SUMO: <u>S</u>ubspace-Aware <u>M</u>oment-<u>O</u>rthogonalization for Accelerating Memory-Efficient LLM Training

Yehonathan Refael
Faculty of Engineering
Tel Aviv University
yehonathan@tau.ac.il

Guy Smorodinsky
Department of Computer science
Ben Gurion University
smorodin@post.bgu.ac.il

Tom Tirer
Faculty of Engineering
Bar-Ilan University
tirer.tom@biu.ac.il

Ofir Lindenbaum
Faculty of Engineering
Bar-Ilan University
ofir.lindenbaum@biu.ac.il

Abstract

Low-rank gradient-based optimization methods have significantly improved memory efficiency during the training of large language models (LLMs), enabling operations within constrained hardware without sacrificing performance. However, these methods primarily emphasize memory savings, often overlooking potential acceleration in convergence due to their reliance on standard isotropic steepest descent techniques, which can perform suboptimally in the highly anisotropic landscapes typical of deep networks, particularly LLMs. In this paper, we propose SUMO (<u>S</u>ubspace-Aware <u>M</u>oment-<u>O</u>rthogonalization), an optimizer that employs exact singular value decomposition (SVD) for moment orthogonalization within a dynamically adapted low-dimensional subspace, enabling norm-inducing steepest descent optimization steps. By explicitly aligning optimization steps with the spectral characteristics of the loss landscape, SUMO effectively mitigates approximation errors associated with commonly used methods, such as the Newton-Schulz orthogonalization approximation. We theoretically establish an upper bound on these approximation errors, proving their dependence on the condition numbers of moments, conditions we analytically demonstrate are encountered during LLM training. Furthermore, we both theoretically and empirically illustrate that exact orthogonalization via SVD substantially improves convergence rates while reducing overall complexity. Empirical evaluations confirm that SUMO accelerates convergence, enhances stability, improves performance, and reduces memory requirements by up to 20% compared to state-of-the-art methods.

1 Introduction

Low-rank gradient-based optimization methods have become powerful tools for reducing memory consumption during the pre-training and fine-tuning of large language models (LLMs), often without sacrificing performance, and sometimes even improving it. For instance, while pre-training Llama 7B typically requires around 58GB of memory, far exceeding the 24GB available on consumer GPUs like RTX 4090, recent advances, such as those discussed in [1–3], have demonstrated that Llama 7B can now be trained from scratch on a single 24GB GPU without the need for costly memory offloading. The theoretical analysis in [1] attributes this efficiency to the inherent low-rank structure of the gradients, which enables optimization in a significantly reduced latent space. Furthermore, [2] found a consistent decrease in gradient rank throughout training, suggesting that low-rank optimization not only reduces memory usage but also converges toward increasingly compact subspaces.

However, despite these advancements, existing methods primarily focus on memory savings and often overlook the potential to accelerate convergence. Current approaches typically rely on standard steepest descent methods and assume isotropic geometry, which can hinder efficiency in ill-conditioned settings. This observation motivates our primary objective: to develop a subspace-aware optimizer that leverages low-rank structure while adapting to the geometry of the loss landscape. By reevaluating the choice of norm and its influence on gradient descent dynamics, we aim to design an algorithm that improves generalization, accelerates convergence, while preserving the memory advantages of low-rank methods.

Classical gradient descent, including SGD [4], performs steepest descent under the Euclidean norm, which reflects isotropic curvature. However, deep networks exhibit highly anisotropic loss landscapes, making this assumption suboptimal. Recent work shows that adaptive optimizers, such as Shampoo [5], SOAP [6], and Muon [7], can be interpreted as steepest descent under non-Euclidean norms tailored to the network architecture and data structure. As shown in [8], these methods implicitly adapt to spectral or operator norms, which better capture local curvature and improve convergence. This motivates the design of subspace-aware optimizers that exploit both low-rank structure and appropriate geometry to accelerate training.

To formalize the role of geometry in optimization, consider a neural network with a differentiable loss function $\mathcal{L} : \mathcal{W} \to \mathbb{R}$ defined on a weight space $\mathcal{W} = \mathbb{R}^n$. The local behavior around a point $\mathbf{w}$ can be approximated by the Taylor expansion, $\mathcal{L}(\mathbf{w} + \Delta\mathbf{w}) \approx \mathcal{L}(\mathbf{w}) + \mathbf{g}^\top \Delta\mathbf{w} + \frac{\lambda}{2}\|\Delta\mathbf{w}\|^2$, where $\mathbf{g} = \nabla_\mathbf{w}\mathcal{L}(\mathbf{w})$, $\lambda > 0$ captures the sharpness or curvature of the loss surface and $\|\cdot\|$ is a chosen norm reflecting the geometry of the optimization landscape. Minimizing this approximation corresponds precisely to performing steepest descent under the given norm constraint. According to [9], the solution to this minimization explicitly takes the form,

$$\Delta\mathbf{w} = -\frac{\|\mathbf{g}\|_*}{\lambda} \operatorname*{argmax}_{\mathbf{t}:\|\mathbf{t}\|=1} \mathbf{g}^\top \mathbf{t},$$

where $\|\cdot\|_*$ denotes the dual norm of $\|\cdot\|$. Adaptive optimizers differ primarily in their norm choices. Adam utilizes a dynamic Max-of-Max norm constraint. Recent optimizers consider matrix norms while applying steepest descent at the layer level. Muon imposes a fixed Schatten-p norm constraint for large p, effectively using the spectral norm on weight matrices [10, 7]. Shampoo [5] dynamically learns the optimal approximate Schatten-p norm for steepest descent, with its variants like SOAP [11] apply momentum to efficiently navigate the space of possible norms. Muon, by contrast, operates within a relatively fixed but large Schatten-p norm, balancing the dynamic adaptability of Shampoo with static spectral-norm constraints. Since neural network weights locally act as linear operators on Euclidean spaces, the induced operator (spectral) norm provides a natural constraint that aligns with the curvature of the loss surface. This perspective motivates gradient orthogonalization, which ensures that optimization updates respect the spectral norm, thereby controlling perturbation magnitude and enhancing optimization stability and efficiency [9].

While norm-induced optimization methods offer a principled way to align updates with the geometry of the loss landscape, their practical deployment often incurs substantial computational overhead. For instance, Shampoo requires computing matrix inverses or root operations at every optimization step, which can be computationally expensive for large-scale neural networks. Similarly, Muon's first-order moments-orthogonalization, though effective, involves a costly approximation to spectral decompositions, computed by applying five iterations of Newton-Schulz [12] (referred to as Newton-Schulz5). Therefore, there is an inherent trade-off between the theoretical optimality provided by these norm-induced optimization approaches and their practical computational demands.

To bridge the gap between the geometric advantages of norm-induced methods and their computational costs, we first analyze the limitations of existing approximations. We derive an upper bound on the error introduced by the Newton-Schulz orthogonalization, demonstrating that this error increases with the condition number of the moment matrix. This finding explains the increasing instability of the Newton–Schulz5 method in ill-conditioned scenarios, which we subsequently demonstrate to occur in the first-order moment matrices during the training of large language models (LLMs). Building on this analysis, we establish a convergence rate for Muon optimization and compare it to an alternative method that replaces the Newton–Schulz approach with exact Singular Value Decomposition (SVD). Remarkably, we find that the SVD-based approach converges faster, with improvements directly proportional to the accumulated errors from the moments' orthogonalization by the Newton–Schulz5 method. Motivated by the empirical observation that gradients in LLMs often exhibit a low-rank

structure, especially during early training, we propose a subspace-aware optimization scheme. This scheme performs exact SVD-based moment orthogonalization within a low-dimensional adaptive subspace. This approach benefits from the relatively low computational cost of SVD calculations for low-rank input matrices and enhances convergence stability. Also, our approach attains an even greater reduction in memory usage than all previous low-rank training methods by relying solely on the first-order moment, as detailed below in Table 1. We support our method with a theoretical convergence guarantee and validate its empirical benefits through experiments, demonstrating faster training and better model performance compared to existing methods.

Table 1: Comparison of properties between SUMO, GaLore, Adam, Shampoo, and SOAP. Assume $\mathbf{W} \in \mathbb{R}^{m \times n}$ with $m \geq n$, a constant projection rank r and a subspace update rate K.

	SUMO	Adam	Shampoo	SOAP	GaLore
Computation	$O(mnr + mn^2/K)$	$O(mn)$	$O(m^3 + n^3)$	$O(m^3 + n^3)$	$O(mnr + mn^2/K)$
Optim. states memory	$nr + mr$	$2mn$	$m^2 + n^2$	$2mn + 2m^2 + 2n^2$	$2nr + mr$
Subspace-aware	✓	✗	✗	✗	✓
Orthogonalization	✓	✗	✗	✗	✗

2 Related Work

Low-rank gradient optimization. Low-rank gradients naturally emerge during neural network training, as shown in both theoretical and empirical studies [13–15]. This structure has been leveraged to reduce memory and computational costs during training [16–18]. Recent work [2] showed that gradients in reversible layers [19] tend to collapse to rank one over time and used this to adaptively adjust gradient rank in Adam. In this paper, we demonstrate that the same low-rank trend is present in the first-order moment, which we utilize to efficiently apply exact orthogonalization—avoiding the accumulation of approximation errors, such as those encountered in Newton-Schultz, during optimization.

Memory efficient optimizers. Reducing the memory demands of training large language models (LLMs) has driven extensive algorithmic research. One research direction, initiated by LoRA [20], reduces the number of trainable parameters via low-rank adaptation. Yet, such methods often fall short of fully parameterized models, especially during pre-training. Another direction does not restrict the set of trainable parameters but instead optimizes the training methods, with notable examples including AdaRankGrad, GaLore, Fira, Flora, Adam-mini, GaLore-mini, LDAdam, GoLore, LoQT, and Apollo [1–3, 21–25], integrating low-rank gradient projections in optimization. In this work, we reduce memory usage even further by relying solely on a first-order momentum, as shown in Table 1.

Gradient preconditioning. Preconditioning the Gradient method is critical in enhancing the efficiency and effectiveness of optimizers. Several notable approaches for using a preconditioner have emerged, including methods based on signed gradients [26–29], gradient clipping [30], normalization [30, 31], and gradient whitening [7, 32–37]. Recent studies [7, 38] explored gradient-orthogonalization strategies, thereby speeding up training. Orthogonalizing gradients effectively constrains updates to lie on directions of uniform magnitude (spectral radius = 1), preventing updates from exaggerating specific gradient directions over others. This procedure ensures a form of normalization that mitigates potential instabilities from ill-conditioned gradients. Unlike these methods, which apply preconditioning or approximate orthogonalization in the high-dimensional parameter space, our approach performs exact SVD-based orthogonalization within an adaptively selected low-rank subspace, offering improved stability and lower computational overhead.

Orthogonal Stochastic Gradient Descent with Momentum (OSGDM). OSGDM [38] is a recently introduced first-order optimization method that speeds up neural network training by orthogonalizing gradients before the optimization step. Specifically, for a data batch $\xi^{(t)}$, OSGDM applies SVD to the gradient matrix $\mathbf{G}_l^{(t)} = \nabla_{\mathbf{W}_l} \mathcal{L}\left(\Phi\left(\xi^{(t)}; \theta\right)\right)$ of each neural network layer l to generate an orthonormal gradient approximation $\mathbf{O}_l$. This ensures diversity among learned representations and reduces redundancy. The update rule for OSGDM with momentum term γ and learning rate η is defined as,

$$\mathbf{O}_l^{(t)} = \mathrm{orth}\left(\mathbf{G}_l^{(t)}\right), \quad \mathbf{M}_l^{(t+1)} \leftarrow \gamma \mathbf{M}_l^{(t)} + \eta \mathbf{O}_l^{(t)}, \quad \mathbf{W}_l^{(t+1)} \leftarrow \mathbf{W}_l^{(t)} - \mathbf{M}_l^{(t+1)},$$

where $\mathrm{orth}(\mathbf{G}) = \left(\mathbf{G}\mathbf{G}^\top\right)^{-1/2}\mathbf{G}$ is the ortogonalization operator, and $\mathbf{M}_l$ is the first order moment of layer l. Despite the additional computational overhead of SVD, OSGDM empirically converges faster and achieves higher accuracy than common methods such as Adam.

Muon optimizer. At iteration t, given weight $\mathbf{W}^{(t)}$, momentum μ, learning rate η_t, and objective $\mathcal{L}_t$, Muon, introduced by [7], constructs the update rule,

$$\mathbf{M}^{(t)} = \mu\mathbf{M}^{(t-1)} + \mathbf{G}_l^{(t)}, \quad \mathbf{O}_l^{(t)} = \mathrm{Newton\text{-}Schulz5}(\mathbf{M}^{(t)}), \quad \mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta^{(t)}\mathbf{O}_l^{(t)}.$$

Here, $\mathbf{M}^{(t)}$ is the momentum at iteration t, initialized as a zero matrix when $t = 0$. The Newton-Schulz5 method [35] approximates $(\mathbf{M}^{(t)}\mathbf{M}^{(t)\top})^{-1/2}\mathbf{M}^{(t)}$, orthogonalizing $\mathbf{M}^{(t)}$ and thus ensuring uniform update directions, avoiding dominance by few directions. Muon explicitly controls the norm of gradient updates—particularly the spectral norm (or Schatten-p norm with large p), which limits updates to smaller, well-conditioned steps in parameter space. By constraining the spectral norm, moment orthogonalization implicitly prevents overly large or ill-conditioned parameter updates. Such updates often lead to poor generalization due to instability or overfitting. Shortly after the introduction of Muon, the study in [39] proposed a framework to scale Muon for larger LLMs, mainly adding weight decay, and carefully adjusting the per-parameter update scale.

