rank curve of GPT-2 gradients suggests an overly optimistic low rank during pretraining.

Participation ratio

To capture all energetic directions, we adopt the *participation ratio* (PR) [10], defined on $\lambda_i = \sigma_i^2$ for $i = 1, 2, \dots, \mu$:

$$\operatorname{pr}(A) \;=\; \frac{\left(\sum_{i=1}^{\mu}\lambda_i\right)^2}{\sum_{i=1}^{\mu}\lambda_i^2} \;=\; \frac{\left(\sum_{i=1}^{\mu}\sigma_i^2\right)^2}{\sum_{i=1}^{\mu}\sigma_i^4}. \tag{12}$$

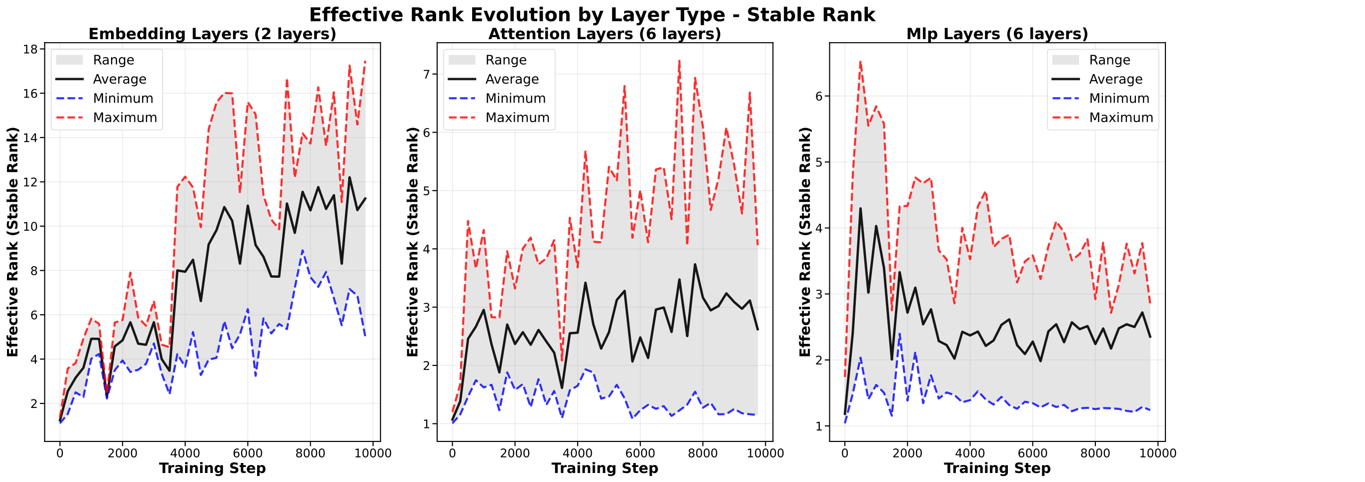

Figure 3: Stable rank (11) of GPT-2 gradients during pretraining. Because it tracks the largest singular values, it indicates a misleadingly low dimensionality across layer types.

As the inverse Herfindahl index [15], the PR estimates *how many singular directions carry non-negligible energy*. Applying (12) to matrix D from the previous section,

$$\mathrm{pr}(D) = \frac{(10\,000 + 19 \times 3\,600)^2}{10\,000^2 + 19 \times 3\,600^2} \approx 17.8,$$

far closer to the algebraic rank of 20 and more than double the stable-rank value. Figure 4 confirms this trend in real gradients: during pretraining, the PR rises to roughly 30% of the layer width.

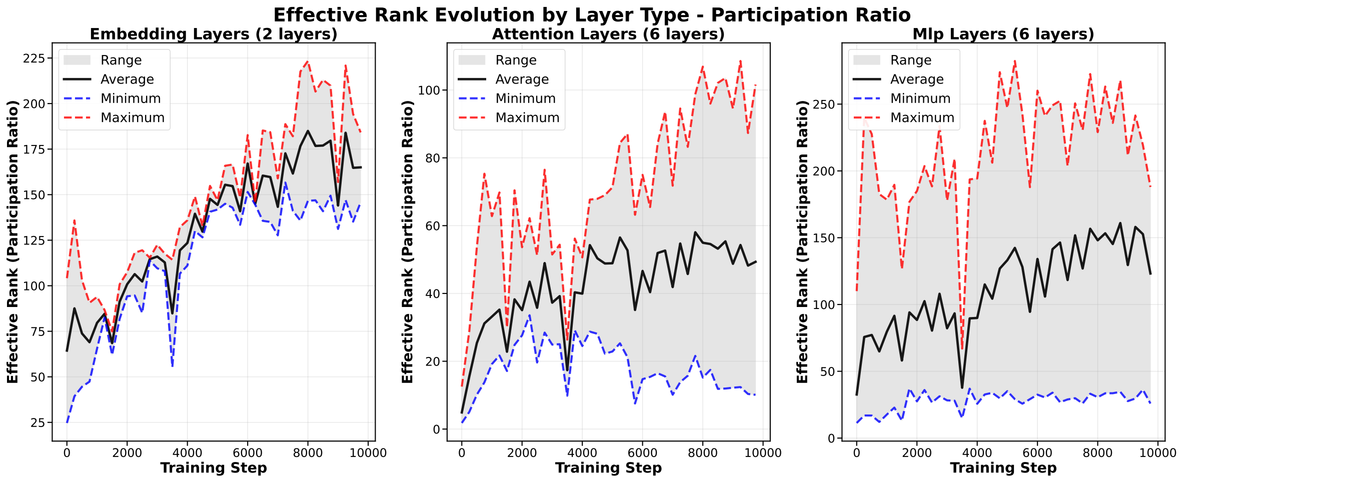

Figure 4: Participation ratio (Equation 12) of the same gradients. Many more directions are energetically relevant, contradicting the low-rank impression given by the stable rank.

Because the PR remains sensitive to medium-sized singular values, it delivers a much more faithful effective rank for LM pretraining gradients, whose spectrum is broad. Although each subplot is noisy and not strictly monotonic, the overall upward trend of both the stable-rank and participation-ratio in Figures 3 and 4 shows that pretraining stochastic gradients do *not* have low-rank; they begin at a moderate rank and increase steadily as training progresses.

Consequently, ZO methods that limit updates to a tiny subspace (e.g., LoZO) are ill-suited to pretraining, whereas full-rank schemes are better aligned with the true dimensionality. This observation is consistent with the results reported in Section 5, where full-rank methods outperform their low-rank counterparts in pretraining. However, MeZO incurs a significant overhead: it must either store the entire matrix $\mathbf{Z}$ or regenerate it from the random seed at the cost of extra flops, an impractical choice for long-running tasks such as pretraining.

To cope with the broad subspace revealed by the PR while keeping memory use and flops manageable, we turn to KronZO, a *full-rank* ZO optimizer that retains all informative directions through a lightweight Kronecker parameterization.

3 ZO Kronecker optimization (KronZO)

In this section, we present KronZO, our proposed ZO Kronecker optimization method for pretraining LMs. Recall that the Kronecker product of $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n_1} \\ a_{21} & a_{22} & \cdots & a_{1n_2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_1 1} & a_{m_1 2} & \cdots & a_{m_1 n_1} \end{bmatrix} \in \mathbb{R}^{m_1 \times n_1}$ and $B \in \mathbb{R}^{m_2 \times n_2}$ is defined as

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n_1}B \\ a_{21}B & a_{22}B & \cdots & a_{1n_1}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_1 1}B & a_{m_1 2}B & \cdots & a_{m_1 n_1}B \end{bmatrix} \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}.$$

The Kronecker product possesses several useful algebraic properties; see [12] for a comprehensive overview. In this work, we focus on its rank-preserving property:

$$\mathrm{rank}(A \otimes B) = \mathrm{rank}(A) \times \mathrm{rank}(B). \tag{13}$$

The Kronecker product has many applications in machine learning, notably thanks to its very efficient and flexible factorizations in CNNs [14]. We use this formulation to generate compact, full-rank random perturbations in KronZO.

KronZO is innovative in two ways. First, its **compression** is more efficient than LoZO (see Algorithm 1), further reducing memory overhead, and the sampled directions are richer, which is key in pretraining. Second, its innovative, criterion-based **parameters update** makes it more efficient than previous ZO methods. In Algorithm 2, the matrix B^l is updated every ν iterations, which is the same behavior as ν for V^l in LoZO. The query budget q is the same as parameter q in (10): at each iteration, q random perturbations are sampled independently. However, the way those perturbations are managed is different from (6) and (10), see Section Parameters update.

Compression

For each layer l, KronZO generates and stores two Kronecker factors $A^l \in \mathbb{R}^{m_1^l \times n_1^l}$ and $B^l \in \mathbb{R}^{m_2^l \times n_2^l}$ such that $m_1^l m_2^l = m^l$ and $n_1^l n_2^l = n^l$ to form perturbation $Z^l = A^l \otimes B^l \in \mathbb{R}^{m^l \times n^l}$. The routine ChooseKronDims (Algorithm 1) enforces these dimensionality constraints: it calls BestPair, which returns the two divisors of an integer closest to its square root, falling back to $(1, n)$ if the integer is prime (which is generally not the case in LMs). In short, ChooseKronDims selects the shapes of A^l and B^l so that each factor is as close to square as possible. For example, a weight matrix of size 768 × 2,304 yields $(m_1^l, m_2^l) = (24, 32)$ and $(n_1^l, n_2^l) = (48, 48)$. In Table 1, we compare the compression rates between the three methods on a typical attention layer of the GPT-2 model.

As highlighted in Section Spectral structure of pretraining gradients, the pretraining task requires exploring the full optimization space. Thus, low-rank methods such as LoZO are less tailored for this task. Using (13), since A^l and B^l almost certainly have full rank, $A^l \otimes B^l$ also almost certainly has full rank. Similarly to MeZO, this allows the perturbation to span the full optimization space. Therefore, KronZO outperforms the memory efficiency of LoZO, and gives insightful information on the complete optimization space, like MeZO.

Algorithm 1 Approximate Square Kronecker Factorization Strategy for KronZO

```
1: function BestPair($n$)
2:     root $\leftarrow \lfloor\sqrt{n}\rfloor$
3:     for offset $= 0$ to root do
4:         for cand $\in$ {root $-$ offset, root $+$ offset} do
5:             if cand $\geq 1$ and $n$ mod cand $= 0$ then
6:                 return (cand, $n$/cand)
7:             end if
8:         end for
9:     end for
10:    return $(1, n)$                                  ▷ Fallback for prime numbers
11: end function

12: function ChooseKronDims($m^l, n^l$)
13:    $(m_1, m_2) \leftarrow$ BestPair($m^l$)                       ▷ Row factorization
14:    $(n_1, n_2) \leftarrow$ BestPair($n^l$)                    ▷ Column factorization
15:    return $(m_1, m_2, n_1, n_2)$
16: end function

    Constraint: $m_1 \times m_2 = m^l$ and $n_1 \times n_2 = n^l$
    Objective: Minimize $|m_1 - \sqrt{m^l}| + |m_2 - \sqrt{m^l}| + |n_1 - \sqrt{n^l}| + |n_2 - \sqrt{n^l}|$
```

Table 1: Storage cost and compression rates for MeZO, LoZO and KronZO on a typical attention layer in GPT-2. For LoZO, we choose $r = 4$. Our method achieves better compression rates while retaining expressive perturbations.

