THE NUCLEAR NORM AND BEYOND: DUAL NORMS AND COMBINATIONS FOR MATRIX OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this article, we explore the use of matrix norms for optimizing functions of weight matrices, a crucial problem in training large language models. Moving beyond the spectral norm that underlies the Muon update, we leverage the nuclear norm, its affine combinations with other norms, and their corresponding duals to develop a new family of Muon-like algorithms. We complement our theoretical analysis with an extensive empirical study of the algorithms across a wide range of tasks and settings.

1 INTRODUCTION

Minimizing loss functions in unprecedentedly high-dimensional spaces has recently become an integral and crucial part in training large language models. Hence, new scalable, time- and memory-efficient algorithms have been demanded. Besides well-known Adam and AdamW (citations), recently proposed Muon has shown promising results on different benchmarks (CIFAR, NanoGPT, citations). Its key difference from earlier algorthims is that it has been constructed specifically for optimizing functions of weight matrices. Such functions are common in modern machine learning (citations), so it does not significantly restrict Muon's applicability to modern problems.

That is what can be said from a practical point of view. From the perspective of theory, Muon's main innovation was an intentional usage of matrix norms, i.e. the spectral norm, to derive the algorithm's update (cite Deriving Muon). Several other attempts have been since made to construct new algorithms, mainly generalizations of Muon's paradigm (cite Scion and Gluon).

Based upon recent theoretical advances that explain some theory behind Muon, Scion and Gluon (cite), we explore application of other matrix norms to optimization of functions of matrices. As it has been done with Muon, we stipulate that our algorithms' updates be fast to compute.

2 PROBLEM STATEMENT

2.1 FUNCTION OF A MATRIX AND ASSUMPTIONS

Warning: the text was copied, while formulas were adapted. We need to rephrase the subsection! We consider the problem of minimizing a differentiable matrix function $F(\cdot)\colon \mathbb{R}^{m\times n} \to \mathbb{R}$

$$\min_{\boldsymbol{X}\in\mathbb{R}^{m\times n}} f(\boldsymbol{X}). \tag{1}$$

To make the theoretical analysis possible in the future, we make three reasonable non-restrictive assumptions. Idea: move gradient assumptions and smoothness to the L-smooth part. Here we need only custom norm and their relation to the Frobenius norm.

Stochastic gradient estimator. We assume access to a stochastic estimator $g(\cdot\,;\xi)\colon \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ of the gradient $\nabla f(\cdot)$, where $\xi \sim \mathcal{D}$ is a random variable sampled from a probability distribution $\mathcal{D}$. We assume that the stochastic gradient estimator $g(\cdot;\xi)$ is unbiased and has bounded variance, that is, the following relations hold:

$$\mathbb{E}_{\xi\sim\mathcal{D}}\left[g(\boldsymbol{X};\xi)\right] = \nabla f(\boldsymbol{X}) \quad \text{and} \quad \mathbb{E}_{\xi\sim\mathcal{D}}\left[\|g(\boldsymbol{X};\xi) - \nabla f(\boldsymbol{X})\|_{\mathrm{F}}^2\right] \le \sigma^2 \quad \text{for all } \boldsymbol{X}\in\mathbb{R}^{m\times n}, \tag{A1}$$

where $\sigma > 0$ is a positive variance parameter, and $\|\cdot\|_{\mathrm{F}}$ is the standard Euclidean, i.e. Frobenius, norm induced by the inner product $\langle \cdot, \cdot \rangle$, i.e., $\|\boldsymbol{X}\|_{\mathrm{F}} = \sqrt{\langle \boldsymbol{X}, \boldsymbol{X} \rangle} = \sqrt{\mathrm{tr}(\boldsymbol{X}^\top \boldsymbol{X})}$. These assumptions have been widely adopted for the analysis of many stochastic gradient optimization algorithms (**??????**).