3 Method and Main Results

3.1 Theoretical Motivation: Exact moments orthogonalization leads to significantly faster convergence

Previous work on pre-training and fine-tuning large language models (LLMs) has primarily focused on reducing memory usage for constrained hardware or lowering computational cost (e.g., [1, 40, 3]). In this paper, we take a step toward accelerating LLM optimization by showing that applying exact orthogonalization (e.g., via SVD) to the first-order moment offers a practical advantage, even over the most accurate approximations, such as the commonly used Newton-Schulz5 method. Specifically, we find that SVD converges faster and incurs lower computational overhead. To support this, we first present a new observation: the moment matrix in LLM training tends to decrease in rank over time. Building on this, we derive an upper bound on the approximation error of Newton-Schulz5, showing that it depends on both the number of iterations and the matrix condition number, thereby highlighting its limitations in ill-conditioned or low-rank settings (precisely the case in LLM optimization moments). This motivates the need for more accurate orthogonalization of moment matrices during LLM training. Applying SVD directly to full-sized layers is generally impractical. The surprising result, however, is that when integrated into a low-rank optimization scheme, the use of SVD becomes not only feasible but preferable. We conclude with a convergence analysis of Muon optimization, which, under these conditions, converges significantly more slowly than the SVD-based alternative. To the best of our knowledge, our convergence analysis of Muon optimization is the first to avoid neglecting the error in the Newton-Schultz approximation [41]. The proofs of all lemmas and theorems of this section are relegated to the Appendix A.

Lemma 3.1 (Moment Becomes Low-Rank During Training). *Let $\mathbf{M}^{(t)} \in \mathbb{R}^{n \times m}$ denote the first moment of a reversible layer[1] in a moment-based optimization algorithm, updated according to $\mathbf{M}^{(t)} = \beta_1\mathbf{M}^{(t-1)} + \mathbf{G}^{(t)}$, where $\mathbf{G}^{(t)}$ is the gradient matrix at iteration t. Let $\mathbf{M}^{(t)} = \mathbf{U}^{(t)}\Sigma^{(t)}\mathbf{V}^{(t)\top}$ be the singular value decomposition (SVD) of $\mathbf{M}^{(t)}$, and define the rank-r orthogonal projection matrix as $\mathbf{P}^{(t)}(r) = \mathbf{U}^{(t)}[:, 1:r]\,\mathbf{U}^{(t)}[:, 1:r]^\top$. Then the relative error of the best rank-one approximation,*

$$\kappa_M(t) \triangleq \frac{\|\mathbf{M}^{(t)} - \mathbf{P}^{(t)}(1)\mathbf{M}^{(t)}\|_F^2}{\|\mathbf{M}^{(t)}\|_F^2}, \tag{1}$$

*satisfies $\kappa_M(t) \leq O(C^{-t})$ for some constant $C > 1$.*

The above result, in (1), implies that $\mathbf{M}^{(t)}$ approaches its rank-one approximation $\mathbf{P}^{(t)}(1)\mathbf{M}^{(t)}$, as the iteration number increases, namely, $\mathbf{M}^{(t)}$ becomes rank-one. The following Lemma 3.2 characterizes the impact of the moments' low-rank structure on the approximation error of the Newton-Schulz5 orthogonalization.

---

[1]Reversible networks are formally defined in Appendix B.1

Lemma 3.2 (Orthogonalization error $\mathcal{E}_i$). *For a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, let σ_1 be the largest singular value of $\mathbf{A}\mathbf{A}^\top$ and σ_m be the smallest (without the loss of generality, assume $m \leq n$). Let $r \leq m$ be the largest index where $\sigma_r > \sigma_{r+1} = \cdots = \sigma_m \geq 0$. Let $\kappa = \frac{\sigma_1}{\sigma_m}$ by the condition number of $\mathbf{A}\mathbf{A}^\top$. Denote $\mathcal{E}_i$ the error of Newton-Schultz after i iterations. Then we have*

$$\|\mathcal{E}_i\|_F \leq \sqrt{r} \cdot \left(1 - \frac{1}{\kappa}\right)^{2^i}. \tag{2}$$

According to the lemma, the approximation error grows exponentially with the condition number. Given the low-rank structure of the first-order moments, low-dimensional optimization can mitigate this error. Specifically, projecting the moment estimates $\hat{\mathbf{M}}^{(t)}$ onto their dominant (small) r-dimensional subspace ensures that the squared moment $\hat{\mathbf{M}}^{(t)}\hat{\mathbf{M}}^{(t)\top}$ is constructed using only the top r squared eigenvalues. These dominant components are significantly larger and exclude near-zero values, resulting in a substantially lower condition number compared to that of the full-rank squared moment matrix. This observation motivates the use of the Muon optimizer within a low-rank optimization framework for LLMs, including 2D reversible layers. Such an approach not only preserves the inherent memory efficiency of low-rank methods but also reduces approximation error during optimization, potentially leading to faster convergence and improved performance compared to full-dimensional training. However, we also empirically observe that the eigenvalues of the moment matrix decay gradually. As shown in Figure 1, even when projecting onto the dominant subspace, the resulting matrix $\hat{\mathbf{M}}^{(t)}\hat{\mathbf{M}}^{(t)\top}$, composed of the top $r = 16$ squared eigenvalues, can still exhibit a large condition number, thereby introducing non-negligible approximation error.
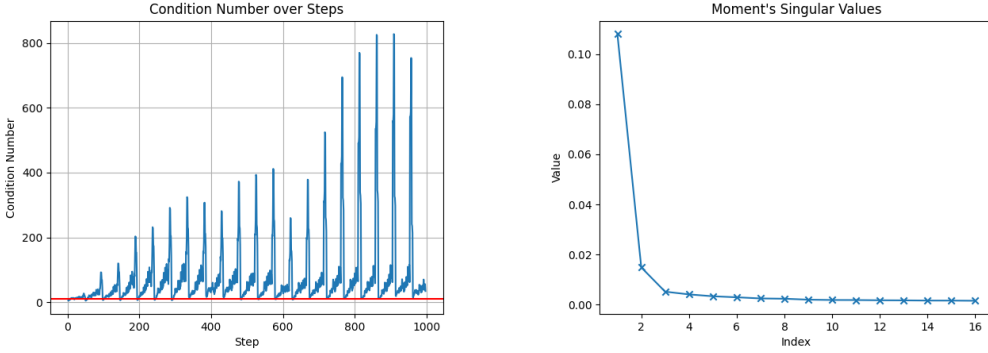
(a) Condition number of the first-order moment vs. training step. The red line marks value 10.

(b) Illustration of the moment's singular value decay, taken arbitrarily at step 100.

Figure 1: Evidence of anisotropy and ill-conditioning in the first-order moment matrix as a function of the Galore steps of the Roberta-base model [42] on the GLUE dataset RTE task [43]: (a) condition number growth, (b) spectral decay of moment.

To comprehend the cumulative error of Newton-Schulz5 orthogonalization at each optimization step, we proceed to derive the convergence rate of the Moun optimization. To that end, we now provide some notations. Consider a neural network denoted as $\Phi(\cdot; \boldsymbol{\theta})$, which consists of L layers and is parameterized by $\boldsymbol{\theta} \triangleq \left[\mathbf{W}_1^{d_1 \times d_0}, \ldots, \mathbf{W}_{L-1}^{d_{L-1} \times d_{L-2}}, \mathbf{W}_L^{d_L \times d_{L-1}}\right]$. Here, $\mathbf{W}_i$ represents the weights tensor parameters associated with the i-th layer, for $i \in [L]$. We denote the differential loss $\mathcal{L}$, where, with a slight abuse of notation, we write the training problem by $\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) = \mathbb{E}_\xi[\mathcal{L}(\Phi(\mathbf{W}, \xi))]$, if the context refers to the weights of a certain layer. We use the Frobenius norm, denoted $\|\cdot\|_F$, which is induced by the inner product $\langle \mathbf{X}, \mathbf{Y} \rangle = \text{tr}(\mathbf{X}^\top \mathbf{Y})$. Assume that the stochastic gradient $\nabla \mathcal{L}(\mathbf{W}, \xi)$ is an unbiased estimator of the full gradient $\nabla \mathcal{L}(\mathbf{W})$, with variance bounded by σ^2, i.e., $\mathbb{E}[\|\nabla \mathcal{L}(\mathbf{W}, \xi) - \nabla \mathcal{L}(\mathbf{W})\|_F^2] \leq \sigma^2$. Let $\mathcal{E}_i^{(t)} = \text{orth}(\mathbf{M}^{(t)}) - \text{Newton-Schulz}(\mathbf{M}^{(t)})$ denote the approximation error of the Newton-Schulz (with $i \geq 1$ iteration) at time t, where $\mathbf{M}^{(t)}$ denotes the moment at iteration t.

Lemma 3.3 (Exact convergence rate of Muon). *Consider the Muon optimizer update w.r.t layer $\mathbf{W} \in \mathbb{R}^{m \times n}$ defined by*

$$\mathbf{M}^{(t)} \leftarrow \beta \mathbf{M}^{(t-1)} + (1 - \beta)\mathbf{G}^{(t)}$$

$$\mathbf{O}^{(t)} \leftarrow \mathbf{U}^{(t)}\mathbf{V}^{(t)^\top} + \mathcal{E}_i^{(t)}, \quad (i \text{ iterations Newton-Schulz approximation })$$
$$\mathbf{W}^{(t+1)} \leftarrow \mathbf{W}^{(t)} - \eta_t \mathbf{O}^{(t)},$$

where $\mathbf{M}^{(t)} = \mathbf{U}^{(t)}\mathbf{S}^{(t)}\mathbf{V}^{(t)^\top}$ *denotes the singular value decomposition of* $\mathbf{M}^{(t)}$, *and* $\mathcal{E}_i^{(t)}$ *represents the Newton-Schulz5 approximation error after* i *iterations. Suppose the following:*

- *The gradient* $\nabla\mathcal{L}(\mathbf{W})$ *is L-Lipschitz continuous.*

- *There exists* $\delta > 0$ *such that* $\|\mathcal{E}_i^{(t)}\| \leq \delta\|\mathbf{U}^{(t)}\mathbf{V}^{(t)^\top}\| = \delta\sqrt{n}$, *for all t.*

If we take $\beta = 1 - \alpha$ *with* $\alpha = \min(\frac{\sqrt{RL}}{\sigma\sqrt{T}}, 1)$, $\eta_t = \eta = \frac{\sqrt{4R}}{\sqrt{(10/(1-\beta)+2m+4m\delta+2m\delta^2)TL}}$, *and* $B = 1$
(batch free convergence) than $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|\right]$ *is bounded by*

$$\mathcal{O}\left(\left[\frac{\sqrt{RLm(2+4\delta+2\delta^2)}}{\sqrt{T}} + \frac{\sigma^2}{\sqrt{RLT}} + \frac{\sigma(RL)^{1/4} + \sqrt{\sigma}(RL)^{1/4}}{T^{1/4}}\right]\frac{1}{1-4\sqrt{m}\delta}\right),$$

where $R = \mathcal{L}(\mathbf{W}^{(0)}) - \mathcal{L}^*$. *If we take* β *as an arbitrary constant, and* $B = T$, *we have,*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| \leq \mathcal{O}\left(\left[\frac{\sqrt{RLm(2+4\delta+2\delta^2)}}{\sqrt{T}} + \frac{\sqrt{RL}}{\sqrt{T}} + \frac{\sigma}{T^{3/2}} + \frac{\sigma}{\sqrt{T}}\right]\frac{1}{1-4\sqrt{m}\delta}\right).$$

Remark 3.4 (Comparison: slower convergence vs exact orthogonalization). When $\delta = 0$, indicating an absence of error, the convergence rate is aligned with the one derived in [41], Theorem 2.1, that is

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|\right] \leq \mathcal{O}\left(\frac{\sqrt{nRL}}{\sqrt{T}} + \frac{\sigma}{T^{3/2}} + \frac{\sigma}{\sqrt{T}}\right).$$

This result overlooks the error associated with the Newton-Schulz5 approximation because it is based on a theoretically exact method of orthogonalization.

Remark 3.5 (The impact of δ on the convergence rate). A reduction in δ is associated with an improvement in the convergence rate. Furthermore, it should be noted that δ influences the step size η; a larger δ yields a smaller η, providing an additional explanation for the convergence rate.

Remark 3.6 (The size of δ). We acknowledge that the findings of our analysis are applicable only under the conditions specified in $1 - 4\sqrt{n}\delta > 0 \Rightarrow \delta < \frac{1}{4\sqrt{n}}$. In scenarios where $\delta > \frac{1}{4\sqrt{n}}$ applies, the algorithm may fail to converge. To ensure that δ remains sufficiently small, the Newton-Schulz5 method necessitates a substantial number of iterations, consequently slowing down the convergence.

Remark 3.7 (Speed-up by SVD vs Newton-Schulz5 approximation). According to Lemma 3.2, these low-rank moments, which inherently possess exceptionally high κ, result in an error expressed by $(1 - \varepsilon)^{2^i}$ concerning a remarkably small ε. This situation necessitates numerous iterations for the Newton-Schultz method to converge. For example, if $(1 - \varepsilon) = 0.99$ is considered and Newton-Schultz5 is utilized with 5 iterations, the error would be $\approx 0.99^{32} = 0.725$, relative to the norm of the moment, namely $\mathbf{M}$.

Recall that in the low-rank setting, accurately computing the pseudoinverse using singular value decomposition (SVD) is numerically advantageous and reasonably computationally affordable compared to iterative methods such as Newton–Schulz. For a general matrix $\mathbf{A} \in \mathbb{R}^{n \times r}$, the SVD provides a decomposition $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$, with $\mathbf{U} \in \mathbb{R}^{n \times n}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times r}$, and $\mathbf{V} \in \mathbb{R}^{r \times r}$. The Moore–Penrose pseudoinverse is then calculated as $\mathbf{A}^\dagger = \mathbf{V}\boldsymbol{\Sigma}^\dagger\mathbf{U}^\top$, requiring approximately $4nr^2 + 8r^3$ floating-point operations (FLOPs) for the initial decomposition, and an additional $rn^2 + r^2n$ FLOPs for subsequent multiplications, totaling roughly $4nr^2 + 8r^3 + rn^2 + r^2n$ FLOPs.