Method	Sampling structure	Parameters stored	Compression	Example size
MeZO	Full matrix Z^l	mn	1	768×2304
LoZO	Low rank U^l, V^l	$mr + nr$	150	$U^l : 768\times4$, $V^l : 2304\times4$
KronZO	Kron. factors A^l, B^l	$m_1n_1 + m_2n_2$	650	$A^l : 24\times48$, $B^l : 32\times48$

Parameters update

The other major innovation in KronZO is in the way parameters are updated. In the regular q-RGE (6), q random perturbations are sampled, and the stochastic gradient estimators are computed and averaged, which results in $2q$ evaluations of $\mathcal{L}$. Some inappropriate random perturbations can dominate promising ones, leading to a poor averaged estimator in practice.

In KronZO (Algorithm 2), we iteratively choose a perturbation that results in the largest decrease in the objective. For this purpose, at iteration k, we sample a fresh batch ξ_k (Line 4) and $\mathbf{B}$ if $k \ \% \ \nu = 0$ (Line 6). Then, we generate q random perturbations $\mathbf{A}_i$, $(i = 1, 2, \ldots, q)$ and compute the projection of the stochastic gradient on $\mathbf{Z}_i = \mathbf{A}_i \otimes \mathbf{B}$: $c_i\mathbf{Z}_i$ (Line 10) before computing $\mathcal{L}(\boldsymbol{\theta}_k - \alpha c_i \mathbf{Z}_i; \xi_k)$ (Line 11). We keep only the best of those q directions (Lines 12-14), i.e., the lowest of the evaluated losses. The way KronZO samples perturbations can be seen as a lucky RGE: we only use one direction to update, but this direction is competing against $q - 1$ others. It is therefore a biased perturbation, and may be a descent direction. This strategy costs $3q$ evaluations of the objective per iteration, which is more expensive than (6) but gives richer directions. We illustrate this mechanism in Figure 5. With the same five random directions, the averaging rule blends useful and misleading signals, so its gradient estimator diverges from the true gradient. KronZO evaluates every candidate, picks the single direction that yields the largest objective decrease, and produces an estimate that tracks the true descent direction more closely. In high-dimensional spaces, e.g., pretraining, this dilution effect grows: the probability that informative directions are drowned out by many irrelevant ones rises, whereas KronZO still retains the best-performing direction. Although the stochastic gradient estimate used by KronZO is biased (whereas the q-RGE estimator in (6) is not), the perturbation-selection mechanism markedly improves convergence. As shown in Section 5, this strategy achieves a substantially lower loss than both (6) and the low-rank variant (10) under the same query budget.

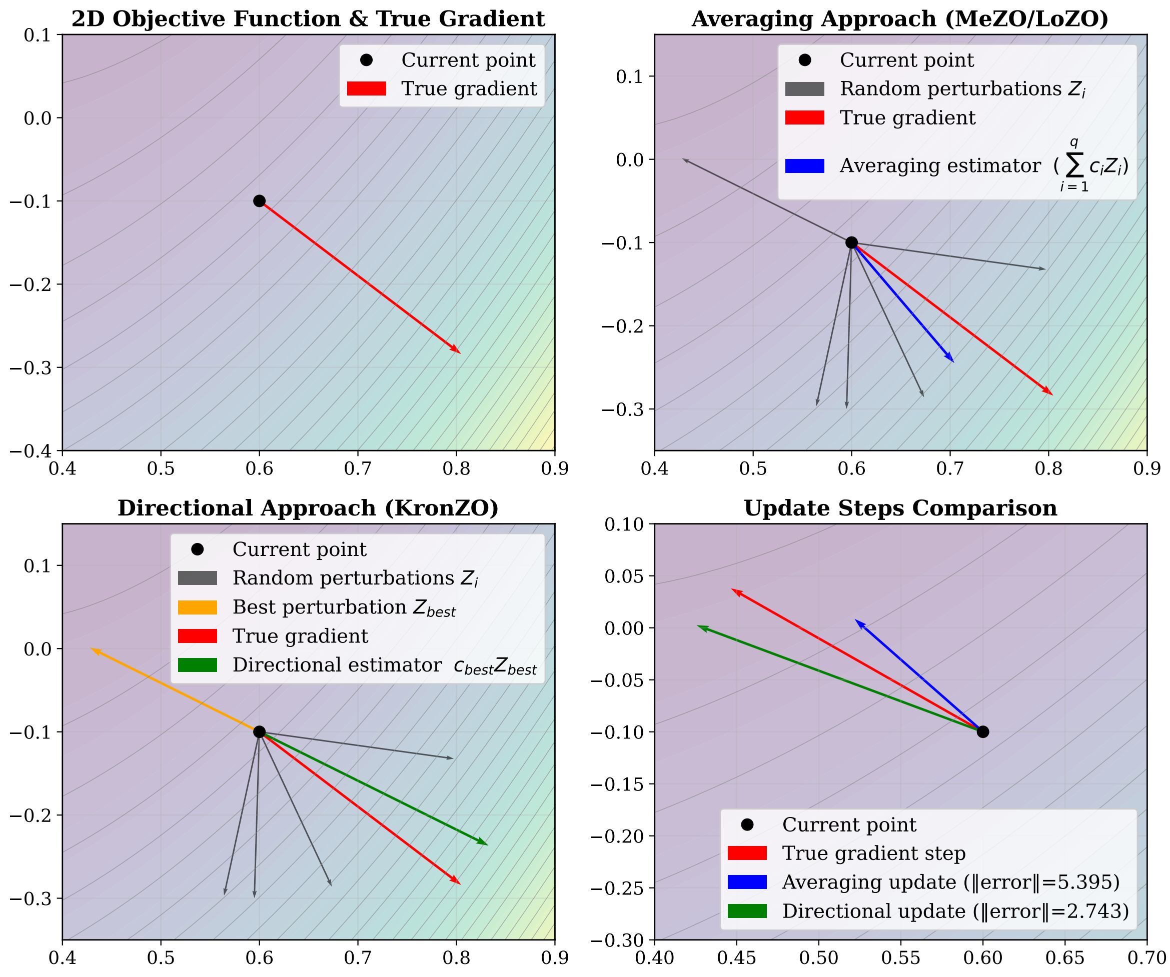

Figure 5: Directional versus averaging updates on a two dimensional Rosenbrock example (same five random directions in all panes). Top-left: objective contours and true gradient (red). Top-right: q-RGE/averaging—individual perturbations (black) and their mean estimator (blue). Bottom-left: KronZO keeps only the best perturbation (orange) and forms its directional estimator (green). Bottom-right: parameter-update vectors: the directional step lies much closer to the true gradient step than the averaged one.

Loss landscapes are highly non-convex, containing many local minima and exhibiting sensitivity to noise due to batch variability. Since the chosen perturbation direction depends on the current batch (which may differ significantly from others) consistently updating along the best batch-dependent direction can be unreliable. Moreover, we only sample a small number of perturbations ($q = 10$ to 50), which is negligible compared to the dimensionality of the model. As a result, the selected best perturbation may not accurately reflect the overall landscape, potentially leading to suboptimal, or even *stale* updates that are uncorrelated with true descent.

To mitigate this, we compare the current best loss (over the current batch) to a sliding window of the past h losses over previous batches (Line 16). If it improves upon the worst loss in the window, we update the parameters (Line 17). Otherwise, we consider the iteration a failure and skip the update. Empirically, such failures occur in approximately 10% of iterations. In both cases, the window is refreshed with current best loss replacing the maximum loss of the window (Line 19). We term this strategy (perturbation selection and acceptance) a *directional update*: parameters are advanced only when a single, carefully vetted direction, chosen for its within-batch effectiveness and cross-batch robustness, outperforms all q others, ensuring that the step aligns with the true descent geometry rather than averaging random perturbations.

Algorithm 2 Kronecker ZO optimization (KronZO)

```
1: Input: objective $\mathcal{L}$, step budget $K$, smoothing parameter $\epsilon$, learning rate $\alpha$, sampling interval $\nu$, sampling strategy
   $S(m,n)$, query budget $q$, history length $h$.

2: Init: ChooseKronDims($m^l, n^l$) for each layer $l$                      ▷ see Algorithm 1
3: for $k = 0, 1, \dots, K-1$ do
4:     Sample fresh batch $\xi_k$
5:     Set $\mathcal{L}_{\text{best}} = \mathcal{L}(\boldsymbol{\theta}_k; \xi_k)$, $c_{\text{best}} = \emptyset$, $\mathbf{Z}_{\text{best}} = \emptyset$
6:     if $k \,\%\, \nu = 0$ then sample $\mathbf{B} = \{B^l\}_{l=1}^L : m_2^l \times n_2^l$             ▷ refresh B
7:     end if
8:     for $i = 1, 2, \dots q$ do
9:         Sample $\mathbf{A}_i = \{A_i^l\}_{l=1}^L : m_1^l \times m_2^l$ and define $\mathbf{Z}_i = \mathbf{A}_i \otimes \mathbf{B}$
10:        Compute $c_i = \dfrac{\mathcal{L}(\boldsymbol{\theta} + \epsilon \mathbf{Z}_i; \xi_k) - \mathcal{L}(\boldsymbol{\theta} - \epsilon \mathbf{Z}_i; \xi_k)}{2\epsilon}$
11:        Compute $\mathcal{L}_i = \mathcal{L}(\boldsymbol{\theta}_k - \alpha c_i \mathbf{Z}_i)$          ▷ evaluate at candidate update (7)
12:        if $\mathcal{L}_i < \mathcal{L}_{\text{best}}$ then
13:            Set $\mathcal{L}_{\text{best}} = \mathcal{L}_i$, $c_{\text{best}} = c_i$, $\mathbf{Z}_{\text{best}} = \mathbf{Z}_i$
14:        end if
15:    end for
16:    if $\mathcal{L}_{\text{best}} \le \max\{\mathcal{L}_{k-1}, \mathcal{L}_{k-2}, \dots, \mathcal{L}_{k-h}\}$ then
17:        Update $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha c_{\text{best}} \mathbf{Z}_{\text{best}}$
18:    end if
19:    Update history: $\max\{\mathcal{L}_{k-1}, \mathcal{L}_{k-2}, \dots, \mathcal{L}_{k-\min\{h,k\}}\} \leftarrow \mathcal{L}_{\text{best}}$
20: end for
```

For each method, Table 2 indicates the *frozen component* that remains fixed for the next ν optimization steps for each layer, the corresponding set of admissible perturbations, i.e., the linear subspace $\mathcal{S}$ explored during that window, $\dim(\mathcal{S})$: the number of degrees of freedom, and the *typical rank* of a single update drawn from $\mathcal{S}$.