Non-Euclidean norm setting and Lipschitz continuous gradient. We assume that matrix space $\mathbb{R}^{m \times n}$ is equipped with a norm $\|\cdot\| \colon \mathbb{R}^{m \times n} \to \mathbb{R}_+$, which possibly does not coincide with the Frobenius, norm $\|\cdot\|_{\mathrm{F}}$. In addition, we assume that the gradient $\nabla f(\cdot)$ is Lipschitz continuous with respect to the norm $\|\cdot\|$, that is, the following inequality holds:

$$\|\nabla f(\boldsymbol{X}) - \nabla f(\boldsymbol{X}')\|_* \leq L \|\boldsymbol{X} - \boldsymbol{X}'\| \quad \text{for all } \boldsymbol{X}, \boldsymbol{X}' \in \mathbb{R}^{m \times n}, \tag{A2}$$

where $L > 0$ is the gradient Lipschitz constant, and $\|\cdot\|^\dagger \colon \mathbb{R}^{m \times n} \to \mathbb{R}_+$ is the dual norm associated with $\|\cdot\|$, i.e., $\|\boldsymbol{X}\|^\dagger = \sup_{\|\boldsymbol{X}'\| \leq 1} \langle \boldsymbol{X}, \boldsymbol{X}' \rangle$ for all $\boldsymbol{X} \in \mathbb{R}^{m \times n}$. The assumption of gradient Lipschitz continuity is also widespread in the analysis of first-order optimization methods (**?????**). It is important to highlight that while Assumption (A2) uses the dual norm $\|\cdot\|^\dagger$ to measure the difference between the gradients, the variance in Assumption (A1) is measured with respect to the Frobenius norm $\|\cdot\|_{\mathrm{F}}^2$, which is necessary to properly utilize the unbiasedness property of the stochastic gradient estimator $g(\cdot; \xi)$. Therefore, we need to provide a connection between these norms using the following inequality:

$$\|\boldsymbol{X}\|^\dagger \leq \rho \cdot \|\boldsymbol{X}\|_{\mathrm{F}} \quad \text{for all } \boldsymbol{X} \in \mathbb{R}^{m \times n}, \tag{A3}$$

where $\rho > 0$ is a positive constant. Note that such a constant always exists due to the norm equivalence theorem, which always holds in the finite-dimensional space $\mathbb{R}^{m \times n}$. We recount ρ for different norms $\|\cdot\|$ in the appendix.

2.2 Linear Minimization Oracle and Trust Region

Let us look at the problem from the perspective of linear minimization oracle (lmo) and unconstrained stochastic conditional gradient descent (uSCG) (Pethick et al. (2025)). lmo is defined as:

$$\mathrm{lmo}(\boldsymbol{S}) \in \arg\min_{\boldsymbol{X} \in \mathcal{S}} \langle \boldsymbol{S}, \boldsymbol{X} \rangle, \tag{2}$$

where $\mathcal{S}$ is some set. We are interested in the case when $\mathcal{S}$ is a ball in our $\|\cdot\|$ norm:

$$\mathcal{S} := \mathcal{B}_\eta := \{\boldsymbol{X} \mid \|\boldsymbol{X}\| \leq \eta\}. \tag{3}$$

uSCG update is defined as: $\boldsymbol{X}^{k+1} = \boldsymbol{X}^k + \gamma_k lmo(\boldsymbol{M}^k)$, where $\boldsymbol{M}^{k+1} = (1 - \alpha_{k+1})\boldsymbol{M}^k + \alpha_{k+1} g(\boldsymbol{X}^k, \xi_k)$ is a momentum.

It can be easily shown that the formula is equvalent to

$$\boldsymbol{X}^{k+1} = \boldsymbol{X}^k - \gamma_k \eta \arg\max_{\boldsymbol{X} \in \mathcal{B}_1} \langle \boldsymbol{S}, \boldsymbol{X} \rangle = \boldsymbol{X}^k - \gamma_k \eta \{\Delta \in \mathcal{B}_1 \mid \langle \boldsymbol{M}^k, \Delta \rangle = \|\boldsymbol{M}^k\|^\dagger\}. \tag{4}$$

Let us set $\gamma_k = 1$. This transforms algorithm defined by eq. (4) into Algorithm 1 from Kovalev (2025). Therefore, we can view eq. (4) both as an lmo-based algorithm and as a trust-region algorithm.