Alternatively, approximating the inverse of $\mathbf{A}^\top\mathbf{A} \in \mathbb{R}^{r \times r}$ using Newton–Schulz iterations involves nr^2 FLOPs to form the matrix $\mathbf{A}^\top\mathbf{A}$, approximately $20r^3 + 10r^2$ FLOPs for five iterations, and an additional r^2n FLOPs to multiply by $\mathbf{A}^\top$, resulting in a total of about $nr^2 + r^2n + 20r^3 + 10r^2$ FLOPs. For example, when the rank is $r = 8$ and $n = 1024$, the SVD approach requires approximately twice as many operations as Newton–Schulz5. Nonetheless, given the superior numerical stability and inherent optimality of the SVD-based method, the increase in computational effort is negligible relative to the other full-matrix operations required in each update step.

3.2 Method

We are now ready to present our main algorithm designed to accelerate the low-rank optimization scheme outlined in Algorithm 1. A detailed mathematical formulation of the weight update rule proposed in this paper is given in Appendix C. The algorithm consists of four primary blocks, all contained within an outer loop that continues until convergence is achieved or a predefined number of epochs is reached. Each block serves a specific purpose, which will be explained in detail below.

- **Block 1 (Adaptive Subspace Selection)**: We select the subspace along the directions of the r largest eigenvectors, but since computing full SVD for large matrices is computationally intensive and memory-demanding, we leverage the Randomized-SVD by [44], which is an efficient technique for producing a "good" proxy for the optimal low-rank approximation. It solves the optimization problem $\arg\min_{\mathbf{Q}^{(t)} \in \mathbb{R}^{m \times r}} \left\| \mathbf{G} - \mathbf{Q}^{(t)} \mathbf{Q}^{(t)^\top} \mathbf{G}(t) \right\|_F$, and approximates the matrix $\mathbf{G}$ as $\hat{\mathbf{G}} \approx \mathbf{Q}^{(t)} \mathbf{Q}^{(t)^\top} \mathbf{G}(t)$, that requires $O(mnr + mr^2)$ operations, instead of $O(\min(mn^2, m^2 n))$ applied by SVD.

- **Block 1.1 (Moment Subspace Transformation)**: We transform the first-order moments evaluated during the low-rank optimization steps, which occur in Block 2, between the preceding and the newly updated subspace. This transformation is required because, as will be demonstrated later, the first moments of the gradients in Block 2 are aligned with the previously projected subspace. Consequently, a transformation is necessary to translate them from the former subspace to the current one.

- **Block 2 (Low-Rank Steepest Descent Optimization)**: Here we calculate the (steepest) optimization step. SVD operation is adopted to solve exactly $(\mathbf{M}^{(t)} \mathbf{M}^{(t)^\top})^{-1/2} \mathbf{M}^{(t)}$. Let $\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top = \hat{\mathbf{M}}^{(t)}$ be the singular value decomposition (SVD) of $\hat{\mathbf{M}}^{(t)}$, we will have $(\mathbf{M}^{(t)} \mathbf{M}^{(t)^\top})^{-1/2} \mathbf{M}^{(t)} = \mathbf{U} \mathbf{V}^\mathbf{T}$, which orthogonalizes $\hat{\mathbf{M}}^{(t)}$. Formally, the

$$\text{Orthogonalization\_SVD}(\mathbf{A}) = \arg\min_{\mathbf{O}} \{\|\mathbf{O} - \mathbf{A}\|_F : \text{either } \mathbf{O}^T \mathbf{O} = \mathbf{I} \text{ or } \mathbf{O} \mathbf{O}^T = \mathbf{I}\}.$$

- **Block 3**: Rather than using standard gradient clipping, we adopt the Norm-growth Limiter (NL) introduced in [21], which has been shown to slightly outperform traditional clipping techniques by better constraining the progression of gradient magnitudes. Specifically, the gradient update is modified as follows, if $\frac{\|\mathbf{O}^{(t)}\|}{\|\mathbf{O}^{(t-1)}\|} > \gamma$ then $\mathbf{O}^{(t)} \leftarrow \frac{\mathbf{O}^{(t)}}{\|\mathbf{O}^{(t)}\|} \cdot \gamma \|\mathbf{O}^{(t-1)}\|$, where the scalar γ serves as a growth threshold to regulate abrupt increases in gradient norm from one iteration to the next. We use $\gamma = 1.1$, which empirically yields the best results.

- **Block 4 (Update Step in the Original Space).** To better exploit already computed gradient information, we suggest to use the orthogonal term of the gradient that lies outside the low-rank subspace spanned $\mathbf{O}^{(t)}$, namely $\mathbf{G}^{(t)\perp} = \mathbf{G}^{(t)} - \mathbf{Q}^{(t)} \underbrace{\mathbf{Q}^{(t)^\top} \mathbf{G}^{(t)}}_{\hat{\mathbf{G}}^{(t)}}$. Importantly, $\mathbf{G}^{(t)\perp}$ does not interfere with the orthogonalized moment estimation $\mathbf{O}^{(t)}$, meaning it can be incorporated without compromising stability. Since $\mathbf{G}^{(t)}$ is already computed and stored in each iteration, no extra memory is required. Furthermore, because $\mathbf{Q}^{(t)}$ is of a low rank (typically rank 4, 8, or 16), the additional computationations overhead is negligible. For efficient memory usage, instead of explicitly forming the perpendicular part $\mathbf{G}^{(t)\perp}$, we work only in the rank-r subspace spanned by $\mathbf{Q}^{(t)}$. Practically, we use fact that

$$\underbrace{\left(\mathbf{G}^{(t)} - \mathbf{Q}^{(t)} \mathbf{Q}^{(t)^\top} \mathbf{G}^{(t)} \right)}_{\mathbf{G}^{(t)\perp}} + \mathbf{Q}^{(t)} \mathbf{O}^{(t)} = \mathbf{G}^{(t)} - \mathbf{Q}^{(t)} (\hat{\mathbf{G}}^{(t)} - \mathbf{O}^{(t)}),$$

which utilizes the already calculated projected gradient $\hat{\mathbf{G}}^{(t)} := \mathbf{Q}^{(t)^\top} \mathbf{G}^{(t)}$. Accordingly, the pre-trained model parameters are updated along with weight decay,

$$\mathbf{W}^{(t)} \leftarrow \mathbf{W}^{(t-1)} - \eta \left(\mathbf{G}^{(t)} - \mathbf{Q}^{(t)} (\hat{\mathbf{G}}^{(t)} - \mathbf{O}^{(t)}) \right) - \eta \lambda \mathbf{W}^{(t-1)}.$$

To ensure stable training across parameter matrices of different shapes, we interpret the root mean square (RMS) magnitude of updates as implicit *layer-wise learning rate adaptation*, following

the approach in [39]. By scaling updates by $\sqrt{\max(m,n)}$, our method compensates for shape-induced magnitude differences, achieving consistent effective learning rates across layers, similar to adaptive optimizers like AdamW.

Algorithm 1 SUMO: Subspace-Aware Moment-Orthogonalization Optimization

---

Input: A weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ with $m \geq n$. Step size η, decay rate μ, weight decay λ, rank r, subspace update frequency K, step clipping ratio γ.
Initialize: $t \leftarrow 0$
repeat
  # Block 1: Calculate low rank gradient projection.
  Sample mini-batch $B = \left\{ \xi_1, \xi_2, \ldots, \xi_{|B|} \right\}$
  Compute $\mathbf{G}^{(t)} \leftarrow \sum_{i=1}^{|B|} \frac{\partial}{\partial \mathbf{W}} \mathcal{L}\left(\Phi(x_i, \boldsymbol{\theta}), y_i \right)$
  if $t \bmod K = 0$ **then**
    $\mathbf{Q}^{(t)} \leftarrow \text{Truncated\_Randomized\_SVD}(\mathbf{G}^{(t)})$ # Alternatively Truncated_SVD($\mathbf{G}^{(t)}$)
    # Block 1.1: Moment subspaces transformation
    $\mathbf{R}^{r \times r} \leftarrow {\mathbf{Q}^{(t)}}^{\top} \mathbf{Q}^{(t-1)}$ if $t \geq 1$, else $\mathbf{0}^{r \times r}$
    $\mathbf{M}^{(t)^{r \times n}} \leftarrow \mathbf{R}\mathbf{M}^{(t-1)}$, if $t \geq 1$, else $\mathbf{0}^{r \times n}$ {1^{st}-order moment}
  end if # Alternatively criteria $\|\hat{\mathbf{G}}^{(t)}\| \leq \varsigma$
  $\hat{\mathbf{G}}^{(t)} \leftarrow {\mathbf{Q}^{(t)}}^{\top} \mathbf{G}^{(t)}$
  # Block 2: Low-rank steepest-decent step (moment ortogonalization)
  $\mathbf{M}^{(t)} \leftarrow \mu \mathbf{M}^{(t-1)} + \hat{\mathbf{G}}^{(t)}$
  $\mathbf{O}^{(t)} \leftarrow \text{Orthogonalization\_SVD}(\mathbf{M}^{(t)})$
  # Block 3 (Optional):
  if $\frac{\|\mathbf{O}^{(t)}\|}{\|\mathbf{O}^{(t-1)}\|} > \gamma$ **then** $\mathbf{O}^{(t)} \leftarrow \frac{\mathbf{O}^{(t)}}{\|\mathbf{O}^{(t)}\|} \cdot \gamma \|\mathbf{O}^{(t-1)}\|$
  # Block 4: Update weight in original space.
  $\mathbf{W}^{(t)} \leftarrow \mathbf{W}^{(t-1)} - \eta \left(\mathbf{G}^{(t)} - \mathbf{Q}^{(t)}(\hat{\mathbf{G}}^{(t)} - \mathbf{O}^{(t)}) \right) - \eta \lambda \mathbf{W}^{(t-1)}$
  $t \leftarrow t + 1$
until convergence criteria met (e.g. epoch number, gradient norm $\|\mathbf{G}^{(t)}\| \leq \xi$)
return $\mathbf{W}^{(T)}$

Note that, for clarity, we can assume, without loss of generality, that $m \geq n$. In the opposite scenario, the projection matrix would multiply the gradient from the right-hand side.

Theorem 3.8 (Convergence of SUMO). *For a loss function $\mathcal{L}$, and given architecture Φ, suppose that the compositions of $f \equiv \mathcal{L}(\Phi(\cdot))$ is β-smooth non-convex function that is bounded by some $M \in \mathbb{R}_+$. Let $\mathbf{G}_j^{(t)}$ denote the gradient matrix w.r.t. the j-th reversible layer $\mathbf{W}_j^{(t)}$, at time $t \in \mathbb{N}$, for all $j \in [L]$, and $T_\ell, \ell \in \mathbb{N}$ times are set by a convergence criterion (that is, $\|\hat{\mathbf{G}}_j^{(T_\ell)}\| \leq \varsigma_\ell$). Then, there exist $\mathsf{C} \in \mathbb{R}_+$ and N such that for all $T_N > \frac{\mathsf{C}}{\varepsilon^2}$, and $\frac{1}{T_N} \sum_{i=0}^{N-1} \sum_{t=T_i}^{T_{i+1}-1} \left\| \mathbf{G}_j^{(t)} \right\|_F^2 \leq \varepsilon$. Namely, Algorithm 1 achieves an ε-critical point,[2] i.e., $\left\| \mathbf{G}_j^{(t)} \right\|_F^2 \leq \varepsilon$, for some $t \in \mathbb{N}$, and any $j \in [L]$.*

The proof of Theorem 1 can be found in Appendix A. We emphasize that the convergence proof for Galore in [1] addresses only optimization within fixed subspaces, ignoring dynamic updates. AdaRankGrad's proof [2] first established guarantees for the complete dynamic-subspace updates, yet both prior works simplified the inner steps as standard SGD. In contrast, SUMO's convergence proof explicitly considers the exact optimization steps without simplifications.

To reduce memory consumption, Algorithm 1 applies per-layer weight updates during backpropagation, following recent works such as [45]. This contrasts with conventional optimizers, which store full gradients and update all weights afterward, potentially leading to inefficiencies. Details for post-hoc adapter extraction are discussed in Appendix B.
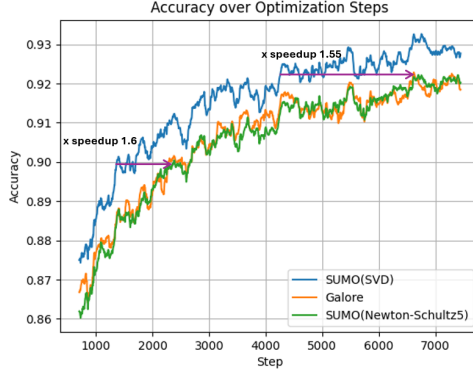
Figure 2: SUMO with SVD demonstrates superior convergence speed ($\sim 1.6\times$ faster), attaining comparable or higher accuracy than GaLore and SUMO with Newton-Schultz5 with significantly fewer optimization steps on QNLI.

4 Experiments

Fine-tuning on GLUE benchmark. Our model was evaluated using the GLUE benchmark [43] through the fine-tuning of the pre-trained Roberta-base model [42] across eight tasks. The comparative analysis includes full fine-tuning, LoRA, and GaLore methodologies, with the results enumerated in Table 2. The reported metrics are the overall accuracy (matched and mismatched) for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, F1-score for MRPC, and accuracy for the remaining tasks. Evidently, our approach improves fine-tuning accuracy while requiring less training memory, using only a single moment compared to GaLore. The experiments were carried out using the NVIDIA A100 GPU.

Table 2: Comparison of SUMO against state-of-the-art memory-efficient fine-tuning methods on the GLUE benchmark using the pre-trained RoBERTa-Base model. For comparison, we provide detailed results for SUMO using both SVD and Newton-Schulz5 orthogonalizations (ablation study).