Table 2: Comparison of the optimization subspaces induced by MeZO, LoZO and KronZO per layer of the model.

Method	Frozen for ν steps	Subspace $\mathcal{S}$	$\dim(\mathcal{S})$	Perturbation rank
MeZO	–	$\{Z^l \in \mathbb{R}^{m^l \times n^l}\}$	$m^l n^l$	$\min(m^l, n^l)$
LoZO	$V^l \in \mathbb{R}^{n^l \times r}$	$\{U^l V^{l\top}, U^l \in \mathbb{R}^{m^l \times r}\}$	$m^l r$	$\le r$
KronZO	$B^l \in \mathbb{R}^{m_2^l \times n_2^l}$	$\{A^l \otimes B^l, A^l \in \mathbb{R}^{m_1^l \times n_1^l}\}$	$m_1^l n_1^l$	$\operatorname{rank}(A^l)\operatorname{rank}(B^l)$

MeZO does not freeze anything, so each step is sampled from the *full* parameter space of size $m^l n^l$ for each layer, and rank $\min(m^l, n^l)$, which guarantees maximal directional coverage. This benefit comes at a cost: either MeZO stores the entire perturbation tensor (memory–intensive) or it keeps only the random seed and regenerates the tensor on demand (memory–light but requiring additional flops).

LoZO freezes a right factor $V^l \in \mathbb{R}^{n^l \times r}$. During the next ν steps every perturbation has the form $U_i^l (V^l)^\top$, confining optimization to a low-rank subspace of dimension $m^l r$. Because $\operatorname{rank}(U_i^l (V^l)^\top) \le r \ll n^l$, LoZO strongly reduces variance and memory, yet necessarily ignores any gradient component not in $\operatorname{span}(V^l)$, a bias that can slow down pretraining where informative subspaces are high-dimensional.

KronZO freezes one Kronecker factor $B^l \in \mathbb{R}^{m_2^l \times n_2^l}$ and varies only $A^l \in \mathbb{R}^{m_1^l \times n_1^l}$. The admissible updates $Z^l = A^l \otimes B^l$ live in a structured subspace of dimension $m_1^l n_1^l \ll m^l n^l$ (strong compression) but still almost surely have *full rank* whenever A^l and B^l have full rank (13). This Kronecker compression keeps the memory overhead negligible while preserving rich directional coverage, offering a better bias–variance trade-off than LoZO and explaining KronZO's slightly faster convergence and lower final loss in pretraining, even without the directional update, see Section Comparison with state-of-the-art.

In short, both LoZO and KronZO optimize within a fixed lower-dimensional subspace for ν iterations: LoZO by explicit rank truncation, KronZO by Kronecker structure. Yet, only KronZO retains the potential for full-rank updates, which is crucial for thoroughly exploring the high-dimensional loss landscape encountered during large-scale pretraining.

4 Convergence results

In this section, we derive convergence results for KronZO. As noted in Section Parameters update, when $q > 1$, the stochastic-gradient estimator used by KronZO is biased. Although this bias speeds up the optimization process in practice, see Section Computational results, unbiasedness is required for our convergence arguments [7, 24]. Therefore, we restrict attention to the case $q = 1$, i.e., one sampled direction per iteration. For clarity, we also assume that every step is accepted; equivalently, the `if` branch at Line 16 of Algorithm 2 is never entered and the update in Line 17 always occur. Finally, we consider a single-layer model with $m \times n$ parameters, but the analysis still holds with multiple layers.

Our convergence analysis builds upon the following standard regularity conditions.

Assumption 1. The stochastic gradient $\nabla\mathcal{L}(\theta;\xi)$ satisfies:

1. **Lipschitz continuity.** There exists $L > 0$ such that for all $(\theta_1, \theta_2) \in \mathbb{R}^{m\times n}$,
$$\left\|\nabla\mathcal{L}(\theta_1) - \nabla\mathcal{L}(\theta_2)\right\|_F \leq L\left\|\theta_1 - \theta_2\right\|_F.$$
2. **Unbiasedness and bounded variance.** There exists $\sigma \geq 0$ such that for all θ,
$$\mathbb{E}_\xi[\nabla\mathcal{L}(\theta;\xi)] = \nabla\mathcal{L}(\theta), \quad \mathbb{E}_\xi\left[\|\nabla\mathcal{L}(\theta;\xi) - \nabla\mathcal{L}(\theta)\|_{\mathrm{F}}^2\right] \leq \sigma^2.$$

In particular, Assumption 1 ensures that $\mathcal{L}$ is continuously differentiable with a Lipschitz continuous gradient, which is a standard requirement in stochastic gradient analysis. Under the setup introduced in this section, the optimization problem reads

$$\theta^* \in \arg\min_{\theta\in\mathbb{R}^{m\times n}} \mathbb{E}_\xi[\mathcal{L}(\theta;\xi)]\,,$$

which is (1).

It is worth noting that (1) corresponds to the case of a fixed training dataset, whose randomness is represented by ξ. The complete formulation is

$$\theta^* \in \arg\min_{\theta\in\mathbb{R}^{m\times n}} \mathbb{E}_{X,Y}\left[\mathbb{E}_{\xi|(X,Y)}[\mathcal{L}(\theta;\xi,X,Y)]\right]$$

where the **outer expectation** is taken with respect to the *data-generating distribution* of (X, Y), that is, the unknown probability distribution on the input–output pairs from which training datasets are drawn. The **inner expectation** is then taken over any additional source of randomness ξ (such as minibatch sampling, data augmentation, or dropout) conditional on having fixed a particular dataset (X, Y). For simplicity, it is common to work only with the inner expectation, i.e., to solve the problem for a given dataset, which reduces to (1).

Consider $q = 1$ and let $A \in \mathbb{R}^{m_1\times n_1}$, $B \in \mathbb{R}^{m_2\times n_2}$ such that $m_1m_2 = m$, $n_1n_2 = n$ and $\epsilon > 0$. Denote $Z = A \otimes B \in \mathbb{R}^{m\times n}$, the KronZO estimator reads

$$\widehat{\nabla}\mathcal{L}(\theta;\xi,Z) = \frac{\mathcal{L}(\theta+\epsilon Z) - \mathcal{L}(\theta-\epsilon Z)}{2\epsilon}Z. \tag{14}$$

KronZO resamples B every ν iterations and A at each iteration, then updates

$$\theta \leftarrow \theta - \alpha\widehat{\nabla}\mathcal{L}(\theta;\xi,Z). \tag{15}$$

We first show in Lemma 1 that under this setup, the stochastic-gradient estimator (14) is unbiased, which is necessary for the analysis. Then, following the analysis in LoZO [7], we show that KronZO is a subspace minimization method that solves a sequence of K subproblems, each one for ν iterations. We bound the expected improvement on a single subproblem (27) and use it to establish Theorem 1. Complete details are provided in Appendix.

Theorem 1. *Let K the number of subproblems solved and $T = K\nu$ the total number of iterations. Let Assumption 1 be satisfied. Denote $\tilde{L} = \|B\|_{\mathrm{F}}^2 L$ and $\bar{L} = \mathbb{E}_B\left[\tilde{L}\right] = m_2 n_2 L$. Denote $\tilde{\sigma}^2 = \|B\|_{\mathrm{F}}^2 \sigma^2$ and $\bar{\sigma}^2 = \mathbb{E}_B\left[\tilde{\sigma}^2\right] = m_2 n_2 \sigma^2$. If the step condition*

$$0 < \alpha < \frac{-28 + \sqrt{28^2 + 144}}{288 \bar{L} m_1 n_1 \nu}$$

holds, the iterates $\{\theta_t\}_{t \in [0,T]}$ generated by KronZO respect the following bound:

$$\begin{aligned}\beta \frac{m_2 n_2}{K} \sum_{k=0}^{K-1} \|\nabla \mathcal{L}(\theta_{k\nu})\|_{\mathrm{F}}^2 \leq & \frac{\mathcal{L}(\theta_0) - \mathcal{L}^*}{T\alpha} \\ & + 536\nu^2 \bar{L}^4 \alpha^2 \epsilon^2 m_1^2 n_1^2 + 24\nu^3 \bar{L}^2 \alpha^3 \bar{\sigma}^2 \\ & + \frac{\bar{L}^2 m_1 n_1 \epsilon^2}{2} + 66 \bar{L}^3 m_1^3 n_1^3 \epsilon^2 \alpha + 12 \bar{\sigma}^2 \bar{L} m_1 n_1 \alpha \end{aligned} \tag{16}$$

where $\beta = \left(\frac{1}{4} - 28\bar{L} m_1 n_1 \nu \alpha - 144 \alpha^2 \nu^2 m_1^2 n_1^2 \bar{L}^2\right) > 0$.

See proof on page 22. Inequality (16) bounds a standard stationarity surrogate, namely the average squared gradient along the iterates:

$$G_K := \frac{1}{K} \sum_{k=0}^{K-1} \|\nabla \mathcal{L}(\theta_{k\nu})\|_{\mathrm{F}}^2 \leq C_T$$

for some nonnegative quantity C_T that depends on $T = K\nu$ and on the algorithmic parameters. In the absence of specific choices for α and ϵ, the right-hand side need not vanish as $T \to \infty$, so convergence to stationarity is not guaranteed.

If $\alpha = \epsilon = T^{-1/2}$ and T is large enough to satisfy the step size condition $0 < \alpha = T^{-1/2} < \frac{-28+\sqrt{28^2+144}}{288\, \bar{L} m_1 n_1 \nu}$ (so that $\beta > 0$), then

$$\begin{aligned}\beta \frac{m_2 n_2}{K} \sum_{k=0}^{K-1} \|\nabla \mathcal{L}(\theta_{k\nu})\|_{\mathrm{F}}^2 \leq & \frac{\mathcal{L}(\theta_0) - \mathcal{L}^*}{T^{1/2}} + \frac{536\, \nu^2\, \bar{L}^4\, m_1^2 n_1^2}{T^2} + \frac{24\, \nu^3\, \bar{L}^2\, \bar{\sigma}^2}{T^{3/2}} \\ & + \frac{\bar{L}^2\, m_1 n_1}{2T} + \frac{66\, \bar{L}^3\, m_1^3 n_1^3}{T^{3/2}} + \frac{12\, \bar{\sigma}^2\, \bar{L}\, m_1 n_1}{T^{1/2}} = \mathcal{O}\left(T^{-1/2}\right). \end{aligned} \tag{17}$$

Consequently, for sufficiently small α and ϵ, the average squared gradient norms of KronZO's iterates admit an upper bound that decays to zero at a sublinear rate as T grows, coinciding with the LoZO rate [7].