3 Different Norms $\|\cdot\|$ imply different updates

Based on different norms $\|\cdot\|$, we simplify the update defined by the aforementioned equation:

$$\boldsymbol{X}^{k+1} = \boldsymbol{X}^k - \eta \{\Delta \in \mathcal{B}_1 \mid \langle \boldsymbol{M}^k, \Delta \rangle = \|\boldsymbol{M}^k\|^\dagger\} \tag{5}$$

In all the work, we define $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, $\boldsymbol{V}^\top$ as components of the singular value decomposition of $\boldsymbol{M}^k$: $\boldsymbol{M}^k = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$. We use common notations: $\boldsymbol{U} = [u_1, u_2, \ldots, u_r]$, $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$, and $\boldsymbol{V} = [v_1, v_2, \ldots, v_r]$.

$\|\boldsymbol{M}^k\|_{\mathrm{F}}$ and NSGD

Lemma. When $\|\cdot\| = \|\cdot\|_{\mathrm{F}}$, eq. (5) turns into:

$$\boldsymbol{X}^{k+1} = \boldsymbol{X}^k - \eta \frac{\boldsymbol{M}^k}{\|\boldsymbol{M}^k\|_{\mathrm{F}}} \tag{6}$$

It is an interesting observation, because in other works (Pethick et al. (2025)), $\|\cdot\|_{\mathrm{F}}$ was used to recover SGD. The difference is in how one states the problem.

$\|\boldsymbol{M}^k\|_{\mathrm{op}}$ and Muon

Lemma. When $\|\cdot\| = \|\cdot\|_{\mathrm{op}}$, eq. (5) turns into:

$$\boldsymbol{X}^{k+1} = \boldsymbol{X}^k - \eta UV^\top \tag{7}$$

$\|\boldsymbol{M}^k\|_{\mathrm{nuc}}$ and Neon

Lemma. When $\|\cdot\| = \|\cdot\|_{\mathrm{nuc}}$, eq. (5) turns into:

$$\boldsymbol{X}^{k+1} = \boldsymbol{X}^k - \eta u_1 v_1^\top \tag{8}$$

We name the derived algorithm *Neon*. In the section Matrix side of updates, we will discuss how to compute an update efficiently.

$\|\boldsymbol{M}^k\|^\dagger_{\mathrm{F}*}$ and F-Muon We define $\|\cdot\|_{\mathrm{F}*}$ as a convex combination of $\|\cdot\|_{\mathrm{nuc}}$ and $\|\cdot\|_{\mathrm{F}}$:

$$\|\boldsymbol{X}\|_{\mathrm{F}*} = \alpha\|\boldsymbol{X}\|_{\mathrm{nuc}} + (1-\alpha)\|\boldsymbol{X}\|_{\mathrm{F}}, \tag{9}$$

where $\alpha \in [0, 1]$ defines a specific norm of $F*$-family.

Lemma. When $\|\cdot\| = \|\cdot\|^\dagger_{\mathrm{F}*}$, eq. (5) turns into:

$$\boldsymbol{X}^{k+1} = \boldsymbol{X}^k - \eta(\alpha UV^\top + (1-\alpha)\frac{\boldsymbol{M}^k}{\|\boldsymbol{M}^k\|_{\mathrm{F}}}) \tag{10}$$

We name the derived algorithm *F-Muon*. It turns out that F-Muon is a convex combination of Normalized SGD and Muon, which is curious. The implications are significant and discussed in the following sections.

$\|\boldsymbol{M}^k\|^\dagger_{\mathrm{F}2}$ and F-Neon We define $\|\cdot\|_{\mathrm{F}2}$ as a convex combination of $\|\cdot\|_{\mathrm{op}}$ and $\|\cdot\|_{\mathrm{F}}$:

$$\|\boldsymbol{X}\|_{\mathrm{F}2} = \alpha\|\boldsymbol{X}\|_{\mathrm{op}} + (1-\alpha)\|\boldsymbol{X}\|_{\mathrm{F}}, \tag{11}$$

where $\alpha \in [0, 1]$ defines a specific norm of $F2$-family.