Model	Memory	CoLA	STS-B	MRPC	RTE	SST2	MNLI	QNLI	QQP
Full Fine-Tuning	747M	62.24	90.92	91.30	79.42	94.57	87.18	92.33	92.28
LoRA (rank=4)	257M	61.38	90.57	91.07	78.70	92.89	86.82	92.18	91.29
GaLore (rank=4)	253M	60.35	90.73	92.25	79.42	94.0	87.0	92.24	91.06
SUMO (Newton-Schulz5, rank=4)	**197M**	61.8	90.82	92.43	79.36	94.17	86.92	92.26	91.27
SUMO (SVD, rank=4)	**197M**	**62.3**	**91.04**	**93.5**	**81.07**	**94.93**	**87.34**	**93.26**	**91.68**
LoRA (rank=8)	264M	**61.83**	90.80	91.90	79.06	93.46	86.94	92.25	91.22
GaLore (rank=8)	257M	60.06	90.82	92.0	79.78	94.38	87.17	92.2	91.11
SUMO (Newton-Schulz5, rank=4)	**198M**	61.74	90.79	91.94	79.69	94.17	87.21	92.24	91.38
SUMO (rank=8)	**198M**	61.7	**91.1**	**93.7**	**81.37**	**94.82**	**87.58**	**93.67**	**91.72**

Pre-training Llama on C4 Dataset. To highlight the effectiveness of our method in pre-training, we pre-train Llama models following the evaluation protocol of [1] and compare performance with the state-of-the-art method, in terms of perplexity and memory usage. Specifically, we compare SUMO's performance with state-of-the-art methods in terms of perplexity and memory efficiency. For this evaluation, we trained large Llama-based models on the C4 dataset, a curated and extensive version of the Common Crawl web corpus [46]. This dataset is widely used for pre-training language models and developing word representations. To better reflect real-world pre-training scenarios, we trained on a non-repeating, large-scale dataset and scaled model sizes up to 1 billion parameters. The results of these experiments are shown in Table 3. Experiments were conducted using an NVIDIA H200 GPU.

Few/Zero-shot reasoning and long-context generalization. To evaluate the performance of our method on a complex reasoning task, we utilize the GSM8K dataset [47] to test systematic generalization. For these experiments, we used a batch size of 32 and 10 epochs for fine-tuning. We present the performance result in Table 4 training Phi-2 (2.7B) model [48], and in Table 5 training

---

[2]Also known as ε-stationary, see, e.g., [14].

Table 3: Comparison of state-of-the-art low-rank algorithms for pre-training Llama models of varying sizes on the C4 dataset. The results are reported in terms of validation perplexity. As shown, SUMO leads to improved performance with substantial memory reduction compared to leading parameter-efficient fine-tuning schemes.

Method	60M	130M	350M	1B
Full-Rank	34.06 (0.36G)	25.08 (0.76G)	18.80 (2.06G)	15.56(7.80G)
GaLore	34.88 (0.24G)	25.36 (0.52G)	18.95 (1.22G)	15.64(4.38G)
Low-Rank	78.18 (0.26G)	45.51 (0.54G)	37.41 (1.08G)	142.53(3.57G)
LoRA	34.99 (0.36G)	33.92 (0.80G)	25.58 (1.76G)	19.21(6.17G)
ReLoRA	37.04 (0.36G)	29.37 (0.80G)	29.08 (1.76G)	18.33(6.17G)
SUMO	**34.26 (0.23G)**	**24.87 (0.51G)**	**18.69** (1.16G)	**14.68** (3.84G)
Training Tokens	1.1B	2.2B	6.4B	13.1B
r/d_{model}	128/256	256/768	256/1024	512/2048

Llama (3B) model [49]. The results demonstrate that the proposed method significantly improves generalization to out-of-distribution data. We used an NVIDIA H200 GPU.

Table 4: Zero-shot evaluation on GSM8K dataset (Phi-2, 2.7B).

Phi-2 (2.7B)	Rank	Accuracy (0-shot)
Base Model	64	15.16%
Galore	64	52.24%
LoRA	64	42.8%
SUMO	64	**54.13**%

Table 5: 8-shot evaluation on GSM8K dataset (LLaMA, 3B).

LLaMA (3B)	Rank	Accuracy (8-shot)
Base Model	64	17.93%
Galore	64	74.9%
LoRA	64	68.3%
SUMO	64	**76.7**%

Additional experiments and ablation studies are presented in the Appendix D.

Zero-shot generalization. Following this, we extend our evaluation to diverse commonsense and reasoning tasks using zero-shot methods, and then conduct the exact evaluation protocol described in Table 4 of [3] (which suggests the Apollo optimizer). The task details are presented in Appendix D.1. To ensure a fair comparison, we repeat the exact experimental setup used for AdamW, APOLLO, and APOLLO-Mini. The results are summarized in Table 6 below.

Table 6: Zero-shot evaluation of Llama-350M models pretrained with sequence length 1024 across reasoning tasks (lower is better for perplexity; higher is better otherwise).

Method	Memory	Perplexity	BoolQ	RTE	HS	WG	OBQA	ARC-E	ARC-C	PIQA	SciQ	MathQA	Avg.
AdamW	1.37G	16.30	0.4917	0.4693	0.3688	0.5233	0.332	0.3729	0.2449	0.6534	0.609	0.2064	0.4272
APOLLO	0.34G	15.64	0.5373	0.4698	0.3850	0.4925	0.322	0.3788	0.2483	0.6681	**0.624**	0.2127	0.4406
APOLLO-Mini	0.15G	16.12	0.5376	0.4562	0.3707	0.5217	**0.324**	0.3758	0.2312	0.6638	0.619	0.2224	0.4374
SUMO	0.18G	**15.49**	**0.5479**	**0.4709**	**0.3937**	**0.5313**	0.321	**0.3832**	**0.2496**	**0.6709**	0.623	**0.2246**	**0.4416**

As shown, the SUMO-pretrained Llama-350M model achieves lower perplexity on average and consistently outperforms on downstream benchmarks.

5 Discussion

Our results highlight that exact moment orthogonalization within a low-dimensional adaptive subspace significantly improves both convergence and stability in memory-efficient LLM training. By avoiding the approximation errors of Newton–Schulz5, the proposed SUMO leverages the low-rank structure of gradients to enable accurate, spectral-norm-aligned updates with minimal overhead.

Empirically, SUMO outperforms prior low-rank methods on both fine-tuning and pre-training tasks, achieving greater memory reduction than memory-efficient benchmarks such as Galore. Our theoretical analysis further confirms its superior convergence properties under practical conditions. These findings position SUMO as a simple yet effective alternative to approximate geometric optimizers. Future work may investigate parallel computations for orthogonalization, integrate quantization techniques, and assess the effectiveness of the method in knowledge editing [50] or domain generalization [51, 52].

Acknowledgment

References

[1] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection, 2024.

[2] Yehonathan Refael, Jonathan Svirsky, Boris Shustin, Wasim Huleihel, and Ofir Lindenbaum. Adarankgrad: Adaptive gradient rank and moments for memory-efficient LLMs training and fine-tuning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[3] Hanqing Zhu, Zhenyu Zhang, Wenyan Cong, Xi Liu, Sem Park, Vikas Chandra, Bo Long, David Z Pan, Zhangyang Wang, and Jinwon Lee. Apollo: Sgd-like memory, adamw-level performance. *arXiv preprint arXiv:2412.05270*, 2024.

[4] Barak Battash, Lior Wolf, and Ofir Lindenbaum. Revisiting the noise model of stochastic gradient descent. In *International Conference on Artificial Intelligence and Statistics*, pages 4780–4788. PMLR, 2024.

[5] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018.

[6] Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam, 2025.

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

[8] Louis Cesista Franz. The Case for Muon, October 2024.

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

[10] Dmitry Kovalev. Understanding gradient orthogonalization for deep learning via non-euclidean trust-region optimization, 2025.

[11] Hao Sun, Li Shen, Qihuang Zhong, Liang Ding, Shixiang Chen, Jingwei Sun, Jing Li, Guangzhong Sun, and Dacheng Tao. Adasam: Boosting sharpness-aware minimization with adaptive learning rate and momentum for training deep neural networks, 2023.

[12] Nicholas J Higham. Newton's method for the matrix square root. *Mathematics of computation*, 46(174):537–549, 1986.

[13] Jingzhao Zhao, Frederik T. Schaefer, and Anima Anandkumar. Zero initialization: Initializing neural networks with only zeros and ones. *Transactions on Machine Learning Research*, 2022.

[14] Romain Cosson, Ali Jadbabaie, Anuran Makur, Armin Reisizadeh, and Devavrat Shah. Low-Rank Gradient Descent. *IEEE Open Journal of Control Systems*, 2023.

[15] Greg Yang, Jacob B. Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023. arXiv:2310.17813.

[16] M. Gooneratne, K. C. Sim, P. Zadrazil, A. Kabel, F. Beaufays, and G. Motta. Low-rank gradient approximation for memory-efficient on-device training of deep neural network. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020.

[17] Shuang Huang, Brian D. Hoskins, Michael W. Daniels, Matthew D. Stiles, and George C. Adam. Low-Rank Gradient Descent for Memory-Efficient Training of Deep In-Memory Arrays. *ACM Journal on Emerging Technologies in Computing Systems*, 2023.

[18] Ionut-Vlad Modoranu, Alexander Kalinov, Ermin Kurtic, Erwin Frantar, and Dan Alistarh. Error Feedback Can Accurately Compress Preconditioners. *ArXiv preprint arXiv:2306.06098*, 2023. arXiv:2306.06098.

[19] Yuandong Tian, Lantao Yu, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning with dual deep networks, 2021.

[20] 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.

[21] Xi Chen, Kaituo Feng, Changsheng Li, Xunhao Lai, Xiangyu Yue, Ye Yuan, and Guoren Wang. Fira: Can we achieve full-rank training of llms under low-rank constraint? *arXiv preprint arXiv:2410.01623*, 2024.

[22] Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. *ArXiv*, abs/2402.03293, 2024.

[23] Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*, 2024.

[24] Thomas Robert, Mher Safaryan, Ionut-Vlad Modoranu, and Dan Alistarh. Ldadam: Adaptive optimization from low-dimensional gradient statistics, 2025.

[25] Sebastian Loeschcke, Mads Toftrup, Michael J. Kastoryano, Serge Belongie, and Vésteinn Snæbjarnarson. Loqt: Low-rank adapters for quantized pretraining, 2024.

[26] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pages 560–569. PMLR, 2018.

[27] Michael Crawshaw, Mingrui Liu, Francesco Orabona, Wei Zhang, and Zhenxun Zhuang. Robustness to unbounded smoothness of generalized signsgd. *Advances in neural information processing systems*, 35:9955–9968, 2022.

[28] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *NeurIPS*, 2023.

[29] Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be. *arXiv preprint arXiv:2304.13960*, 2023.

[30] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

[31] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

[32] Zhirong Yang and Jorma Laaksonen. Principal whitened gradient for information geometry. *Neural Networks*, 21(2-3):232–240, 2008.

[33] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

[34] Dongseong Hwang. Fadam: Adam is a natural gradient optimizer using diagonal empirical fisher information. *arXiv preprint arXiv:2405.12807*, 2024.

[35] Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology, 2024.

[36] Jeremy Bernstein and Laker Newhouse. Modular duality in deep learning, 2024.

[37] David E Carlson, Edo Collins, Ya-Ping Hsieh, Lawrence Carin, and Volkan Cevher. Preconditioned spectral descent for deep learning. *Advances in neural information processing systems*, 28, 2015.

[38] Mark Tuddenham, Adam Prügel-Bennett, and Jonathan Hare. Orthogonalising gradients to speed up neural network optimisation. *arXiv preprint arXiv:2202.07052*, 2022.

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

[40] Yehonathan Refael, Adam Hakim, Lev Greenberg, Tal Aviv, Satya Lokam, Ben Fishman, and Shachar Seidman. Slip: Securing llms ip using weights decomposition, 2024.

[41] Jiaxiang Li and Mingyi Hong. A note on the convergence of muon and further. *arXiv preprint arXiv:2502.02900*, 2025.

[42] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[43] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[44] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions, 2010.

[45] Kai Lv, Hang Yan, Qipeng Guo, Haijun Lv, and Xipeng Qiu. Adalomo: Low-memory optimization with adaptive learning rate, 2024.

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

[47] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *Arxiv*, 2021.

[48] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. Phi-2: The surprising power of small language models. *Microsoft Research Blog*, 1(3):3, 2023.

[49] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[50] Amit Rozner, Barak Battash, Lior Wolf, and Ofir Lindenbaum. Knowledge editing in language models via adapted direct preference optimization. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 4761–4774, 2024.

[51] Yehonathan Refael, Iftach Arbel, Ofir Lindenbaum, and Tom Tirer. Lorenza: Enhancing generalization in low-rank gradient llm training via efficient zeroth-order adaptive sam, 2025.

[52] Amit Rozner, Barak Battash, Lior Wolf, and Ofir Lindenbaum. Domain-generalizable multiple-domain clustering. *Transactions on Machine Learning Research*, 2023.

[53] Ashok Cutkosky and Harsh Mehta. Momentum improves normalized sgd, 2020.

[54] Kenji Kawaguchi. Deep learning without poor local minima. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[55] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. MAWPS: A math word problem repository. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, 2015.

[56] C. Clark, K. Lee, M.W. Chang, T. Kwiatkowski, M. Collins, and K. Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[57] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2018.

[58] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[59] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[60] Todor Mihaylov, Peter Clark, Oyvind Tafjord, and Tushar Khot. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, 2018.

[61] Peter Clark and Oren Etzioni. Think you have solved question answering? try arc, the ai2 reasoning challenge. In *arXiv preprint arXiv:1803.05457*, 2018.

[62] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439, 2020.

[63] Matt Gardner Johannes Welbl, Nelson F. Liu. Crowdsourcing multiple choice science questions, 2017.