This decay has the usual stationarity implications. Let G_K be as above. From (17), $G_K \to 0$ as $T \to \infty$, hence

$$\min_{0 \leq k < K} \|\nabla \mathcal{L}(\theta_{k\nu})\|_{\mathrm{F}}^2 \leq G_K \to 0,$$

so at least one iterate is asymptotically stationary. Moreover, by Markov's inequality, for any $\varepsilon > 0$,

$$\frac{1}{K} \left| \left\{ 0 \leq k < K : \|\nabla \mathcal{L}(\theta_{k\nu})\|_{\mathrm{F}}^2 > \varepsilon \right\} \right| \leq \frac{G_K}{\varepsilon} \to 0,$$

which shows that the algorithm spends an increasing fraction of iterations near stationarity. These guarantees are in the average sense and do not by themselves imply convergence of the entire sequence $\{\theta_t\}$. In practice, choosing ϵ small and a diminishing step size (e.g., $\alpha_t \propto t^{-1/2}$) ensures the average stationarity measure converges to zero.

5 Computational results

In this section, we present our setup and results on KronZO versus existing state-of-the-art ZO methods.

Experimental setup

All experiments were run on a single NVIDIA A100 with 80GB of GPU memory. We pretrained GPT-2 Small (125M parameters) [26] on the OpenWebText corpus [5] with a step budget $K = 10,000$.

The model uses a batch size of 32 and a context length of 1024, following the reference configuration in `nanoGPT` [19], with *no* dropout and *no* bias terms. Hyper-parameters are selected by a coarse grid search: $\epsilon \in [10^{-4}, 5 \times 10^{-4}]$; a cosine learning-rate schedule with initial value $\alpha = 3 \times 10^{-4}$. For LoZO, we keep the hyper-parameter choices of [7] (rank $r = 2$, refresh period $\nu = 50$). KronZO instead sets a shorter refresh period of $\nu = 5$, which introduces only a negligible overhead because the auxiliary matrix $\mathbf{B}$ is inexpensive to generate; it also maintains a history loss buffer of size $h = 10$.

Cost per step

MeZO and LoZO rely on antithetic sampling: one perturbation requires two objective evaluations, $\mathcal{L}(\boldsymbol{\theta} + \epsilon \mathbf{Z}_i)$ and $\mathcal{L}(\boldsymbol{\theta} - \epsilon \mathbf{Z}_i)$, so each step costs $2q$ forward passes to compute gradient estimator (6) and (10). KronZO re-uses the same pair to compute a scaling coefficient c_i for $i = 1, 2, \ldots, q$, then evaluates the candidate $\mathcal{L}(\boldsymbol{\theta} - \alpha c_i \mathbf{Z}_i)$; each sample therefore costs three forward passes, i.e., $3q$ per step.

Directional versus averaging updates

Table 3 compares the classical *averaging* update (q-RGE (6)) with its *directional* counterpart that we presented in Section Parameters update. To equalize wall-clock effort, the directional variants use $q' = \lceil 2q/3 \rceil$ samples so that the total number of forward passes remains identical. Both variants store a single perturbation vector, hence their memory footprint is the same.

Table 3 confirms that substituting the averaging step with our directional update systematically enhances optimization. Hence, directional update yields richer descent directions and markedly lower objectives across all methods with equal total evaluation budget. The restricted optimization subspace explored by LoZO does not adequately span the broad singular-value spectrum characteristic of pretraining, which accounts for its slightly reduced effectiveness. The extra run-time for LoZO and KronZO stems from their additional linear-algebra steps ($U^l V^{l\top}$ and $A^l \otimes B^l$, respectively); MeZO avoids this at the cost of higher memory.

Table 3: Averaging vs. directional updates under a fixed forward-pass budget.

Method	Variant	Step budget q	Memory (GB)	Time/iter (s)	Final loss
MeZO	Averaging	50	1.0	6.7	8.62
	Directional	33	1.0	6.7	**7.01**
LoZO	Averaging	50	0.6	7.2	8.82
	Directional	33	0.6	7.2	**7.53**
KronZO	Averaging	50	0.5	7.4	8.63
	Directional	33	0.5	7.4	<u>**6.95**</u>

Comparison with state-of-the-art

Table 4 collects the best state-of-the-art variant of each ZO method and contrasts it with first-order SGD. Starting from an untrained loss of about 11, MeZO and LoZO achieve a $\approx 20\%$ reduction, while KronZO reaches $\approx 37\%$ with an even smaller memory footprint than LoZO.

Table 4: Resource usage and final loss. KronZO matches LoZO's memory footprint while markedly improving optimization performance.

Method	Step budget q	Memory (GB)	Time/iter (s)	Final loss
SGD (1st-order)	—	1.5	0.3	3.50
Untrained	—	—	—	11.0
MeZO	50	1.0	6.7	8.62
LoZO	50	0.6	7.2	8.82
KronZO	33	0.5	7.5	**6.95**
KronZO (GPT-2 XL 1.5B)	33	6.4	66.1	8.21
SGD (GPT-2 XL 1.5B)	—	20	2.6	2.85

SGD converges more quickly and attains a lower loss, but it consumes about three-times the GPU memory, and the discrepancy grows with model size. The last two rows of Table 4, which cover a 1.55-billion-parameter model confirm that KronZO still requires only a fraction of the first-order memory. Its training efficiency, however, degrades at this scale, as expected: in very high dimensions the likelihood that a randomly sampled direction aligns well with the true stochastic gradient becomes vanishingly small. Designing more scalable direction-selection schemes to restore efficiency on billion-parameter LMs is therefore a key avenue for future work.

Figure 6 plots the memory footprint across the OPT family [32]: the KronZO over AdamW memory ratio increases from 7 times on OPT-1.3B to roughly 11 times on OPT-30B. Results were extrapolated from [24], assuming $s_{\text{ZO}} = 1$, corresponding to storing the full matrix $\mathbf{Z}$. These results make ZO methods such as KronZO a compelling choice whenever memory is the binding resource.

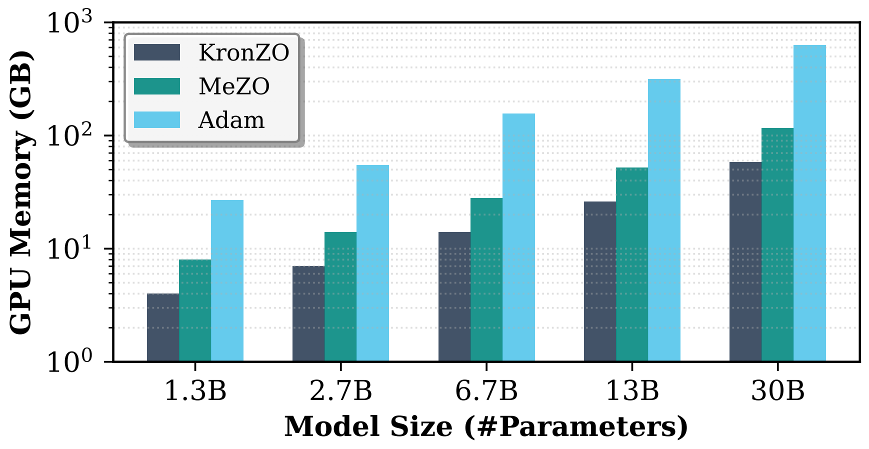

Figure 6: Memory usage for KronZO, MeZO and classic first-order training with Adam on different sizes of OPT model in logarithmic scale.

6 Conclusion and future work

We analyzed pretraining stochastic gradients through *stable rank* and *participation ratio*, revealing that the gradients effective dimensionality is higher than suggested by the stable rank alone. Consequently, low-rank ZO methods ignore critical directions in a pretraining context. On the other hand, existing full-rank ZO methods add significant memory or flops overhead. KronZO bridges this gap by pairing (i) a Kronecker parameterization that preserves rich subspaces with (ii) a directional criterion-based update, leading to better convergence. The result is a ZO optimizer that, under equal forward-pass budgets, outperforms MeZO and LoZO while using less memory. This work significantly extends the preliminary study of Allaire et al. [1], scaling to a model six times larger and surpassing state-of-the-art

ZO methods in pretraining performance. Although KronZO still trails SGD in final loss, we believe that ZO methods have the potential to reach performance comparable to first-order methods during pretraining on medium-sized models, while preserving their exceptionally low memory footprint, at the cost of a higher training time.

Practical takeaways

Use participation ratio for pretraining. It provides a faithful estimate of the number of active directions in LMs and should guide the choice of ZO rank.

Prefer directional update. Replacing averaging by a directional test yields systematic gains.

Choose KronZO when RAM is scarce. At fixed memory, it enables training large models more efficiently than full-rank ZO baselines.

Limitations and future work

KronZO increases the number of forward passes by 50% ($3q$ vs. $2q$ per iteration). While this overhead is acceptable on inexpensive computations (for example on small models), latency-critical scenarios may require further pruning, or better exploiting the $3q$ query budget per step, e.g., via *secant-constrained updates*.

The method is also highly sensitive to the hyper-parameters choice, which were selected via a coarse grid search. Automated tuning methods, such as the Mesh Adaptive Direct Search (MADS) [4], could markedly enhance efficiency.

A natural next step is to investigate techniques for closing the loss gap in KronZO that widens at larger scales, such as ZO-Kronecker momentum, or ZO parameter ϵ schedule.

Finally, we did not evaluate KronZO in the fine-tuning setting; given its strong performance for zeroth-order pretraining, it may also prove effective for fine-tuning tasks.

Declarations

Ethical Approval Not applicable

Funding This research was supported by the following NSERC Discovery Grants: RGPIN–2024–05086 (Le Digabel), RGPIN–2020–06535 (Orban), and RGPIN–2025–06911 (Partovi Nia). On behalf of all authors, the corresponding author states that there is no conflict of interest.

Availability of data and materials The datasets analyzed in this study are publicly available: the Shakespeare dataset [19] and the OpenWebText dataset [5].

7 Appendix

This section is dedicated to the convergence analysis of KronZO. We first prove the unbiasedness of the estimator (14), then show that KronZO can be seen as a subspace optimization problem before showing Theorem 1 from Convergence results.