Lemma. When $\|\cdot\| = \|\cdot\|^\dagger_{\mathrm{F}2}$, eq. (5) turns into:

$$\boldsymbol{X}^{k+1} = \boldsymbol{X}^k - \eta(\alpha u_1 v_1^\top + (1-\alpha)\frac{\boldsymbol{M}^k}{\|\boldsymbol{M}^k\|_{\mathrm{F}}}) \tag{12}$$

We name the derived algorithm *F-Neon*. It turns out that F-Neon is a convex combination of Normalized SGD and Neon, which is curious. The implications are significant and discussed in the following sections.

TO-DO: F*-neon? (it would be hard to do it)

3.1 F*-NEON

Motivation: we need updates of higher rank. Hence, F* (why? regularization?)

4 MATRIX SIDE OF UPDATES

4.1 THEORY

Not only formulas, but also citations! Solutions to efficiently find some parts of SVD:

- Basic SVD (complexity asymptotic!)
- Newton-Schulz (complexity asymptotic!)
- Our job: Power iterations, Randomized SVD and Lanczos

4.2 EXPERIMENTS

Nikolay's task: SVD, RSVD, Lanczos, Newton-Schulz. We use torch and cupy to test it. Options: all possible on torch, or additionally to compare all on cupy. The goal: to find how much do we lose due to cupy

5 "SMOOTH" THEORY VIA TRUST REGION BOUNDS

- Probably statement of the trust region problem
- Assumption on unbiasedness and limited variance of stoch. grad.
- Assumption on L-smoothness as in the Kovalev's article.
- Assumption on rho. To be more precise, the statement of lemmas on rho for all 4 norms.
- We utilize the Kovalev's corollary to derive convergence
- Bonus: convert all the norms into a common sense (L2?) norm to get the true dependency on dimensionality

6 EXPERIMENTS

6.1 SYNTHETIC QUADRATICS

Description of how to construct L-smooth functions in a given norm, and how to estimate variance of the gradient. Then check of the theory. It must be the only experiment where we might beat vanilla Muon.

6.2 MLP

Deep learning?

6.3 BENCHMARKS

CNN benchmark NanoGPT benchmark

7 CONCLUSION

- Future work: (L0, L1)-smoothness, probably extrapolation (to make dependency on eps milder, but we don't to spend memory here so never mind), hyperparameter-free (Bernstein2023)

8 APPENDIX

8.1 NORM DERIVATIONS

8.2 UPDATES DERIVATIONS

Derivation of eq. (7) follows from eq. (10) with $\alpha = 1$. Indeed, $\|\cdot\|_{\text{op}}^{\dagger} = 1 \cdot \|\cdot\|_{\text{nuc}} + 0 \cdot \|\cdot\|_{\text{F}}$.

Derivation of eq. (10): Since $\|\cdot\|^\dagger = \|\cdot\|_{\mathrm{F}*}^{\dagger\dagger} = \|\cdot\|_{\mathrm{F}*}$, the goal is to reach $\|\boldsymbol{M}^k\|_{\mathrm{F}*} = \alpha \operatorname{tr} \boldsymbol{\Sigma} + (1-\alpha)\|\boldsymbol{M}^k\|_{\mathrm{F}}$.

Let us note that $\Delta = \alpha(\boldsymbol{U}\boldsymbol{V}^\top) + (1-\alpha)\frac{\boldsymbol{M}^k}{\|\boldsymbol{M}^k\|_{\mathrm{F}}}$ delivers this value. Indeed, by the trace property, $\langle \boldsymbol{M}^k, \Delta\rangle = \langle \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \alpha\boldsymbol{U}\boldsymbol{V}^\top + (1-\alpha)\frac{\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top}{\|\boldsymbol{M}^k\|_{\mathrm{F}}}\rangle = \alpha \operatorname{tr} \boldsymbol{\Sigma} + (1-\alpha)\|\boldsymbol{M}^k\|_{\mathrm{F}} = \|\boldsymbol{M}^k\|_{\mathrm{F}*}$, which completes the proof.