[64] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms, 2019.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: we propose SUMO (Subspace-Aware Moment-Orthogonalization), an optimizer that employs exact singular value decomposition (SVD) for moment orthogonalization within a dynamically adapted low-dimensional subspace, enabling norm-inducing steepest descent optimization steps. By explicitly aligning optimization steps with the spectral characteristics of the loss landscape, SUMO effectively mitigates approximation errors associated with commonly used methods like Newton-Schulz orthogonalization approximation (claim. 3.3). We theoretically establish an upper bound on these approximation errors (claim. 3.2), proving their dependence on the condition numbers of moments, conditions we analytically demonstrate are encountered during LLM training. Furthermore, we both theoretically and empirically illustrate that exact orthogonalization via SVD substantially improves convergence rates while reducing overall complexity. Empirical evaluations confirm that SUMO accelerates convergence, enhances stability, improves performance, and reduces memory requirements by up to 20% compared to state-of-the-art methods. (Figure. 2, Table. 1)

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Our methods work when the layers are reversible, and improves other methods especially when the moments are ill-conditioned.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.

- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: all assumptions of theoretical results are presented in the paper, and all proof are in the Appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: Everything needed for reproducibility is presented in Section 4.

Guidelines:

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

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

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

Answer: [No]

Justification: We provide experimental details in the main text and appendix. A GitHub link well be published soon.

Guidelines:

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

6. **Experimental setting/details**

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

Answer:[Yes]

Justification: All training and test details are presented in Section 4.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [Yes]

Justification: All detailed in Section 4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

Answer: [Yes]

Justification: All detailed in Section 4.

Guidelines:

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

9. **Code of ethics**

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

Answer: [Yes]

Justification: The research conforms with the NeurIPS Code of Ethics.

Guidelines:

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

10. **Broader impacts**

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

Answer:[NA]

Justification: The paper talks about optimizers. It does not have societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer:[NA]

Justification: The paper poses no such risks.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We followed and gave proper credits for our use of data and models in Section 4.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

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

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

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

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human objects.

Guidelines:

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

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

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

Answer: [NA]

Justification: This paper does not involve crowdsourcing nor research with human objects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

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

A Proofs of Section 3

In this section, we prove all the theorems and results of Section 3.

Proof of Lemma 3.1. We aim to show that if the gradient $\mathbf{G}^{(t)}$ becomes approximately rank-one exponentially fast, then the exponentially weighted moving average of the gradients (i.e., the momentum $\mathbf{M}^{(t)}$) also exhibits exponential decay of higher-rank components.

Consider the singular value decomposition of the gradient $\mathbf{G}^{(t)} = \mathbf{U}^{(t)}\Sigma^{(t)}\mathbf{V}^{(t)\top}$, at iteration t. For all natural numbers $r < m$, we define $\mathbf{H}^{(t)^{m \times r}}(r) = \mathbf{U}[:, 1:r]$. To enhance notation clarity, denote $\mathbf{P}^{(t)}(r) = \mathbf{H}^{(t)}(r)\mathbf{H}^{(t)\top}(r)$, where $\mathbf{P}^{(t)}(r)$ represents an orthogonal projection matrix, satisfying the conditions $\mathbf{P}^{(t)\top}(r)\mathbf{P}^{(t)}(r) = \mathbf{P}^{(t)}(r)$, and $\mathbf{P}^{(t)}(r) = \mathbf{P}^{(t)\top}(r)$. Without compromising generality, it is assumed that at $t = 0$, the rank of $\mathbf{G}^{(0)}$ is characterized by $\mathrm{rank}\left(\mathbf{G}^{(0)}\right) > r$. For reversible networks, it has been established in [1][Theorem 3.2] that the gradients assume the form $\mathbf{G}^{(t)} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{A}_i - \mathbf{B}_i\mathbf{W}^{(t)}\mathbf{C}_i\right)$, characterized by constant matrices $\{\mathbf{A}_i\}_i$ and positive semi-definite (PSD) matrices $\{\mathbf{B}_i, \mathbf{C}_i\}_i$, for $t \geq \mathsf{t}_0$, where $\mathsf{t}_0 \in \mathbb{N}$ holds. It is pertinent to recall that the vanilla weight update can be represented as $\mathbf{W}^{(t)} = \mathbf{W}^{(t-1)} + \eta\mathbf{G}^{(t-1)}$. Let $\mathbf{S} \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathbf{C}_i \otimes \mathbf{B}_i$ and $\lambda_1 < \lambda_2$ denote its two smallest distinct eigenvalues. To substantiate our findings, we utilize several results and arguments presented in the proof of Lemma 3.3 in [1]. Specifically, consider $\mathbf{G}^{(\mathsf{t}_0)}$ as the projection of $\mathbf{G}^{(\mathsf{t}_0)}$ onto the minimal eigenspace $\mathcal{V}_1$ of S corresponding to λ_1. According to our assumption, the rank of $\mathbf{G}^{(\mathsf{t}_0)}$ is L, and its singular value decomposition (SVD) is given by $\mathbf{G}^{(\mathsf{t}_0)} = \sum_{l=1}^{L}c_l z_l y_l^\top$, where $\{z_l\}_{l=1}^{L}$ and $\{y_l\}_{l=1}^{L}$ are orthonormal unit vectors, and $\{c_l\}_{l=1}^{L}$ are the corresponding singular values. Therefore, as per Lemma 3.3 in [1], the gradient can be decomposed into,

$$\left\|\mathbf{G}^{(t)}\right\|_F^2 \leq (1 - \eta\lambda_2)^{2t}\left\|g_0^\perp\right\|_2^2 + (1 - \eta\lambda_1)^{2t}\left\|g_0^\|\right\|_2^2,$$

where $g_0^\|$ is the projection of $\mathbf{G}^{(0)}$ onto the minimal eigenspace $\mathcal{V}_1$ of $S = \frac{1}{N}\sum_{i=1}^{N}\mathbf{C}_i \otimes \mathbf{B}_i$, and $g_0^\perp$ is orthogonal to $\mathcal{V}_1$. Here, $\lambda_1 < \lambda_2$ are the smallest distinct eigenvalues of S.

We now unroll the momentum update, $\mathbf{M}^{(t)} = \sum_{s=1}^{t}\beta^{t-s}\mathbf{G}^{(s)}$. Substitute the decomposition of $\mathbf{G}^{(s)}$,

$$\left\|\mathbf{M}^{(t)}\right\|_F^2 \leq \sum_{s=1}^{t}\beta^{t-s}\left[(1 - \eta\lambda_1)^s g_0^\| + (1 - \eta S)^s g_0^\perp\right] = \sum_{s=1}^{t}\beta^{t-s}(1-\eta\lambda_1)^s g_0^\| + \sum_{s=1}^{t}\beta^{t-s}(1-\eta S)^s g_0^\perp.$$

Let us define $a_t \triangleq \sum_{s=1}^{t}\beta^{t-s}(1 - \eta\lambda_1)^s, \quad b_t \triangleq \sum_{s=1}^{t}\beta^{t-s}(1 - \eta S)^s g_0^\perp$, so that $\left\|\mathbf{M}^{(t)}\right\|_F^2 = a_t g_0^\| + b_t$. Now, compute the squared Frobenius norm:

$$\|\mathbf{M}^{(t)}\|_F^2 = \|a_t g_0^\| + b_t\|_F^2 = a_t^2\|g_0^\|\|_F^2 + 2a_t\langle g_0^\|, b_t\rangle + \|b_t\|_F^2.$$

Since $g_0^\| \perp g_0^\perp$ and b_t lies in the span of $g_0^\perp$, we have $\langle g_0^\|, b_t\rangle = 0$, thus,

$$\|\mathbf{M}^{(t)}\|_F^2 = a_t^2\|g_0^\|\|_F^2 + \|b_t\|_F^2.$$

Likewise, the spectral norm $\|\mathbf{M}^{(t)}\|_2^2 \geq a_t^2\|g_0^\|\|_2^2$. Hence, the ratio

$$\kappa_m(t) = \frac{\|\mathbf{M}^{(t)} - \mathbf{P}^{(t)}(1)\mathbf{M}^{(t)}\|_F^2}{\|\mathbf{M}^{(t)}\|_F^2} \leq \frac{\|\mathbf{M}^{(t)}\|_F^2 - \|\mathbf{M}^{(t)}\|_2^2}{\|\mathbf{M}^{(t)}\|_F^2} \leq \frac{a_t^2\|g_0^\|\|_F^2 + \|b_t\|_F^2 - a_t^2\|g_0^\|\|_F^2}{a_t^2\|g_0^\|\|_F^2 + \|b_t\|_F^2}.$$

Using that $\|g_0^\|\|_2^2 = \sigma_1^2$, and the decay bound $\|b_t\|_F^2 = O((\max\{\beta, 1 - \eta\lambda_2\})^{2t})$, while $a_t^2 = \Omega((\max\{\beta, 1 - \eta\lambda_1\})^{2t})$, we conclude:

$$\kappa_m(t) \leq O\left(\left(\frac{\max\{\beta, 1-\eta\lambda_2\}}{\max\{\beta, 1-\eta\lambda_1\}}\right)^{2t}\right) = O(C^{-t}),$$

for some constant $C > 1$. $\qquad\square$

Before proving Lemma 3.3, we shortly present the following two preliminary lemmas. To that end, we present the following notations,

- $\mathbf{M}^{(t)}$ - The moment in iteration t. Its dimensions are $n \times m$, where $n < m$.
- $\|\cdot\|$ - The Frobenius norm: $\|\mathbf{A}\| = \|\mathbf{A}\|_F = \sqrt{\mathbf{A}\mathbf{A}^\top}$
- $\mathcal{L}^*$ - A stationary point to which the loss $\mathcal{L}$ converges.
- B - Batch size.
- For $\mathbf{A} \in \mathbb{R}^{m \times m}$ and $\mathbf{B} \in \mathbb{R}^{n \times n}$ we denote $\mathbf{A}\mathbf{I}_{m \times n}\mathbf{B}^\top$ by $\mathbf{A}\mathbf{B}^\top$ for convenience.

Additionally, we note that our proof is based on an equivalent but slightly modified formulation of moment's update. Specifically, instead of using the standard formulation of the moment's update

$$\mathbf{M}^{(t+1)} = \beta\mathbf{M}^{(t)} + \mathbf{G}^{(t)},$$

we consider the convex combination,

$$\mathbf{M}^{(t+1)} = \beta\mathbf{M}^{(t)} + (1-\beta)\mathbf{G}^{(t)}.$$

This alternative formulation simplifies the analysis, but equivalent. To show that, we point out that we can choose an modefied learning step $\eta^* = \frac{\eta}{1-\beta} > 0$ we get the same weight's updating step.

$$\eta^* Orth\left(\beta\mathbf{M}^{(t)} + (1-\beta)\mathbf{G}^{(t)}\right)$$
$$= Orth\left(\eta^*\beta\mathbf{M}^{(t)} + \eta^*(1-\beta)\mathbf{G}^{(t)}\right)$$
$$= Orth\left(\eta\frac{\beta}{1-\beta}\mathbf{M}^{(t)} + \eta\mathbf{G}^{(t)}\right)$$
$$= \eta Orth\left(\frac{\beta}{1-\beta}\mathbf{M}^{(t)} + \mathbf{G}^{(t)}\right),$$

where $Orth$ is the SVD orthogonalization step, formaly solving

$$\arg\min_{\mathbf{O}}\{\|\mathbf{O} - \mathbf{A}\|_F : \text{either } \mathbf{O}^T\mathbf{O} = \mathbf{I} \text{ or } \mathbf{O}\mathbf{O}^T = \mathbf{I}\}.$$

Obviously, $\beta > 0$ could be chosen in a way that $\frac{\beta}{1-\beta}\mathbf{M}^{(t)}$ would result in any required positive real number.

We assume the following 4 assumptions throughout our proofs:

(A1) The gradient $\nabla\mathcal{L}(\mathbf{W})$ is L-Lipschitz continuous.

(A2) $\nabla\mathcal{L}(\mathbf{W}, \xi)$ is an unbiased estimator of $\nabla\mathcal{L}(\mathbf{W})$ where $\mathcal{L}(\mathbf{W}, \xi)$ is the gradient of $\mathcal{L}(\mathbf{W})$ when taking a single training sample ξ.

(A3) $\mathbb{E}\|\nabla\mathcal{L}(\mathbf{W}, \xi) - \nabla\mathcal{L}(\mathbf{W})\| \leq \sigma^2$.

(A4) There exists $\delta > 0$ such that $\|\mathcal{E}_5^{(t)}\| \leq \delta\|\mathbf{U}^{(t)}\mathbf{V}^{(t)\top}\| = \delta\sqrt{m}$ for all t.