7.1 Unbiasedness of KronZO

Definition 1 (Unbiased estimator)**.** A gradient estimator with parameters P, $\widehat{\nabla}\mathcal{L}(\theta;\xi,P)$ is *unbiased* if

$$\mathbb{E}_P\left[\widehat{\nabla}\mathcal{L}(\theta;\xi;P)\right] = \nabla\mathcal{L}(\theta;\xi).$$

Lemma 1. *The gradient estimator in KronZO is unbiased. For all $\theta \in \mathbb{R}^{m\times n}, \xi \in \mathbb{R}$,*

$$\mathbb{E}_Z\left[\widehat{\nabla}\mathcal{L}(\theta;\xi,Z)\right] = \nabla\mathcal{L}(\theta;\xi).$$

Proof. Using vectorized notation,

$$\lim_{\epsilon\to 0} vec(\widehat{\nabla}\mathcal{L}(\theta;\xi,Z,\epsilon)) = vec(Z)vec(Z^\top)vec(\nabla\mathcal{L}(\theta;\xi)).$$

Taking the expectation on Z,

$$\mathbb{E}_Z\left[vec(\widehat{\nabla}\mathcal{L}(\theta;\xi))\right] = \mathbb{E}_Z\left[vec(Z)vec(Z^\top)vec(\nabla\mathcal{L}(\theta;\xi))\right] = \Sigma \times vec(\nabla\mathcal{L}(\theta;\xi)),$$

with $\Sigma = \mathbb{E}_Z\left[vec(Z)vec(Z)^\top\right]$ the covariance matrix of Z.

Each component of Z is sampled from $\mathcal{N}(0,1)$ therefore $\Sigma = I_{mn}$ which leads to

$$\mathbb{E}_Z\left[\widehat{\nabla}\mathcal{L}(\theta;\xi,Z,\epsilon)\right] = \nabla\mathcal{L}(\theta;\xi).$$

Definition 1 concludes the proof. □

7.2 KronZO viewed as subspace minimization

Let $\phi : \mathbb{R}^{p\times q} \to \mathbb{R}^{m\times n}$ be a *fixed* linear map that defines the search subspace.

$$\phi(X) = \begin{cases} X\,V^\top & \text{for LoZO with fixed } U \\ X \otimes B & \text{for KronZO with fixed } B. \end{cases}$$

Define

$$\mathcal{G}_\theta(X;\xi) := \mathcal{L}\big(\theta + \phi(X);\xi\big), \qquad \mathcal{G}_\theta(X) := \mathbb{E}_\xi[\mathcal{G}_\theta(X;\xi)]\,.$$

For a search direction A we abbreviate the two–point estimator

$$\widehat{\nabla}\mathcal{G}_\theta(X;\xi,A) \;:=\; \frac{\mathcal{G}_\theta(X+\epsilon A;\xi) - \mathcal{G}_\theta(X-\epsilon A;\xi)}{2\epsilon}\,A.$$

To solve (1), KronZO iteratively solves the following subproblem for ν iterations with fixed matrix B:

$$X^* \in \arg\min_{X\in\mathbb{R}^{m_1\times n_1}} \mathcal{G}_\theta(X). \tag{18}$$

At subproblem k, (18) is solved by following the update rule

$$X_{k,s+1} = X_{k,s} - \alpha\widehat{\nabla}_X\mathcal{L}(\tilde{\theta}_k + X_{k,s}\otimes B_k;\xi_{k,s}), \qquad s = 0,1,\ldots,\nu-1, \tag{19}$$

$$\tilde{\theta}_{k+1} = \tilde{\theta}_k + X_{k,\nu}\otimes B_k. \tag{20}$$

Here, $X_{k,0} = \mathbf{0}$, and

$$\widehat{\nabla}_X\mathcal{L}(\tilde{\theta}_k + X_{k,s}\otimes B_k;\xi_{k,s}) = \frac{\mathcal{L}(\tilde{\theta}_k + (X_{k,s}+\epsilon A_{k,s})\otimes B_k;\xi_{k,s}) - \mathcal{L}(\tilde{\theta}_k + (X_{k,s}-\epsilon A_{k,s})\otimes B_k;\xi_{k,s})}{2\epsilon}A_{k,s},$$

where $A_{k,s}$ is sampled from a normal distribution every iteration. Let $Y_{k,s} := \tilde{\theta}_k + X_{k,s}\otimes B_k$. Using the update rule (19) to derive the update of Y gives

$$Y_{k,s+1} = Y_{k,s} - \alpha\frac{\mathcal{L}(Y_{k,s}+\epsilon A_{k,s}\otimes B_k;\xi_{k,s}) - \mathcal{L}(Y_{k,s}-\epsilon A_{k,s}\otimes B_k;\xi_{k,s})}{2\epsilon}A_{k,s}\otimes B_k, \tag{21}$$

which is exactly (15). Details on the equivalence between methods such as LoZO and KronZO and the subspace minimization problem (18) can be found in [7, Section 4.2].

7.3 Convergence analysis

The analysis is divided in two parts. First, we derive a bound on the subproblem iterations in Lemma 7, then use it to derive an upper bound on the whole optimization process in Theorem 1. In the remainder of this section, we suppose Assumption 1 is satisfied.

Lemma 2 (Lipschitz continuity in the X-space)**.** *Let $B \in \mathbb{R}^{m_2 \times n_2}$ be fixed and define $\phi(X) = X \otimes B$. Under Assumption 1,*

$$X \longmapsto \nabla_X \mathcal{G}_\theta(X)$$

is $\widetilde{L}$–Lipschitz w.r.t. the Frobenius norm, where

$$\widetilde{L} \; = \; L \, \|B\|_F^2.$$

Where L is the Lipschitz constant of $\theta \mapsto \nabla_\theta \mathcal{L}(\theta)$ from Assumption 1.

Proof. Set $g(X) := \theta + X \otimes B$, so that $\mathcal{G}_\theta(X) = \mathcal{L}(g(X))$. For any perturbation ΔX, by the chain rule,

$$\mathrm{d}\mathcal{G}_\theta(X)[\Delta X] = \big\langle \nabla_\theta \mathcal{L}\big(g(X)\big), \, \Delta X \otimes B \big\rangle_F.$$

By Frobenius duality, $\nabla_X \mathcal{G}_\theta(X) = \mathcal{C}_B\big(\nabla_\theta \mathcal{L}\big(g(X)\big)\big)$, where the contraction $\mathcal{C}_B$ (adjoint of $X \mapsto X \otimes B$) satisfies $\|\mathcal{C}_B\|_{\mathrm{op}} \le \|B\|_F$.

Let X_1, X_2 and $Y_k = g(X_k)$. Then

$$\|\nabla_X \mathcal{G}_\theta(X_1) - \nabla_X \mathcal{G}_\theta(X_2)\|_F \le \|B\|_F \, \|\nabla_\theta \mathcal{L}(Y_1) - \nabla_\theta \mathcal{L}(Y_2)\|_F \le L \, \|B\|_F \, \|Y_1 - Y_2\|_F.$$

Since $\|Y_1 - Y_2\|_F = \|(X_1 - X_2) \otimes B\|_F = \|X_1 - X_2\|_F \, \|B\|_F$,

$$\|\nabla_X \mathcal{G}_\theta(X_1) - \nabla_X \mathcal{G}_\theta(X_2)\|_F \le L \, \|B\|_F^2 \, \|X_1 - X_2\|_F. \qquad \square$$

Lemma 3 (Variance bound of the projected gradient)**.** *Let $B \in \mathbb{R}^{m_2 \times n_2}$ be fixed and define $\phi(X) = X \otimes B$. Under Assumption 1, for any $X \in \mathbb{R}^{m_1 \times n_1}$,*

$$\mathbb{E}_\xi \Big[\|\nabla_X \mathcal{G}_\theta(X; \xi) - \nabla_X \mathcal{G}_\theta(X)\|_F^2 \Big] \le \widetilde{\sigma}^2$$

where $\widetilde{\sigma}^2 := \sigma^2 \|B\|_F^2$ and σ^2 is the variance bound from Assumption 1.

Proof. With $\phi(X) = X \otimes B$ and the Frobenius duality (see the proof of Lemma 2) we have

$$\nabla_X \mathcal{G}_\theta(X; \xi) \; = \; \mathcal{C}_B\big(\nabla_\theta \mathcal{L}(\theta + X \otimes B; \xi)\big),$$

where the *contraction operator* $\mathcal{C}_B : \mathbb{R}^{(m_1 m_2) \times (n_1 n_2)} \longrightarrow \mathbb{R}^{m_1 \times n_1}$ is defined block–wise by

$$\big[\mathcal{C}_B(T)\big]_{ij} := \langle T_{(ij)}, \, B \rangle_F, \qquad T_{(ij)} \in \mathbb{R}^{m_2 \times n_2} \text{ is the } (i,j)\text{–block of } T.$$

For any block matrix T

$$\big\|\mathcal{C}_B(T)\big\|_F^2 \le \|B\|_F^2 \sum_{i,j} \|T_{(ij)}\|_F^2 = \|B\|_F^2 \|T\|_F^2,$$

hence $\|\mathcal{C}_B\|_{\mathrm{op}} \le \|B\|_F$.

Define the zero-mean random matrix

$$\Delta(\xi) := \nabla_\theta \mathcal{L}(\theta + X \otimes B; \xi) - \nabla_\theta \mathcal{L}(\theta + X \otimes B).$$

By Assumption 1, $\mathbb{E}_\xi\left[\|\Delta(\xi)\|_{\mathrm{F}}^2\right] \leq \sigma^2$.

$$\begin{aligned}\mathbb{E}_\xi\left[\|\nabla_X \mathcal{G}_\theta(X;\xi) - \nabla_X \mathcal{G}_\theta(X)\|_{\mathrm{F}}^2\right] &= \mathbb{E}_\xi\left[\left\|\mathcal{C}_B\big(\Delta(\xi)\big)\right\|_{\mathrm{F}}^2\right] \\ &\leq \|\mathcal{C}_B\|_{\mathrm{op}}^2\, \mathbb{E}_\xi\left[\|\Delta(\xi)\|_{\mathrm{F}}^2\right] \\ &\leq \|B\|_{\mathrm{F}}^2\, \sigma^2.\end{aligned}$$

Therefore $\mathbb{E}_\xi\left[\|\nabla_X \mathcal{G}_\theta(X;\xi) - \nabla_X \mathcal{G}_\theta(X)\|_{\mathrm{F}}^2\right] \leq \sigma^2\, \|B\|_{\mathrm{F}}^2$, which completes the proof. □

Lemma 4. *Under Assumption 1, for the gradient estimator $\widehat{\nabla}\mathcal{G}_\theta(X;\xi)$, the following bound holds:*

$$\mathbb{E}_A\left[\left\|\widehat{\nabla}\mathcal{G}_\theta(X;\xi)\right\|_{\mathrm{F}}^2\right] \leq \frac{\tilde{L}(m_1n_1+4)^3\epsilon^2}{4} + 6m_1^2n_1^2\,\|\nabla\mathcal{G}_\theta(X;\xi)\|_{\mathrm{F}}^2 .$$