Derivation of eq. (8) follows from eq. (12) with $\alpha = 1$. Indeed, $\|\cdot\|_{\mathrm{nuc}}^\dagger = 1 \cdot \|\cdot\|_{\mathrm{op}} + 0 \cdot \|\cdot\|_{\mathrm{F}}$.

Derivation of eq. (12): Since $\|\cdot\|^\dagger = \|\cdot\|_{\mathrm{F2}}^{\dagger\dagger} = \|\cdot\|_{\mathrm{F2}}$, the goal is to reach $\|\boldsymbol{M}^k\|_{\mathrm{F2}} = \alpha\sigma_1 + (1-\alpha)\|\boldsymbol{M}^k\|_{\mathrm{F}}$.

Let us note that $\Delta = \alpha(u_1 v_1^\top) + (1-\alpha)\frac{\boldsymbol{M}^k}{\|\boldsymbol{M}^k\|_{\mathrm{F}}}$ delivers this value. Indeed, by the trace property and singular vectors orthogonality, $\langle \boldsymbol{M}^k, \Delta\rangle = \langle \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \alpha u_1 v_1^\top + (1-\alpha)\frac{\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top}{\|\boldsymbol{M}^k\|_{\mathrm{F}}}\rangle = \alpha \operatorname{tr} \operatorname{diag}(\sigma_1, 0, \ldots, 0), +(1-\alpha)\|\boldsymbol{M}^k\|_{\mathrm{F}} = \|\boldsymbol{M}^k\|_{\mathrm{F2}}$, which completes the proof.

8.3

Note: nuSAM and their update. They had no Lanczos Technical details on experiments

9 IDEA

9.1 PROBLEM (PROJECT DESCRIPTION)

In this subsection, we provide a more detailed description of our idea and formulate it as a mathematical problem. The authors of Bernstein & Newhouse (2024) suggest obtaining the update step as a solution to the optimization problem:

$$\langle g, \delta w\rangle + \lambda\|\delta w\|^2 \to \min_{\delta w}, \tag{13}$$

where w is the weight vector, g is a gradient-like vector (e.g., obtained via momentum SGD), and $\|\cdot\|$ represents a certain norm. Many popular optimizers, such as Adam (with exponential moving average disabled) and vanilla SGD, can be cast within this framework Bernstein & Newhouse (2024).

In large language models, most weights are structured as matrices, which offers additional opportunities for optimization. Let W be the weight matrix of a linear layer, and G be a gradient-like matrix. Then, the update step δW can be obtained as a solution to the optimization problem:

$$\langle G, \delta W\rangle + \lambda\|\delta W\|^2 \to \min_{\delta W}, \tag{14}$$

where $\|\cdot\|$ denotes a certain matrix norm. By setting this norm to the RMS-to-RMS norm (a scaled version of the spectral norm), we recover the Muon optimizer Bernstein (2025); Bernstein & Newhouse (2024) with an update step defined by:

$$\delta W = -\frac{1}{\lambda}\sqrt{\frac{n}{m}}UV^T, \tag{15}$$

where m is the input dimension of the layer, n is the output dimension, and U and V are obtained from the singular value decomposition of the gradient matrix $G = U\Sigma V$.

Motivated by the recent achievements of the Muon optimizer (e.g., Liu et al. (2025)), we consider alternative choices of norms, specifically the nuclear norm $\|\cdot\|_*$ and a custom $F*$ norm, given by

$$\|X\|_{F*}^2 = \frac{\|X\|_F^2 + \|X\|_*^2}{2}, \tag{16}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

Using the nuclear norm in equation 14 leads to a rank-one update of the weight matrices:

$$\delta W = -\frac{1}{2\lambda}u_1\sigma_1 v_1^T, \tag{17}$$

where σ_1 is the largest singular value, and u_1 and v_1 are the corresponding singular vectors. We expect one iteration of this method to be significantly faster than one iteration of Muon.