Lemma A.1 (Descent Lemma with Newton-Schulz Approximation Error). *Consider the Muon optimizer update defined by*

$$\mathbf{M}^{(t)} \leftarrow \beta\mathbf{M}^{(t-1)} + (1-\beta)\mathbf{G}^{(t)},$$
$$\mathbf{O}^{(t)} \leftarrow \mathbf{U}^{(t)}\mathbf{V}^{(t)\top} + \mathcal{E}_5^{(t)}, \quad \textit{(Newton-Schulz 5 iteration approximation),}$$
$$\mathbf{W}^{(t+1)} \leftarrow \mathbf{W}^{(t)} - \eta_t\mathbf{O}^{(t)},$$

where $\mathbf{M}^{(t)} = \mathbf{U}^{(t)}\mathbf{S}^{(t)}\mathbf{V}^{(t)\top}$ *is the singular value decomposition of* $\mathbf{M}^{(t)}$, *and* $\mathcal{E}_5^{(t)}$ *represents the Newton-Schulz (5 iterations) approximation error. Additionally, assume (A1) - (A4). Then the following holds:*

$$\mathcal{L}(\mathbf{W}^{(t+1)}) \leq$$

$$\mathcal{L}(\mathbf{W}^{(t)}) - \left(\frac{\eta_t}{4} - \eta_t\sqrt{m}\delta\right)\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| + \eta_t\frac{5}{2}\|\nabla\mathcal{L}(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\| + \frac{\eta_t^2 mL}{2} + \eta_t^2 mL\delta + \frac{\eta_t^2 Lm\delta^2}{2}$$

Proof. Since $\mathcal{L}$ is L-lipschitz function, the descent lemma holds. Thus we have

$$\mathcal{L}(\mathbf{W}^{(t+1)}) \leq \mathcal{L}(\mathbf{W}^{(t)}) + \langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \mathbf{W}^{(t+1)} - \mathbf{W}^{(t)}\rangle + \frac{L}{2}\|\mathbf{W}^{(t+1)} - \mathbf{W}^{(t)}\|^2$$

$$= \mathcal{L}(\mathbf{W}^{(t)}) - \eta_t\langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \mathbf{U}^{(t)}\mathbf{V}^{(t)\top} + \mathcal{E}_5^{(t)}\rangle + \frac{L\eta_t^2}{2}\|\mathbf{U}^{(t)}\mathbf{V}^{(t)\top} + \mathcal{E}_5^{(t)}\|^2$$

$$= \mathcal{L}(\mathbf{W}^{(t)}) - \eta_t\langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \mathbf{U}^{(t)}\mathbf{V}^{(t)\top}\rangle - \eta_t\langle\nabla\mathcal{L}(\mathbf{W}^{(t)}), \mathcal{E}_5^{(t)}\rangle$$

$$+ \frac{L\eta_t^2}{2}\left(n + 2\langle\mathbf{U}^{(t)}\mathbf{V}^{(t)\top}, \mathcal{E}_5^{(t)}\rangle + \|\mathcal{E}_5^{(t)}\|^2\right)$$

$$\underset{(*)}{\leq} \mathcal{L}(\mathbf{W}^{(t)}) - \frac{\eta_t}{4}\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| + \eta_t\frac{5}{2}\|\nabla\mathcal{L}(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\| + \frac{\eta_t^2 mL}{2}$$

$$+ \eta_t\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|\|\mathcal{E}_5^{(t)}\| + L\eta_t^2\sqrt{m}\|\mathcal{E}_5^{(t)}\| + \frac{L\eta_t^2}{2}\|\mathcal{E}_5^{(t)}\|^2$$

$$\leq \mathcal{L}(\mathbf{W}^{(t)}) - \frac{\eta_t}{4}\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| + \eta_t\frac{5}{2}\|\nabla\mathcal{L}(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\| + \frac{\eta_t^2 mL}{2}$$

$$+ \eta_t\delta\sqrt{m}\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| + L\eta_t^2\sqrt{m}\delta\sqrt{m} + \frac{L\eta_t^2}{2}\delta^2 n$$

$$= \mathcal{L}(\mathbf{W}^{(t)}) - \left(\frac{\eta_t}{4} - \eta_t\sqrt{m}\delta\right)\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| + \eta_t\frac{5}{2}\|\nabla\mathcal{L}(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\| + \frac{\eta_t^2 mL}{2} +$$

$$+ \eta_t^2 mL\delta + \frac{\eta_t^2 Lm\delta^2}{2}$$

Where in $(*)$ we used [53], equation 2.8. $\qquad\square$

Lemma A.2. *For constant* $\eta_t = \eta > 0$*, the following holds*

$$\frac{\eta - 4\eta\sqrt{m}\delta}{4}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| \leq$$

$$\mathcal{L}(\mathbf{W}^{(1)}) - \mathcal{L}^* + \eta_t\frac{5}{2}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\| + \frac{\eta^2 mLT}{2} + T\eta^2 mL\delta + \frac{T\eta^2 Lm\delta^2}{2}.$$

Proof. Using Lemma A.1, isolating $\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|$ and summing over all steps

$$\frac{\eta - 4\eta\sqrt{m}\delta}{4}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| \leq$$

$$\mathcal{L}(\mathbf{W}^{(1)}) - \mathcal{L}^* + \eta_t\frac{5}{2}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\| + \frac{\eta^2 mLT}{2} + T\eta^2 mL\delta + \frac{T\eta^2 Lm\delta^2}{2}$$

$$\square$$

Proof of Lemma 3.3. The proof follows [53]. Using the same notations as [53], we denote $\hat{\gamma}^{(t)} = \mathbf{M}^{(t)} - \nabla\mathcal{L}(\mathbf{W}^{(t)})$, $\gamma^{(t)} = \mathbf{G}^{(t)} - \nabla\mathcal{L}(\mathbf{W}^{(t)})$ and $S(\mathbf{X}, \mathbf{Y}) = \nabla\mathcal{L}(\mathbf{X}) - \nabla\mathcal{L}(\mathbf{Y})$. Note that we have the following

- $\mathbb{E}[\gamma^{(t)}] = 0$ from A(2).

- $\mathbb{E}[\|\gamma^{(t)}\|^2] \leq \frac{\sigma^2}{m}$ from A(3).

24

- $\mathbb{E}[\langle \gamma^{(i)}, \gamma^{(j)} \rangle] = 0$, $\forall i \neq j$ since $\gamma^{(i)}$ and $\gamma^{(j)}$ are independent.
- $\|S(\mathbf{X}, \mathbf{Y})\| \leq L\|\mathbf{X} - \mathbf{Y}\|$ from A(1).

Now following the update in (2), we get

$$\hat{\gamma}^{(t+1)} = \beta\hat{\gamma}^{(t)} + (1-\beta)\gamma^{(t)} + S(\mathbf{X}^{(t)}, \mathbf{X}^{(t+1)})$$

$$= \beta^t\hat{\gamma}^{(1)} + (1-\beta)\sum_{\tau=0}^{t-1}\beta^\tau\gamma^{(t-\tau)} + \sum_{\tau=0}^{t-1}\beta^\tau S(\mathbf{X}^{(t-\tau)}, \mathbf{X}^{(t+1-\tau)}),$$

therefore

$$\|\hat{\gamma}^{(t+1)}\| \leq \beta^t\|\hat{\gamma}^{(1)}\| + (1-\beta)\left\|\sum_{\tau=0}^{t-1}\beta^\tau\gamma^{(t-\tau)}\right\| + \eta L\sum_{\tau=0}^{t-1}\beta^\tau.$$

Taking expectation we get (using the fact that $\hat{\delta}_1 = \delta_1$):

$$\mathbb{E}\|\hat{\gamma}^{(t+1)}\| \leq \beta^t\frac{\sigma}{m} + (1-\beta)\sqrt{\sum_{\tau=0}^{t-1}\beta^{2\tau}\frac{\sigma^2}{B}} + \eta L\sum_{\tau=0}^{t-1}\beta^\tau$$

$$\leq \frac{\sigma}{m}\beta^t + \frac{\sigma}{m}\frac{1-\beta}{\sqrt{1-\beta^2}} + \eta L\frac{1}{1-\beta}$$

$$\leq \frac{\sigma}{m}\beta^t + \frac{\sigma}{m}\sqrt{1-\beta} + \eta L\frac{1}{1-\beta}.$$

All in all, we get

$$\sum_{t=1}^{T}\mathbb{E}[\|\hat{\gamma}^{(t+1)}\|] \leq \frac{\sigma}{(1-\beta)B} + T\sqrt{1-\beta}\frac{\sigma}{m} + \frac{T\eta L}{1-\beta}.$$

Using Lemma A.2, we get

$$\frac{\eta_t - 4L\eta_t\sqrt{m}\delta}{4}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| \leq$$

$$\mathcal{L}(\mathbf{W}^{(1)}) - \mathcal{L}^* + \eta_t\frac{5}{2}\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\| + \frac{\eta^2 mLT}{2} + T\eta^2 mL\delta + \frac{T\eta^2 Lm\delta^2}{2}.$$

Dividing both sides by $\frac{\eta - 4\eta L\sqrt{m}\delta}{4}$ we get

$$\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\| \leq$$

$$\left[\frac{4(\mathcal{L}(\mathbf{W}^{(1)}) - \mathcal{L}^*)}{\eta} + 10\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\| + 2\eta mLT + 4T\eta mL\delta + 2T\eta Lm\delta^2\right] \cdot \frac{1}{1-4\sqrt{m}\delta}$$

$$\leq \left[\frac{4R}{\eta} + 10\frac{\sigma}{(1-\beta)m} + 10T\sqrt{1-\beta}\frac{\sigma}{m} + 10\frac{T\eta L}{1-\beta} + 2\eta mLT + 4T\eta mL\delta + 2T\eta Lm\delta^2\right] \cdot \frac{1}{1-4\sqrt{m}\delta}$$

By taking $\eta = \sqrt{\frac{4R}{(10/(1-\beta)+2m+4m\delta+2m\delta^2)TL}}$ we get

$$\sum_{t=1}^{T}\|\nabla\mathcal{L}(\mathbf{W}^{(t)})\|$$

$$\left[\leq 4\sqrt{RTL(10/(1-\beta)+2m+4m\delta+2m\delta^2)} + \frac{10\sigma}{(1-\beta)m} + 10T\sqrt{1-\beta}\frac{\sigma}{(1-\beta)m}\right]\frac{1}{1-4\sqrt{m}\delta}.$$

Now we have two types of parameter choice. If we take $B = 1$ (batch size free), we need to take $1 - \beta = \min(1, \frac{\sqrt{RL}}{\sigma\sqrt{T}})$ so that we have

25

$$\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{W}^{(t)})\|$$

$$\leq \left[2\sqrt{RTL(2m + 4m\delta + 2m\delta^2)} + 2\sqrt{10 \cdot \sigma} \cdot (RL)^{1/4}T^{3/4} + 10\sigma^2 \sqrt{\frac{T}{RL}} + 10\sqrt{\sigma}(RL)^{1/4}T^{3/4} \right] \frac{1}{1 - 4\sqrt{m}\delta},$$

thus

$$\frac{1}{T}\sum_{t=1}^{T} E\left[\|\nabla \mathcal{L}(\mathbf{W}^{(t)})\|\right] \leq \mathcal{O}\left(\left[\frac{\sqrt{RLn(2 + 4\delta + 2\delta^2)}}{\sqrt{T}} + \sigma \cdot \frac{(RL)^{1/4}}{T^{1/4}} + \frac{\sigma^2}{\sqrt{RLT}} + \frac{\sqrt{\sigma}(RL)^{1/4}}{T^{1/4}} \right] \frac{1}{1 - 4\sqrt{m}\delta} \right).$$

If we take β as an arbitrary constant in $(0, 1)$, then we will need to take $B = T$, so that

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla \mathcal{L}(\mathbf{W}^{(t)})\| \leq \mathcal{O}\left(\left[\frac{\sqrt{RLn(2 + 4\delta + 2\delta^2)}}{\sqrt{T}} + \frac{\sqrt{RL}}{\sqrt{T}} + \frac{\sigma}{T^{3/2}} + \frac{\sigma}{\sqrt{T}} \right] \frac{1}{1 - 4\sqrt{m}\delta} \right).$$

$\square$

Proof of Lemma 3.2. We denote $\mathbf{B} = \mathbf{A}\mathbf{A}^{\top}$, $\mathbf{X}_k$ the result after k Newton-Schultz iterations, $\mathbf{X}_0 = \frac{\mathbf{B}}{\|\mathbf{B}\|_2}$ and $\mathbf{O} = \mathbf{U}\mathbf{V}^{\top}$ where $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$ is the SVD decomposition with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m$ the singular values.

It is known that Newton-Schultz converges quadratically, so we have

$$\|\mathcal{E}_k\|_2 = \|\mathbf{X}_k - \mathbf{O}\|_2 \leq \|\mathbf{X}_0 - \mathbf{O}\|_2^{2^k}$$

We now bound $\|\mathbf{X}_0 - \mathbf{O}\|_2$. We know that $\mathbf{X}_0 = \frac{\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}}{\|\mathbf{B}\|_2} = \frac{\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}}{\sigma_1}$

$$\|\mathbf{X}_0 - \mathbf{O}\|_2 = \left\| \frac{\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}}{\sigma_1} - \mathbf{U}\mathbf{V}^{\top} \right\|_2 = \left\| \mathbf{U}(\frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I})\mathbf{V}^{\top} \right\|_2 = \left\| \frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I} \right\|_2$$

Where last equality is due to the fact that $\| \cdot \|_2$ is unitary invariant. The matrix $\frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I}$ is diagonal with values $\frac{\sigma_i}{\sigma_1} - 1$ on the diagonal. From that observation we get that

$$\left\| \frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I} \right\|_2 = \max_i \left| \frac{\sigma_i}{\sigma_1} - 1 \right| = \max_i \left(1 - \frac{\sigma_i}{\sigma_1} \right) = 1 - \frac{\sigma_m}{\sigma_1} = 1 - \frac{1}{\kappa}$$

For the Frobenius norm, we get a similar analysis. $\|\mathbf{X}_0 - \mathbf{O}\|_F = \left\| \frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I} \right\|_F$ since $\| \cdot \|_F$ is unitary invariant, so we just need to calculate $\left\| \frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I} \right\|_F$. It is known that $\left\| \frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I} \right\|_F \leq \sqrt{r} \left\| \frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I} \right\|_2$ so all in all we have

$$\left\| \frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I} \right\|_F \leq \sqrt{r} \left\| \frac{\mathbf{\Sigma}}{\sigma_1} - \boldsymbol{I} \right\|_2 = \sqrt{r} \left(1 - \frac{1}{\kappa} \right).$$