Proof. First,

$$\mathbb{E}_A\left[\left\|\widehat{\nabla}\mathcal{G}_\theta(X;\xi)\right\|_{\mathrm{F}}^2\right] \leq 2\mathbb{E}_A\left[\left\|\widehat{\nabla}\mathcal{G}_\theta(X;\xi) - \langle\nabla\mathcal{G}_\theta(X;\xi),A\rangle A\right\|_{\mathrm{F}}^2\right] + 2\mathbb{E}_A\left[\|\langle\nabla\mathcal{G}_\theta(X;\xi),A\rangle A\|_{\mathrm{F}}^2\right]. \quad (22)$$

For the first term in Equation 22, we use the Taylor inequality along with Lemma 2:

$$\begin{aligned}|\mathcal{G}_\theta(X+\epsilon A;\xi) - \mathcal{G}_\theta(X) - \epsilon\langle\nabla\mathcal{G}_\theta(X;\xi),A\rangle| &\leq \frac{\tilde{L}\epsilon^2}{2}\,\|A\|_{\mathrm{F}}^2 , \\ |\mathcal{G}_\theta(X-\epsilon A;\xi) - \mathcal{G}_\theta(X) + \epsilon\langle\nabla\mathcal{G}_\theta(X;\xi),A\rangle| &\leq \frac{\tilde{L}\epsilon^2}{2}\,\|A\|_{\mathrm{F}}^2 .\end{aligned}$$

Combining both inequalities give

$$|\mathcal{G}_\theta(X+\epsilon A;\xi) - \mathcal{G}_\theta(X-\epsilon A;\xi) - 2\epsilon\langle\nabla\mathcal{G}_\theta(X;\xi),A\rangle| \leq \tilde{L}\epsilon^2\,\|A\|_{\mathrm{F}}^2 .$$

Dividing both sides by 2ϵ, multiplying by $\|A\|_F$ and taking the square give

$$\left\|\widehat{\nabla}\mathcal{G}_\theta(X;\xi) - \langle\nabla\mathcal{G}_\theta(X;\xi),A\rangle A\right\|_{\mathrm{F}}^2 \leq \frac{\tilde{L}\epsilon^2}{4}\|A\|_F^6.$$

Taking the expectation on A and multiplying by two give

$$2\mathbb{E}_A\left[\left\|\widehat{\nabla}\mathcal{G}_\theta(X;\xi) - \langle\nabla\mathcal{G}_\theta(X;\xi),A\rangle A\right\|_{\mathrm{F}}^2\right] \leq \frac{\tilde{L}\epsilon^2}{2}\mathbb{E}_A\left[\|A\|_F^6\right].$$

Since $A : m_1 \times n_1$ has i.i.d $\mathcal{N}(0,1)$ entries, the Frobenius norm satisfies $\|A\|_{\mathrm{F}}^2 \sim \chi^2(m_1n_1)$. Then, $\mathbb{E}_A\left[\|A\|_F^6\right] = \mathbb{E}_A\left[(\|A\|_{\mathrm{F}}^2)^3\right] = \mathbb{E}_A\left[(\chi^2(m_1n_1))^3\right] = m_1n_1(m_1n_1+2)(m_1n_1+4) \leq (m_1n_1+4)^3$.

Eventually,

$$2\mathbb{E}_A\left[\left\|\widehat{\nabla}\mathcal{G}_\theta(X;\xi) - \langle\nabla\mathcal{G}_\theta(X;\xi),A\rangle A\right\|_{\mathrm{F}}^2\right] \leq \frac{\tilde{L}(m_1n_1+4)^3\epsilon^2}{4}.$$

The second term in Equation 22 can be calculated directly with the Cauchy-Schwarz inequality:

$2\mathbb{E}_A\left[\|\langle\nabla\mathcal{G}_\theta(X;\xi),A\rangle A\|_{\mathrm{F}}^2\right] \leq 2\,\|\nabla\mathcal{G}_\theta(X;\xi)\|_{\mathrm{F}}^2\,\mathbb{E}_A\left[\|A\|_F^4\right] \leq 6m_1^2n_1^2\,\|\nabla\mathcal{G}_\theta(X;\xi)\|_{\mathrm{F}}^2$, which completes the proof. □

We now introduce the *Gaussian smoothing function*:

$$\mathcal{G}_\theta^\epsilon(X) = \mathbb{E}_A[\mathcal{G}_\theta(X+\epsilon A)] = \frac{1}{\kappa}\int \mathcal{G}_\theta(X+\epsilon A)e^{-\frac{1}{2}\left\|A\right\|_{\mathrm{F}}^2}\,dA,$$

where κ is the normalization parameter: $\kappa = \int e^{-\frac{1}{2}\|A\|_{\mathrm{F}}^2}\,dA$.

Lemma 5. *For the Gaussian smoothing function $\mathcal{G}_\theta^\epsilon(X)$, the following properties hold:*

- $\mathbb{E}_{A,\xi}\left[\widehat{\nabla}\mathcal{G}_\theta(X;\xi)\right] = \nabla\mathcal{G}_\theta^\epsilon(X)$,
- $\mathcal{G}_\theta^\epsilon(X)$ *is* $\tilde{L}$*-smooth,*
- $|\mathcal{G}_\theta^\epsilon(X) - \mathcal{G}_\theta(X)| \leq \frac{\tilde{L}m_1n_1\epsilon^2}{2}$,
- $\|\nabla\mathcal{G}_\theta^\epsilon(X) - \nabla\mathcal{G}_\theta(X)\|_{\mathrm{F}}^2 \leq \tilde{L}^2 m_1 n_1 \epsilon^2$.

Proof. The three first properties can be found in [25, Section 2]. The proof of the last claim can be found in [7, Lemma B.3]. □

Lemma 6. *If the step size condition $36\tilde{L}^2\alpha^2\nu m_1^2 n_1^2 \leq \frac{1}{\nu-1}$ is satisfied, then for any $t \leq \nu$, the following bound holds:*

$$\mathbb{E}_{A,\xi}\left[\|X_t - X_0\|_{\mathrm{F}}^2\right] \leq 144\alpha^2\nu^2 m_1^2 n_1^2 \mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta(X_0;\xi_0)\|_{\mathrm{F}}^2\right] + 536\alpha^2\tilde{L}^2\epsilon^2 m_1^3 n_1^3 \nu^2 + 24\nu^2\alpha^2\tilde{\sigma}^2. \tag{23}$$

Proof. Let $t \in [0, \nu-1]$. By definition of X_{t+1},

$$\mathbb{E}_{A,\xi}\left[\|X_{t+1} - X_0\|_{\mathrm{F}}^2\right] = \mathbb{E}_{A,\xi}\left[\left\|X_t - \alpha\widehat{\nabla}\mathcal{G}_\theta(X_t;\xi_t) - X_0\right\|_{\mathrm{F}}^2\right] \tag{24}$$

$$\leq \mathbb{E}_{A,\xi}\left[\|X_t - \alpha\nabla\mathcal{G}_\theta^\epsilon(X_t) - X_0\|_{\mathrm{F}}^2\right] + \alpha^2\mathbb{E}_{A,\xi}\left[\left\|\nabla\mathcal{G}_\theta^\epsilon(X_t) - \widehat{\nabla}\mathcal{G}_\theta(X_t;\xi_t)\right\|_{\mathrm{F}}^2\right]. \tag{25}$$

Since the squared Frobenius norm is convex, for any function ϕ and for any $c \in (0,1)$,

$$\begin{aligned}\|\phi(x)+\phi(y)\|_{\mathrm{F}}^2 &= \left\|c\frac{\phi(x)}{c} + (1-c)\frac{\phi(y)}{1-c}\right\|_{\mathrm{F}}^2 \\ &\leq c\left\|\frac{\phi(x)}{c}\right\|_{\mathrm{F}}^2 + (1-c)\left\|\frac{\phi(y)}{1-c}\right\|_{\mathrm{F}}^2 \\ &= \frac{1}{c}\|\phi(x)\|_{\mathrm{F}}^2 + c\|\phi(y)\|_{\mathrm{F}}^2.\end{aligned}$$

For the first term in (25), applying the previous convexity inequality with $c = \frac{\nu-1}{\nu} \in (0,1)$ gives

$$\mathbb{E}_{A,\xi}\left[\|X_t - \alpha\nabla\mathcal{G}_\theta^\epsilon(X_t) - X_0\|_{\mathrm{F}}^2\right] \leq (1+\frac{1}{\nu-1})\mathbb{E}_{A,\xi}\left[\|X_t - X_0\|_{\mathrm{F}}^2\right] + \nu\mathbb{E}_{A,\xi}\left[\|\alpha\nabla\mathcal{G}_{\theta^\epsilon}(X_t)\|_{\mathrm{F}}^2\right].$$

For the second term, we reuse the convexity inequality to get

$$\alpha^2\mathbb{E}_{A,\xi}\left[\left\|\nabla\mathcal{G}_\theta^\epsilon(X_t) - \widehat{\nabla}\mathcal{G}_\theta(X_t;\xi_t)\right\|_{\mathrm{F}}^2\right] \leq \frac{\nu\alpha^2}{\nu-1}\mathbb{E}_{A,\xi}\left[\left\|\widehat{\nabla}\mathcal{G}_\theta(X_t,\xi_t)\right\|_{\mathrm{F}}^2\right] + \alpha^2\nu\mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta^\epsilon(X_t)\|_{\mathrm{F}}^2\right].$$

Combining the two previous inequalities gives

$$\mathbb{E}_{A,\xi}\left[\|X_{t+1} - X_0\|_{\mathrm{F}}^2\right] \leq (1+\frac{1}{\nu-1})\mathbb{E}_{A,\xi}\left[\|X_t - X_0\|_{\mathrm{F}}^2\right] + 2\alpha^2\nu\mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta^\epsilon(X_t)\|_{\mathrm{F}}^2\right]$$

$$+ \alpha^2 \big(1 + \frac{1}{\nu - 1}\big) \mathbb{E}_{A,\xi} \left[\left\| \widehat{\nabla} \mathcal{G}_\theta(X_t, \xi_t) \right\|_{\mathrm{F}}^2 \right]. \tag{26}$$

To bound the second term in (26), we add and subtract a compensatory term along with Jensen's inequality to get

$$\begin{aligned} 2\alpha^2 \nu \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta^\epsilon(X_t)\|_{\mathrm{F}}^2 \right] =& 2\alpha^2 \nu \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta^\epsilon(X_t) - \nabla \mathcal{G}_\theta(X_t) + \nabla \mathcal{G}_\theta(X_t) - \nabla \mathcal{G}_\theta(X_t; \xi_t) + \nabla \mathcal{G}_\theta(X_t; \xi_t)\|_{\mathrm{F}}^2 \right] \\ \leq& 6\nu\alpha^2 \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta^\epsilon(X_t) - \nabla \mathcal{G}_\theta(X_t)\|_{\mathrm{F}}^2 \right] + 6\nu\alpha^2 \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta(X_t) - \nabla \mathcal{G}_\theta(X_t; \xi_t)\|_{\mathrm{F}}^2 \right] \\ &+ 6\nu\alpha^2 \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta(X_t; \xi_t)\|_{\mathrm{F}}^2 \right]. \end{aligned}$$