$\square$

Proof of Theorem 3.8. for any layer $j \in [L]$; in the following, for simplicity of notation, we ignore the index j and use $\mathbf{G}^{(t)}$ instead. By Lemma A.3, the low-rank optimization block 1 in Algorithm 1 is guaranteed to converge; we denote by $\mathsf{T}_\ell \in \mathbb{N}$ the time index t at which we enter block 1 for the ℓth time (i.e., $\|\hat{\mathbf{G}}_j^{(\mathsf{T}_\ell)}\| \leq \varsigma_2$), for $\ell \in \mathbb{N}$. Furthermore, we recall that $\mathbf{G}_j^{(t)} \triangleq \nabla_{\mathbf{W}_j} f\left(\boldsymbol{\theta}^{(t)}\right)$; when clear from the context, we omit j from $\mathbf{W}_j$, and use instead $\nabla_{\mathbf{W}_j} f\left(\boldsymbol{\theta}^{(t)}\right) = \nabla f\left(\mathbf{W}^{(t)}\right)$. Consider the SVD decomposition of the gradient $\nabla_{\mathbf{W}_j} f\left(\boldsymbol{\theta}_{\mathsf{T}_i}\right) = \mathbf{U}^{(\mathsf{T}_i)}\mathbf{\Sigma}^{(\mathsf{T}_i)}\mathbf{V}^{(\mathsf{T}_i)^{\top}}$. For $t \in [\mathsf{T}_i, \mathsf{T}_{i+1} - 1]$, we define the projected gradient as $\hat{\mathbf{G}}^{(t)} \triangleq \mathbf{P}^{(\mathsf{T}_i)}(r)\mathbf{G}^{(t)}$, where $\mathbf{P}^{(\mathsf{T}_i)}(r) = \mathbf{U}^{(\mathsf{T}_i)}[:, : r]^{\top}$, using the exact truncated-SVD calculation (in Block 1). For simplicity, we refer to $\mathbf{Q}^{(\mathsf{T}_i)}(r)$ as $\mathbf{Q}^{(\mathsf{T}_i)}$, and we

denote $\mathbf{P}^{(\mathsf{T}_i)} = \mathbf{Q}^{(\mathsf{T}_i)^\top} \mathbf{Q}^{(\mathsf{T}_i)}$. Next, let $h_t \triangleq f(\mathbf{W}^{(t)}) - f(\mathbf{W}^{(\mathsf{T}_{i+1})})$, and η_t denote the learning rate. Then,

$$
\begin{aligned}
h_{t+1} &= f(\mathbf{W}^{(t+1)}) - f(\mathbf{W}^{(\mathsf{T}_{i+1})}) \\
&= f\left(\mathbf{W}^{(t)} - \eta_t\left(\mathbf{W}^{(t+1)} - \mathbf{W}^{(t)}\right)\right) - f\left(\mathbf{W}^{(\mathsf{T}_{i+1})}\right) \\
&\leq f(\mathbf{W}^{(t)}) - f\left(\mathbf{W}^{(\mathsf{T}_{i+1})}\right) - \eta_t\langle\mathbf{G}^{(t)}, \mathbf{O}^{(t)}\rangle + \frac{\beta\eta_t^2}{2}\|\mathbf{O}^{(t)}\|_F^2 \\
&\underset{(1)}{\leq} f(\mathbf{W}^{(t)}) - f\left(\mathbf{W}^{(\mathsf{T}_{i+1})}\right) - \frac{\eta_t}{4}\|\hat{\mathbf{G}}^{(t)}\|_F^2 + \frac{5}{2}\cdot\frac{\eta_t\sigma}{\sqrt{m}} + \frac{\beta\eta_t^2 n}{2} \\
&\underset{(2)}{\leq} h_t - \frac{\eta_t}{4}\|\hat{\mathbf{G}}^{(t)}\|_F^2 + \frac{5}{2}\cdot\frac{\eta_t\sigma}{\sqrt{m}} + \frac{\beta\eta_t^2 n}{2}.
\end{aligned}
$$

where (1) follows Eq 7, and (2) follows Lemma A.3. Thus, summing over $t \in [\mathsf{T}_i, \mathsf{T}_{i+1})$, we get

$$
\sum_{t=\mathsf{T}_i}^{\mathsf{T}_{i+1}-1}(h_{t+1} - h_t) \leq \sum_{t=\mathsf{T}_i}^{\mathsf{T}_{i+1}-1}\left(-\frac{\eta_t}{4}\|\hat{\mathbf{G}}^{(t)}\|_F^2 + \frac{5}{2}\cdot\frac{\eta_t\sigma}{\sqrt{m}} + \frac{\beta\eta_t^2 n}{2}\right)
$$

$$
h_{\mathsf{T}_{i+1}} \leq h_{\mathsf{T}_i} - \sum_{t=\mathsf{T}_i}^{\mathsf{T}_{i+1}-1}\left(\frac{\eta_t}{4}\|\hat{\mathbf{G}}^{(t)}\|_F^2\right) + \sum_{t=\mathsf{T}_i}^{\mathsf{T}_{i+1}-1}\left(\frac{5}{2}\cdot\frac{\eta_t\sigma}{\sqrt{m}} + \frac{\beta\eta_t^2 n}{2}\right). \tag{3}
$$

Assume a constant learning rate $\eta_t = \eta$, and define $T_i := \mathsf{T}_{i+1} - \mathsf{T}_i$. For each interval $[\mathsf{T}_i, \mathsf{T}_{i+1}-1]$, we have,

$$
\sum_{t=\mathsf{T}_i}^{\mathsf{T}_{i+1}-1}\frac{\eta}{4}\|\hat{\mathbf{G}}^{(t)}\|_F^2 \leq h_{\mathsf{T}_i} - h_{\mathsf{T}_{i+1}} + T_i\left(\frac{5}{2}\cdot\frac{\eta\sigma}{\sqrt{m}} + \frac{\beta\eta^2 n}{2}\right).
$$

Summing over all $i = 1, \ldots, N$,

$$
\begin{aligned}
\sum_{i=1}^{N}\sum_{t=\mathsf{T}_i}^{\mathsf{T}_{i+1}-1}\frac{\eta}{4}\|\hat{\mathbf{G}}^{(t)}\|_F^2 &\leq \sum_{i=1}^{N}\left(h_{\mathsf{T}_i} - h_{\mathsf{T}_{i+1}}\right) + \sum_{i=1}^{N}T_i\left(\frac{5}{2}\cdot\frac{\eta\sigma}{\sqrt{m}} + \frac{\beta\eta^2 n}{2}\right) \\
&= h_{\mathsf{T}_0} - h_{\mathsf{T}_{N+1}} + T\left(\frac{5}{2}\cdot\frac{\eta\sigma}{\sqrt{m}} + \frac{\beta\eta^2 n}{2}\right),
\end{aligned}
$$

where $T = \sum_{i=1}^{N}T_i$ is the total number of iterations. Using $h_{\mathsf{T}_0} - h_{\mathsf{T}_{N+1}} \leq M$, we get,

$$
\sum_{i=1}^{N}\sum_{t=\mathsf{T}_i}^{\mathsf{T}_{i+1}-1}\|\hat{\mathbf{G}}^{(t)}\|_F^2 \leq \frac{4M}{\eta} + 2T\left(\frac{5\sigma}{\sqrt{m}} + \beta\eta n\right).
$$

Let $\eta = \sqrt{\frac{2M}{\beta n}}\cdot\frac{1}{\sqrt{T}}$. Then, the average squared norm of the projected gradients satisfies,

$$
\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\|\hat{\mathbf{G}}^{(t)}\|_F^2 &\leq \frac{4M}{\eta T} + 2\left(\frac{5\sigma}{\sqrt{m}} + \beta\eta n\right) \\
&= \frac{4M}{\left(\sqrt{\frac{2M}{\beta n}}\cdot\frac{1}{\sqrt{T}}\right)T} + 2\left(\frac{5\sigma}{\sqrt{m}} + \beta n\cdot\sqrt{\frac{2M}{\beta n}}\cdot\frac{1}{\sqrt{T}}\right) \\
&= \frac{4M\sqrt{\beta n}}{\sqrt{2M}}\cdot\frac{1}{T^{3/2}} + 2\cdot\frac{5\sigma}{\sqrt{m}} + 2\beta n\cdot\sqrt{\frac{2M}{\beta n}}\cdot\frac{1}{\sqrt{T}} \\
&= \frac{4\sqrt{2M\beta n}}{T^{3/2}} + \frac{10\sigma}{\sqrt{m}} + 2\sqrt{2M\beta n}\cdot\frac{1}{\sqrt{T}}.
\end{aligned}
$$

Thus, the bound becomes,

$$\frac{1}{T}\sum_{t=1}^{T}\|\hat{\mathbf{G}}^{(t)}\|_F^2 \le \frac{4\sqrt{2M\beta n}}{T^{3/2}} + \frac{10\sigma}{\sqrt{m}} + \frac{2\sqrt{2M\beta n}}{\sqrt{T}}.$$

Recall that for $\mathbf{P}^{(\mathsf{T}_i)}$, it holds that

$$\|\hat{\mathbf{G}}^{(\mathsf{T}_i)} - \mathbf{Q}^{(\mathsf{T}_i)}\hat{\mathbf{G}}^{(\mathsf{T}_i)}\|_F^2 \le \alpha\|\hat{\mathbf{G}}^{(\mathsf{T}_i)}\|_F^2,$$

for some $\alpha \in (0, 1]$. Then,

$$\|\mathbf{Q}^{(\mathsf{T}_i)\perp}\hat{\mathbf{G}}^{(\mathsf{T}_i)}\|_F^2 \le \frac{\alpha}{1-\alpha}\|\mathbf{Q}^{(\mathsf{T}_i)}\hat{\mathbf{G}}^{(\mathsf{T}_i)}\|_F^2.$$

From Lemma B.3 in [1], under $\eta \le \frac{2}{\lambda_{\max}}$, we get,

$$\mathbb{E}\|\hat{\mathbf{G}}^{(t)}\|_F^2 \le \mathbb{E}\|\hat{\mathbf{G}}^{(\mathsf{T}_i)}\|_F^2, \quad \forall t \in [\mathsf{T}_i, \mathsf{T}_{i+1}).$$

Hence,

$$
\begin{aligned}
\frac{1}{\mathsf{T}_N}\sum_{i=0}^{N-1}\sum_{t=\mathsf{T}_i}^{\mathsf{T}_{i+1}-1}\|\mathbf{G}^{(t)}\|_F^2 &\le \frac{1}{\mathsf{T}_N}\sum_{i=0}^{N-1}(\mathsf{T}_{i+1} - \mathsf{T}_i)\|\mathbf{G}^{(\mathsf{T}_i)}\|_F^2 \\
&\le \frac{1}{(1-\alpha)\mathsf{T}_N}\sum_{i=0}^{N-1}(\mathsf{T}_{i+1} - \mathsf{T}_i)\|\mathbf{Q}^{(\mathsf{T}_i)}\hat{\mathbf{G}}^{(\mathsf{T}_i)}\|_F^2 \\
&\le \frac{1}{(1-\alpha)\mathsf{T}_N}\left(\frac{4\sqrt{2M\beta n}}{\mathsf{T}_N^{3/2}} + \frac{10\sigma}{\sqrt{m}} + \frac{2\sqrt{2M\beta n}}{\sqrt{\mathsf{T}_N}}\right) \\
&\le \frac{4\sqrt{2M\beta n}}{(1-\alpha)\mathsf{T}_N^{5/2}} + \frac{10\sigma}{(1-\alpha)\mathsf{T}_N\sqrt{m}} + \frac{2\sqrt{2M\beta n}}{(1-\alpha)\mathsf{T}_N^{3/2}} \qquad (4)
\end{aligned}
$$

Accordingly, for any $\varepsilon > 0$, if T_N satisfies

$$\frac{4\sqrt{2M\beta n}}{(1-\alpha)\mathsf{T}_N^{5/2}} + \frac{10\sigma}{(1-\alpha)\mathsf{T}_N\sqrt{m}} + \frac{2\sqrt{2M\beta n}}{(1-\alpha)\mathsf{T}_N^{3/2}} \le \varepsilon,$$

a sufficient condition for this to hold is,

$$\mathsf{T}_N \ge \max\left\{\left(\frac{12\sqrt{2M\beta n}}{(1-\alpha)\varepsilon}\right)^{2/5}, \frac{30\sigma}{(1-\alpha)\varepsilon\sqrt{m}}, \left(\frac{6\sqrt{2M\beta n}}{(1-\alpha)\varepsilon}\right)^{2/3}\right\},$$

it follows that

$$\min_{0\le t\le\mathsf{T}_N}\|\mathbf{G}^{(t)}\|_F^2 \le \frac{1}{\mathsf{T}_N}\sum_{i=0}^{N-1}\sum_{t=\mathsf{T}_i}^{\mathsf{T}_{i+1}-1}\|\mathbf{G}^{(t)}\|_F^2 \le \varepsilon.$$

Thus, there exists an iteration index $t \in [0, \mathsf{T}_N]$ such that

$$\|\mathbf{G}^{(t)}\|_F^2 \le \varepsilon,$$

which, by definition, implies that the algorithm reaches an ε-critical point.

This concludes that Algorithm 1 achieves an ε-critical point.

$\square$

In the following, we provide the auxiliary lemma that is used in the proof of Theorem 3.8.