Now, Lemma 4 and Lemma 5 give

$$2\alpha^2 \nu \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta^\epsilon(X_t)\|_{\mathrm{F}}^2 \right] \leq 6\nu\alpha^2 \tilde{L}^2 m_1 n_1 \epsilon^2 + 6\nu\alpha^2 \tilde{\sigma}^2 + 6\nu\alpha^2 \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta(X_t; \xi_t)\|_{\mathrm{F}}^2 \right].$$

To bound the third term in (26), using Lemma 4 and noticing $m_1^2 n_1^2 \geq m_1 n_1 \geq 1$,

$$\begin{aligned} \alpha^2 \big(1 + \frac{1}{\nu - 1}\big) \mathbb{E}_{A,\xi} \left[\left\| \widehat{\nabla} \mathcal{G}_\theta(X_t, \xi_t) \right\|_{\mathrm{F}}^2 \right] &\leq 2\alpha^2 \mathbb{E}_{A,\xi} \left[\left\| \widehat{\nabla} \mathcal{G}_\theta(X_t, \xi_t) \right\|_{\mathrm{F}}^2 \right] \\ &\leq 2\alpha^2 \frac{\tilde{L}(m_1 n_1 + 4)^3 \epsilon^2}{4} + 12\alpha^2 m_1^2 n_1^2 \|\nabla \mathcal{G}_\theta(X; \xi)\|_{\mathrm{F}}^2 \\ &\leq 128\alpha^2 \tilde{L}^2 \epsilon^2 m_1^3 n_1^3 \nu + 12\alpha^2 m_1^2 n_1^2 \|\nabla \mathcal{G}_\theta(X; \xi)\|_{\mathrm{F}}^2. \end{aligned}$$

We can now bound the right term in (26):

$$\begin{aligned} \mathbb{E}_{A,\xi} \left[\|X_{t+1} - X_0\|_{\mathrm{F}}^2 \right] \leq& \big(1 + \frac{1}{\nu - 1}\big) \mathbb{E}_{A,\xi} \left[\|X_t - X_0\|_{\mathrm{F}}^2 \right] \\ &+ (6\nu\alpha^2 + 12\alpha^2 m_1^2 n_1^2) \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta(X_t; \xi_t)\|_{\mathrm{F}}^2 \right] \\ &+ 6\nu\alpha^2 m_1^3 n_1^3 \epsilon^2 + 6\nu\alpha^2 \tilde{\sigma}^2 + 128\alpha^2 \tilde{L}^2 \epsilon^2 m_1^3 n_1^3. \end{aligned}$$

Using $\tilde{L}$-smoothness of $\nabla \mathcal{G}_\theta(X, \xi)$ from Lemma 2 and $\nu \geq 2$,

$$\begin{aligned} \mathbb{E}_{A,\xi} \left[\|X_{t+1} - X_0\|_{\mathrm{F}}^2 \right] \leq& \big(1 + \frac{1}{\nu - 1} + 2\tilde{L}^2 (6\nu\alpha^2 + 12\alpha^2 m_1^2 n_1^2)\big) \mathbb{E}_{A,\xi} \left[\|X_t - X_0\|_{\mathrm{F}}^2 \right] \\ &+ (12\alpha^2 m_1^2 n_1^2 \nu + 24\alpha^2 m_1^2 n_1^2 \nu) \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta(X_0; \xi_0)\|_{\mathrm{F}}^2 \right] \\ &+ 6\nu\alpha^2 \tilde{\sigma}^2 + 134\alpha^2 \tilde{L}^2 \epsilon^2 m_1^3 n_1^3 \nu. \end{aligned}$$

Now we simplify the expression

$$\begin{aligned} \mathbb{E}_{A,\xi} \left[\|X_{t+1} - X_0\|_{\mathrm{F}}^2 \right] \leq& \big(1 + \frac{1}{\nu - 1} + 36\tilde{L}^2 \alpha^2 \nu m_1^2 n_1^2\big) \mathbb{E}_{A,\xi} \left[\|X_t - X_0\|_{\mathrm{F}}^2 \right] \\ &+ 36\alpha^2 \nu m_1^2 n_1^2 \mathbb{E}_{A,\xi} \left[\|\nabla \mathcal{G}_\theta(X_0; \xi_0)\|_{\mathrm{F}}^2 \right] \\ &+ 134\alpha^2 \tilde{L}^2 \epsilon^2 m_1^3 n_1^3 \nu + 6\nu\alpha^2 \tilde{\sigma}^2. \end{aligned}$$

Applying the step size condition $36\tilde{L}^2 \alpha^2 \nu m_1^2 n_1^2 \leq \frac{1}{\nu - 1}$,

$$\mathbb{E}_{A,\xi} \left[\|X_{t+1} - X_0\|_{\mathrm{F}}^2 \right] \leq \big(1 + \frac{2}{\nu - 1}\big) \mathbb{E}_{A,\xi} \left[\|X_t - X_0\|_{\mathrm{F}}^2 \right]$$

$$+ 36\alpha^2\nu m_1^2 n_1^2 \mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta(X_0;\xi_0)\|_{\mathrm{F}}^2\right]$$
$$+ 134\alpha^2\tilde{L}^2\epsilon^2 m_1^3 n_1^3\nu + 6\nu\alpha^2\tilde{\sigma}^2.$$

By induction, noting that $\sum_{s=0}^{t-1}(1+\frac{2}{\nu-1}) \le 4\nu$ for $t \le \nu$,

$$\begin{aligned}\mathbb{E}_{A,\xi}\left[\|X_t - X_0\|_{\mathrm{F}}^2\right] &\le \sum_{s=0}^{t-1}(1+\frac{2}{\nu-1})\left(36\alpha^2\nu m_1^2 n_1^2\mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta(X_0;\xi_0)\|_{\mathrm{F}}^2\right] + 134\alpha^2\tilde{L}^2\epsilon^2 m_1^3 n_1^3\nu + 6\nu\alpha^2\tilde{\sigma}^2\right)\\ &\le 144\alpha^2\nu^2 m_1^2 n_1^2\mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta(X_0;\xi_0)\|_{\mathrm{F}}^2\right] + 536\alpha^2\tilde{L}^2\epsilon^2 m_1^3 n_1^3\nu^2 + 24\nu^2\alpha^2\tilde{\sigma}^2,\end{aligned}$$

which concludes the proof. □

In Lemma 6, the step size condition $36\tilde{L}^2\alpha^2\nu m_1^2 n_1^2 \le \frac{1}{\nu-1} \implies \alpha \le \frac{1}{6\tilde{L}m_1 n_1(\nu-1)}$, which is a weaker condition in expectation than $\alpha < \frac{-28+\sqrt{28^2+144}}{288\tilde{L}m_1 n_1\nu}$. Therefore, if α satisfies condition for Theorem 1, the condition in Lemma 6 is also satisfied.

Lemma 7. *The following bound holds for subproblem* (18)*:*

$$\begin{aligned}\mathbb{E}_{A,\xi}[\mathcal{G}_\theta(X_\nu) - \mathcal{G}_\theta(X_0)] \le &\left(-\frac{\nu\alpha}{4} + 28\tilde{L}m_1 n_1\nu^2\alpha^2 + 144\alpha^3\nu^3 m_1^2 n_1^2\tilde{L}^2\right)\mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta(X_0)\|_{\mathrm{F}}^2\right]\\ &+ 536\nu^3\tilde{L}^4\alpha^3\epsilon^2 m_1^2 n_1^2 + 24\nu^3\tilde{L}^2\alpha^3\tilde{\sigma}^2\\ &+ \frac{\tilde{L}^2 m_1 n_1\epsilon^2\alpha\nu}{2} + 66\tilde{L}^3 m_1^3 n_1^3\epsilon^2\nu\alpha^2 + 12\tilde{\sigma}^2\tilde{L}m_1 n_1\alpha^2\nu. \qquad (27)\end{aligned}$$

Proof. The proof is very similar to [7, Lemma B.5], therefore we rewrite in our notation only the important steps of the proof. First, using $\tilde{L}$-smoothness of $\mathcal{G}_\theta^\epsilon$, and the inequality $X_\nu - X_0 = -\alpha\sum_{t=0}^{\nu-1}\widehat{\nabla}\mathcal{G}_\theta(X_t;\xi_t)$,

$$\begin{aligned}\mathbb{E}_{A,\xi}[\mathcal{G}_\theta(X_\nu) - \mathcal{G}_\theta(X_0)] &\le \mathbb{E}_{A,\xi}[\langle\nabla\mathcal{G}_\theta^\epsilon(X_0), X_\nu - X_0\rangle] + \frac{\tilde{L}}{2}\mathbb{E}_{A,\xi}\left[\|X_\nu - X_0\|_{\mathrm{F}}^2\right]\\ &= -\alpha\sum_{t=0}^{\nu-1}\mathbb{E}_{A,\xi}\left[\langle\nabla\mathcal{G}_\theta^\epsilon(X_0), \widehat{\nabla}\mathcal{G}_\theta(X_t;\xi_t)\rangle\right] + \frac{\alpha^2\tilde{L}}{2}\mathbb{E}_{A,\xi}\left[\left\|\sum_{t=0}^{\nu-1}\widehat{\nabla}\mathcal{G}_\theta()X_t;\xi_t\right\|_{\mathrm{F}}^2\right].\end{aligned} \qquad (28)$$

The first term in (28) can be bounded using $\langle a,b\rangle \le \frac{\|a^2\|+\|b\|^2}{2}$ along with Lemma 2 and Lemma 5 to get

$$\begin{aligned}-\alpha\sum_{t=0}^{\nu-1}\mathbb{E}_{A,\xi}\left[\langle\nabla\mathcal{G}_\theta^\epsilon(X_0), \widehat{\nabla}\mathcal{G}_\theta(X_t;\xi_t)\rangle\right] \le &-\frac{\alpha\nu}{4}\mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta(X_0)\|_{\mathrm{F}}^2\right]\\ &+ \frac{\tilde{L}^2\alpha}{2}\sum_{t=0}^{\nu-1}\mathbb{E}_{A,\xi}\left[\|X_t - X_0\|_{\mathrm{F}}^2\right]\\ &+ \frac{\tilde{L}^2 m_1 n_1\epsilon^2\alpha\nu}{2}.\end{aligned} \qquad (29)$$