Lemma A.3 (Convergence of the Inner Fixed Low-Rank Optimization). *Consider the same setting and assumptions as in Theorem 3.8. Then, the second time $t = \mathsf{T}_\ell \in \mathbb{N}$ in which Algorithm 1 enters Block 1 (where it updates the projection matrix) happens for a finite $\ell \in \mathbb{N}$.*

Proof. By the β-smoothness of f, we have

$$f(\mathbf{W}^{(t+1)}) \leq f(\mathbf{W}^{(t)}) + \langle \mathbf{G}^{(t)}, \mathbf{W}^{(t+1)} - \mathbf{W}^{(t)} \rangle + \frac{\beta}{2} \|\mathbf{W}^{(t+1)} - \mathbf{W}^{(t)}\|_F^2.$$

Substituting the update rule $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta_t \mathbf{O}^{(t)}$, we get

$$f(\mathbf{W}^{(t+1)}) \leq f(\mathbf{W}^{(t)}) - \eta_t \langle \mathbf{G}^{(t)}, \mathbf{O}^{(t)} \rangle + \frac{\beta \eta_t^2}{2} \|\mathbf{O}^{(t)}\|_F^2. \tag{5}$$

Since $\hat{\mathbf{G}}^{(t)} = \mathbf{P}^{(t)}(r)^\top \mathbf{G}^{(t)}$ and $\mathbf{O}^{(t)} \in \mathrm{range}(\mathbf{P}^{(t)}(r)^\top)$, it holds that

$$\langle \mathbf{G}^{(t)}, \mathbf{O}^{(t)} \rangle = \langle \hat{\mathbf{G}}^{(t)}, \mathbf{O}^{(t)} \rangle.$$

By equation (2.8) in [41], we have

$$-\langle \hat{\mathbf{G}}^{(t)}, \mathbf{O}^{(t)} \rangle \leq -\frac{1}{4} \|\hat{\mathbf{G}}^{(t)}\|_F^2 + \frac{5}{2} \|\hat{\mathbf{G}}^{(t)} - \mathbf{M}^{(t)}\|_F^2.$$

Now we bound $\|\hat{\mathbf{G}}^{(t)} - \mathbf{M}^{(t)}\|_F$:

$$\begin{aligned}
\|\hat{\mathbf{G}}^{(t)} - \mathbf{M}^{(t)}\|_F &= \|\hat{\mathbf{G}}^{(t)} - \nabla f(\mathbf{W}^{(t)}) + \nabla f(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\|_F \\
&\leq \|\hat{\mathbf{G}}^{(t)} - \nabla f(\mathbf{W}^{(t)})\|_F + \|\nabla f(\mathbf{W}^{(t)}) - \mathbf{M}^{(t)}\|_F \\
&\leq \frac{\sigma}{\sqrt{m}}, \quad \text{(by assumptions A(1)-(2)).}
\end{aligned}$$

Substituting this into the previous inequality gives

$$-\langle \hat{\mathbf{G}}^{(t)}, \mathbf{O}^{(t)} \rangle \leq -\frac{1}{4} \|\hat{\mathbf{G}}^{(t)}\|_F^2 + \frac{5}{2} \cdot \frac{\sigma}{\sqrt{m}}. \tag{6}$$

Substituting into equation (5), we obtain

$$f(\mathbf{W}^{(t+1)}) \leq f(\mathbf{W}^{(t)}) - \frac{\eta_t}{4} \|\hat{\mathbf{G}}^{(t)}\|_F^2 + \frac{5}{2} \cdot \frac{\eta_t \sigma}{\sqrt{m}} + \frac{\beta \eta_t^2}{2} \|\mathbf{O}^{(t)}\|_F^2.$$

Since by definition $\|\mathbf{O}^{(t)}\|_F^2 \leq n$, we get

$$f(\mathbf{W}^{(t+1)}) \leq f(\mathbf{W}^{(t)}) - \frac{\eta_t}{4} \|\hat{\mathbf{G}}^{(t)}\|_F^2 + \frac{5}{2} \cdot \frac{\eta_t \sigma}{\sqrt{m}} + \frac{\beta \eta_t^2 n}{2}. \tag{7}$$

For constant step size $\eta_t = \eta$, summing over $t = 1$ to T, we get

$$f(\mathbf{W}^{(T+1)}) \leq f(\mathbf{W}^{(1)}) - \frac{\eta}{4} \sum_{t=1}^{T} \|\hat{\mathbf{G}}^{(t)}\|_F^2 + \frac{5}{2} \cdot \frac{\eta \sigma T}{\sqrt{m}} + \frac{\beta \eta^2 n T}{2}.$$

Rearranging and using $f(\mathbf{W}^{(1)}) - f^* \leq M$, we conclude

$$\frac{1}{T} \sum_{t=1}^{T} \|\hat{\mathbf{G}}^{(t)}\|_F^2 \leq \frac{4M}{\eta T} + 10 \cdot \frac{\sigma}{\sqrt{m}} + 2\eta \beta n.$$

$\square$

B   Additional Information

Definition B.1. (Reversibility [19]) A neural network ϕ that maps the input $\boldsymbol{x}$ to output $\boldsymbol{y} = \phi(\boldsymbol{x}; \theta)$ is reversible, if there exists $L(\boldsymbol{x}; \theta)$ so that $\boldsymbol{y} = L(\boldsymbol{x}; \theta)\boldsymbol{x}$, and the backpropagated gradient $\boldsymbol{g_x}$ satisfies $\boldsymbol{g_x} = L^\top(\boldsymbol{x}; \theta)\boldsymbol{g_y}$, where $\boldsymbol{g_y}$ is the backpropagated gradient at the output $\boldsymbol{y}$. $L(\boldsymbol{x}; \theta)$ depends on the input $\boldsymbol{x}$ and weight θ in the network ϕ.

Several critical observations regarding Algorithm 1 warrant attention. Initially, in order to minimize memory consumption, Algorithm 1 implements a per-layer weight update during the process of backpropagation, as advocated by contemporary studies, see, e.g., [45]. This approach contrasts with conventional optimizers, which typically update all weights after backpropagation by retaining the complete gradients in memory, a method potentially marked by significant inefficiency. Should there be a desire to generate an adapter (i.e., a parallel low-dimensional LoRA-type model) subsequent to fine-tuning, this can be achieved with efficiency through the following steps. Firstly, the training weights gap $\Delta \triangleq \mathbf{W}_{\text{Fine-Tuned}} - \mathbf{W}_{\text{Pretrained}}$ is computed, where $\mathbf{W}_{\text{Fine-Tuned}}$ denotes the model weight upon process completion, and $\mathbf{W}_{\text{Pretrained}}$ refers to the original model weight. Subsequently, $r_{\text{Adaptor}} \triangleq \text{rank}(\Delta)$ is determined utilizing a matrix ranking algorithm, followed by the resolution of $\min_{\mathbf{A} \in \mathbb{R}^{n \times r_{\text{Adaptor}}}, \mathbf{B} \in \mathbb{R}^{r_{\text{Adaptor}} \times m}} \|\Delta - \mathbf{AB}\|_F^2$ through any optimization algorithm (e.g., gradient descent). It is noteworthy that any solution to this matrix factorization optimization problem is well-known as a global optimum [54].

C Update Step Rule Formulation

Definition C.1. [Subspace-Aware Moment-Orthogonalization (SUMO)] SUMO formulates the subsequent gradient update rules. Refer to

$$\textbf{SUMO} \begin{cases} \hat{\mathbf{G}}^{(t)} = \mathbf{Q}^{(t)^\top} \nabla_{\mathbf{W}} f\left(\mathbf{W}_t; \xi_t\right) \mathbf{R}^{(t)} \\ \mathbf{M}^{(t+1)} = \beta \mathbf{M}^{(t)} + (1-\beta)\hat{\mathbf{G}}^{(t)} \\ \mathbf{O}^{(t+1)} = \text{Orthogonalization\_SVD}\left(\mathbf{M}^{(t+1)}\right) \\ \mathbf{W}^{(T)} = \mathbf{W}^{(0)} + \eta \sum_{t=0}^{T-1} \left(\mathbf{G}^{(t)} - \mathbf{Q}^{(t)}(\hat{\mathbf{G}}^{(t)} - \mathbf{O}^{(t+1)})\mathbf{R}^{(t)^\top}\right), \end{cases}$$

with $\mathbf{Q}_t \in \mathbb{R}^{m \times r}$ and $\mathbf{R}_t \in \mathbb{R}^{r \times n}$ denoting projection matrices, $T \in \mathbb{N}$ representing the subspace update interval, η indicating the learning rate, ξ_t constituting a stochastic batch, and Orthogonalization_SVD($\mathbf{A}$) as the operator that resolves the following through Singular Value Decomposition (SVD), as described in

$$\arg\min_{\mathbf{O}}\{\|\mathbf{O} - \mathbf{A}\|_F : \text{either } \mathbf{O}^T\mathbf{O} = \mathbf{I} \text{ or } \mathbf{OO}^T = \mathbf{I}\}.$$

D Additional Experiments

In Table 7, we evaluated SUMO and state-of-the-art memory-efficient fine-tuning methods on the MAWPS[55] dataset using the LLaMA2-7B model. We report results across two rank settings (32 and 128), comparing training time, memory usage, and task accuracy. SUMO consistently achieves superior accuracy while maintaining competitive efficiency in both memory and time (comparing to Galore).

Table 7: Fine-tuning LLaMA2-7B on MAWPS[55]

Methods	Rank	Time(h) ↓	Memory (GB) ↓	Accuracy (%) ↑
LoRA	32	**0.40**	14.36	45.80
DoRA	32	0.69	15.01	44.96
GaLore	32	2.59	15.15	58.40
SUMO (Newton-Shultz5)	32	1.83	**13.86**	**58.47**
SUMO (SVD)	32	1.56	**13.86**	**61.23**
LoRA	128	**0.45**	15.64	65.97
DoRA	128	0.72	16.17	66.81
GaLore	128	2.61	15.79	64.29
SUMO (Newton-Shultz5)	128	1.78	**14.12**	64.41
SUMO (SVD)	128	1.62	**14.12**	**68.03**

Comparison with Muon. Table 8 below complements Table 2 by comparing full fine-tuning, vanilla Muon, and our SUMO; results are reported as mean ± standard deviation when available.

Table 8: Additional comparison to Table 2: full fine-tuning vs. vanilla Muon and our SUMO (mean ± std when reported).

Model	Memory	CoLA	STS-B	MRPC	RTE	SST-2
Full Fine-Tuning	747M	62.24	90.92	91.30	79.42	94.57
Muon Full Fine Tuning	458M	61.19	90.98	92.14	80.83	94.71
SUMO (Newton-Schulz5, rank=4)	197M	61.81 ± 0.02	90.81 ± 0.013	92.43 ± 0.034	79.33 ± 0.031	94.14 ± 0.028
SUMO (SVD, rank=4)	197M	62.32 ± 0.015	91.05 ± 0.007	93.48 ± 0.022	81.08 ± 0.019	94.93 ± 0.01
SUMO (Newton-Schulz5, rank=8)	198M	61.73 ± 0.021	90.77 ± 0.032	91.93 ± 0.04	79.66 ± 0.03	94.13 ± 0.025
SUMO (SVD, rank=8)	198M	61.69 ± 0.014	91.11 ± 0.02	93.72 ± 0.018	81.38 ± 0.011	94.83 ± 0.01

These results show that our SUMO, achieves better performance with a significantly smaller memory footprint compared to Muon full fine-tuning approach.

Hyperparameters Grid search. To evaluate the impact of Subspace Update Frequency (K) and Ranks (r), we performed a grid search during the pretraining of the LLaMA 130M model on the C4 dataset. This specific setup allows for a direct comparison with the Galore method. Table: Perplexity results from a grid-search of Subspace Update Frequency (K) and Ranks (r) for the LLaMA 130M model pretrained on the C4 dataset. Values are presented as Galore/SUMO.

Table 9: Perplexity from a grid-search over Subspace Update Frequency (K) and Rank (r) for LLaMA-130M on C4. Values are **Galore/SUMO**

Update Frequency	Rank = 128	Rank = 256	Rank = 512
100	29.7/**28.27**	27.9/**26.74**	27.4/**26.73**
250	28.1/**27.86**	26.5/**24.87**	26.2/**24.82**
500	27.2/**25.91**	25.6/**24.98**	25.3/**24.31**
1k	**26.8**/25.83	**25.1**/25.42	**24.8**/24.93

D.1 Details of benchmarks in Table 6

Specifically, the pretrained models is evaluated on the following tasks:

- **Perplexity:** Measured on the C4 dataset [46].

- **Commonsense Reasoning:** BoolQ [56], RTE [57], HellaSwag (HS) [58], Winogrande (WG) [59], OpenBookQA (OBQA) [60], ARC-Easy (ARC-E), and ARC-Challenge (ARC-C) [61].

- **Physical and Scientific Reasoning:** PIQA [62], SciQ [63], and MathQA [64].

D.2 Details of Fine-Tuning on GLUE

We fine-tune the pre-trained RoBERTa-Base model on the GLUE benchmark using the model provided by the Hugging Face. In Table 10 and, we detail the hyper parameters used in fine-tuning.

	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
Batch Size	16	16	16	32	16	16	16	16
# Epochs	30	30	30	30	30	30	30	30
Learning Rate	1E-05	1E-05	3E-05	3E-05	1E-05	1E-05	1E-05	1E-05
Rank Config.				$r = 4$				
Projection back scale				4				
Max Seq. Len.				512				

Table 10: Hyperparameters of fine-tuning RoBERTa base for the comparison in Table 2 with respect only to rank=4.

	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
Batch Size	16	16	16	32	16	16	16	16
# Epochs	30	30	30	30	30	30	30	30
Learning Rate	1E-05	2E-05	2E-05	1E-05	1E-05	2E-05	2E-05	3E-05
Rank Config.				$r = 8$				
Projection back scale				2				
Max Seq. Len.				512				

Table 11: Hyperparameters of fine-tuning RoBERTa base for the comparison in Table 2 with respect only to rank=8.

D.3 Comparison with vanilla Muon

Following we conducted a full fine-tuning experiment with vanilla Muon to complete the comparison for Table 2 in the paper, as presented below.

Model	Memory	CoLA	STS-B	MRPC	RTE	SST-2
Full Fine-Tuning	747M	62.24	90.92	91.30	79.42	94.57
Muon Full Fine Tuning	458M	61.19	90.98	92.14	80.83	94.71
SUMO (Newton-Schulz5, rank=4)	197M	61.81 ± 0.02	90.81 ± 0.013	92.43 ± 0.034	79.33 ± 0.031	94.14 ± 0.028
SUMO (SVD, rank=4)	197M	62.32 ± 0.015	91.05 ± 0.007	93.48 ± 0.022	81.08 ± 0.019	94.93 ± 0.01
SUMO (Newton-Schulz5, rank=8)	198M	61.73 ± 0.021	90.77 ± 0.032	91.93 ± 0.04	79.66 ± 0.03	94.13 ± 0.025
SUMO (SVD, rank=8)	198M	61.69 ± 0.014	91.11 ± 0.02	93.72 ± 0.018	81.38 ± 0.011	94.83 ± 0.01

Table 12: Comparison with vanilla Muon fine-tuning results.