On the other hand, using Lemma 4 and Lemma 5

$$\frac{\alpha^2\tilde{L}}{2}\mathbb{E}_{A,\xi}\left[\left\|\sum_{t=0}^{\nu-1}\widehat{\nabla}\mathcal{G}_\theta(X_t;\xi_t)\right\|_{\mathrm{F}}^2\right] \le 28\tilde{L}m_1 n_1\nu^2\alpha^2\mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta(X_0)\|_{\mathrm{F}}^2\right]$$

$$+ 28\tilde{L}^3 m_1 n_1 \nu\alpha^2 \sum_{t=0}^{\nu-1} \mathbb{E}_{A,\xi}\left[\|X_t - X_0\|_{\mathrm{F}}^2\right]$$
$$+ 66\tilde{L}^3 m_1^3 n_1^3 \epsilon^2 \nu\alpha^2 + 12\tilde{\sigma}^2 \tilde{L} m_1 n_1 \alpha^2 \nu. \tag{30}$$

Combining (29) and (30) gives

$$\begin{aligned}\mathbb{E}_{A,\xi}[\mathcal{G}_\theta(X_\nu) - \mathcal{G}_\theta(X_0)] \le &\left(-\frac{\nu\alpha}{4} + 28\tilde{L}m_1 n_1 \nu^2\alpha^2 + 144\alpha^3\nu^3 m_1^2 n_1^2 \tilde{L}^2\right) \mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{G}_\theta(X_0)\|_{\mathrm{F}}^2\right] \\ &+ 536\nu^3\tilde{L}^4\alpha^3\epsilon^2 m_1^2 n_1^2 + 24\nu^3\tilde{L}^2\alpha^3\tilde{\sigma}^2 \\ &+ \frac{\tilde{L}^2 m_1 n_1 \epsilon^2 \alpha\nu}{2} + 66\tilde{L}^3 m_1^3 n_1^3 \epsilon^2\nu\alpha^2 + 12\tilde{\sigma}^2\tilde{L}m_1 n_1\alpha^2\nu. \qquad \square\end{aligned}$$

Now we have established the bound for solving the subproblem (18). Next, we investigate the impact of updating θ and resampling B and establish the convergence result for our proposed KronZO algorithm. This leads to Theorem 1.

Proof of Theorem 1. Recalling the update rule for the subproblem, it follows that

$$\mathcal{G}_{\theta_{k\nu}}(X_\nu) = \mathcal{L}(\theta_{k\nu} + X_{(k,\nu)} \otimes B_k) = \mathcal{L}(\theta_{(k+1)\nu}), \quad \mathcal{G}_{\theta_{k\nu}}(X_0) = \mathcal{L}(\theta_{k\nu}).$$

Moreover, note that $\nabla\mathcal{G}_{\theta_{k\nu}}(X_0) = \nabla\mathcal{L}(\theta_{k\nu})\otimes B_k$. Denote $\bar{L} = \mathbb{E}_B\left[\tilde{L}\right] = m_2 n_2 L$ and $\bar{\sigma}^2 = \mathbb{E}_B\left[\tilde{\sigma}^2\right] = m_2 n_2 \sigma^2$. Applying Lemma 7 gives

$$\begin{aligned}\mathbb{E}_{A,B,\xi}\left[\mathcal{L}(\theta_{(k+1)\nu} - \mathcal{L}(\theta_{k\nu})\right] \le &\left(-\frac{\nu\alpha}{4} + 28\bar{L}m_1 n_1 \nu^2\alpha^2 + 144\alpha^3\nu^3 m_1^2 n_1^2 \bar{L}^2\right) \mathbb{E}_{A,\xi}\left[\|\nabla\mathcal{L}(\theta_{k\nu})\otimes B_k\|_{\mathrm{F}}^2\right] \\ &+ 536\nu^3\bar{L}^4\alpha^3\epsilon^2 m_1^2 n_1^2 + 24\nu^3\bar{L}^2\alpha^3\bar{\sigma}^2 \\ &+ \frac{\bar{L}^2 m_1 n_1 \epsilon^2 \alpha\nu}{2} + 66\bar{L}^3 m_1^3 n_1^3 \epsilon^2\nu\alpha^2 + 12\bar{\sigma}^2\bar{L}m_1 n_1\alpha^2\nu.\end{aligned}$$

Using identity $\|P\otimes Q\|_{\mathrm{F}}^2 = \|P\|_{\mathrm{F}}^2\,\|Q\|_{\mathrm{F}}^2$ and $\mathbb{E}\left[\|B_k\|_{\mathrm{F}}^2\right] = m_2 n_2$, dividing by $\nu\alpha K$ on both sides then summing over K and rearranging the terms give

$$\begin{aligned}\beta\frac{m_2 n_2}{K}\sum_{k=0}^{K-1}\|\nabla\mathcal{L}(\theta_{k\nu})\|_{\mathrm{F}}^2 \le &\frac{\mathcal{L}(\theta_0) - \mathcal{L}^*}{T\alpha} + \\ &+ 536\nu^2\bar{L}^4\alpha^2\epsilon^2 m_1^2 n_1^2 + 24\nu^3\bar{L}^2\alpha^3\bar{\sigma}^2 \\ &+ \frac{\bar{L}^2 m_1 n_1\epsilon^2}{2} + 66\bar{L}^3 m_1^3 n_1^3\epsilon^2\alpha + 12\bar{\sigma}^2\bar{L}m_1 n_1\alpha,\end{aligned}$$

which is (16). Moreover, letting $a := \bar{L}m_1 n_1 \nu\alpha$, the quantity $\beta = \frac{1}{4} - 28a - 144a^2$ is a concave quadratic function of a with two real roots. Therefore, $\beta > 0$ if and only if

$$0 < a < \frac{-28 + \sqrt{28^2 + 144}}{288},$$

which is equivalent to the step condition

$$0 < \alpha < \frac{-28 + \sqrt{28^2 + 144}}{288\,\bar{L}m_1 n_1 \nu}. \qquad \square$$

References

[1] N. Allaire, M. Ghazvini Nejad, S. Le Digabel, and V. Partovi Nia. Zeroth order optimization for pretraining language models. In Proceedings of the 14th International Conference on Pattern Recognition Applications and Methods – Volume 1: ICPRAM, pages 113–121, Porto, Portugal, 2025. SciTePress. doi: 10.5220/0013261100003905.

[2] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. Neurocomputing, 5(4-5): 185–196, 1993. doi: https://doi.org/10.1016/0925-2312(93)90006-O.

[3] C. Audet and W. Hare. Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland, 2017. doi: 10.1007/978-3-319-68913-5.

[4] Charles Audet and J. E. Dennis. Mesh adaptive direct search algorithms for constrained optimization. SIAM Journal on Optimization, 17(1):188–217, 2006. doi: 10.1137/040603371.

[5] Stella Biderman, Leo Gao, Joel Lehman, Eric Hallahan, Cory Zue, Scott Gray, Kyle McDonell, Jason Phang, Samuel Weinbach, and Ellie Pavlick. Openwebtext corpus. https://skylion007.github.io/OpenWebTextCorpus/, 2019.

[6] Julius R. Blum. Multidimensional stochastic approximation methods. The annals of mathematical statistics, pages 737–744, 1954. doi: 10.1214/aoms/1177728659.

[7] Y. Chen, Y. Zhang, L. Cao, K. Yuan, and Z. Wen. Enhancing zeroth-order fine-tuning for language models with low-rank structures, 2024. URL https://arxiv.org/abs/2410.07698.

[8] Amit Dhurandhar, Tejaswini Pedapati, Avinash Balakrishnan, Pin-Yu Chen, Karthikeyan Shanmugam, and Ruchir Puri. Model agnostic contrastive explanations for structured data, 2019. URL https://arxiv.org/abs/1906.00117.

[9] 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, 2014. URL https://arxiv.org/abs/1312.2139.

[10] Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. bioRxiv, 2017. doi: 10.1101/214262.

[11] Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. SIAM Journal on Optimization, 23(4):2341–2368, 2013. doi: 10.1137/120880811.

[12] Alexander Graham. Kronecker products and matrix calculus with applications. Courier Dover Publications, Mineola, NY, 2018.

[13] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8342–8360, Online, hosted in Seattle, WA, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740.

[14] M.G.A Hameed, A. Mosleh, M.S Tahaei, and V. Partovi Nia. Sekron: A decomposition method supporting many factorization structures, 2022. URL https://arxiv.org/abs/2210.06299.

[15] O. C. Herfindahl. Concentration in the Steel Industry. PhD thesis, Columbia University, 1950. URL https://archive.org/details/herfindahl-concentration-in-the-steel-industry-1950-publish. PhD thesis; accessible via Internet Archive (published online 2021).

[16] 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, 2021. URL https://arxiv.org/abs/2106.09685.

[17] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information, 2018. URL https://arxiv.org/abs/1804.08598.

[18] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

[19] Andrej Karpathy. Nanogpt. https://github.com/karpathy/nanoGPT, 2023. GitHub repository.

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

[21] D. Lakhmiri, S. Le Digabel, and C. Tribes. HyperNOMAD: Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search. ACM Transactions on Mathematical Software, 47(3), 2021. doi: 10.1145/3450975. URL https://dx.doi.org/10.1145/3450975.

[22] Brett W. Larsen, Stanislav Fort, Nic Becker, and Surya Ganguli. How many degrees of freedom do we need to train deep networks: a loss landscape perspective, 2022. URL https://arxiv.org/abs/2107.05802.

[23] Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes, 2018. URL https://arxiv.org/abs/1804.08838.

[24] S. Malladi, T. Gao, E. Nichani, A. Damian, J.D Lee, D. Chen, and S. Arora. Fine-tuning language models with just forward passes, 2024. URL https://arxiv.org/abs/2305.17333.

[25] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. Foundations of Computational Mathematics, 17(2):527–566, 2017. doi: 10.1007/s10208-015-9296-2.

[26] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. Technical report, OpenAI.

[27] James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE transactions on automatic control, 37(3):332–341, 1992. doi: 10.1109/9.119632.

[28] C. Tribes, S. Benarroch-Lelong, P. Lu, and I. Kobyzev. Hyperparameter Optimization for Large Language Model Instruction-Tuning. Technical Report G-2023-62, Les cahiers du GERAD, 2023.

[29] Chun-Chen Tu, Paishun Ting, Pin-Yu Chen, Sijia Liu, Huan Zhang, Jinfeng Yi, Cho-Jui Hsieh, and Shin-Ming Cheng. Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks, 2020. URL https://arxiv.org/abs/1805.11770.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. CoRR, 2017. URL http://arxiv.org/abs/1706.03762.

[31] L. Zhang, B. Li, K.K Thekumparampil, S. Oh, M. Muehlebach, and N. He. Zeroth-order optimization finds flat minima, 2025. URL https://arxiv.org/abs/2506.05454.

[32] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068.

[33] 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. URL https://arxiv.org/abs/2402.11592.

[34] Pu Zhao, Sijia Liu, Pin-Yu Chen, Nghia Hoang, Kaidi Xu, Bhavya Kailkhura, and Xue Lin. On the design of black-box adversarial examples by leveraging gradient-free optimization and operator splitting method, 2019. URL https://arxiv.org/abs/1907.11684.