Another choice is the $F*$ norm. With this choice, equation 14 yields

$$\delta W = -\frac{1}{\lambda} U D V^T \tag{18}$$

with $D = \text{diag}(d_i)$, where $d_i = [\sigma_i - \tau]_+$, and τ is given by

$$\sum_i [\sigma_i - \tau]_+ = \tau \,. \tag{19}$$

We anticipate that the method with this update step will perform well with large batch sizes.

In this article we show how one can quickly compute weight updates defined by equation 17 or equation 18. Then we finilize the method by adding momentum and test their performance against those of Muon at training multilayer perceptron and transformer. The results will be fast algorithm, which we will convert into a new optimizer classes for PyTorch, as was done with Muon.

10 Related Work

Our review primarily focuses on the Muon and Shampoo optimizers, as our algorithm extends the ideas used to derive these methods. We highlight the advantages and disadvantages of these approaches, the unique effects they introduce, and compare them to Neon.

10.1 Muon optimizer

In the previous section, we described the theoretical foundation behind the weight update step in the Muon optimizer, but we did not discuss how to obtain the required matrices in practice. The update step is defined by equation 15, which requires UV^T with U and V^T from the singular value decomposition of the gradient-like matrix G. A naive solution would involve computing the SVD of G and constructing the required expression. However, the developers of the Muon optimizer introduced a workaround using Newton-Schulz iterations Jordan et al. (2024). The Newton-Schulz iterations from the original article Jordan et al. (2024) require 10 matrix-matrix multiplications to achieve the desired accuracy. The asymptotic complexity of such an operation is identical to that of SVD and equals $O(mn \min\{m, n\})$ for an $m \times n$ matrix. Nevertheless, matrix multiplication on modern GPUs can be performed much more efficiently.

The performance of Muon in training large language models was tested Liu et al. (2025) against AdamW. The testing demonstrated excellent performance by Muon, which was approximately 2 times more efficient in terms of FLOPs required to reach a certain loss value. This is even more remarkable when considering the cost of one iteration: Muon requires an additional $O(mn \min\{m, n\})$ FLOPs per $m \times n$ matrix, while AdamW needs only $O(mn)$.

Another interesting discovery about the Muon optimizer is that it accelerates grokking Tveit et al. (2025). In the test problem, Muon achieved grokking significantly faster than AdamW in terms of passed epochs, with a mean grokking epoch of 102.89 for Muon and 153.09 for AdamW. The authors suggest that this may be due to the fact that Muon stimulates broader exploration by orthogonalizing the gradient matrix, thus avoiding memorization.

Recently, theoretical guarantees for Muon convergence have been derived Li & Hong (2025). In particular, in the L-smooth convex case, it achieves $O(1/T^{\frac{1}{2}})$ (with full gradient) and $O(1/T^{\frac{1}{4}})$ (with stochastic gradient) bounds on the Frobenius norm of the gradient or the mathematical expectation of the gradient norm, respectively, where T is the number of iterations.

10.2 Shampoo optimizer

Another optimizer that exploits the matrix (and even tensor) structure of weights in neural networks is the Shampoo optimizer. We avoid a detailed description of this method here and refer the reader to the original article Gupta et al. (2018), but we outline the key properties of Shampoo and its relation to the Muon optimizer.

The Shampoo optimizer uses left and right preconditioning for the gradient-like matrix, leveling its spectrum. The preconditioners are computed from the exponentially averaged gradients, and their computation requires $O(n^3 + m^3)$ per $m \times n$ matrix. This exponential averaging is a key feature that provides several distinct interpretations for the preconditioners. They can be viewed as an approximation of the Gauss-Newton component of the Hessian or the first step of the power iteration algorithm for computing the optimal Kronecker product approximation Morwani et al. (2025). With exponential averaging turned off, the update step of Shampoo can be simplified and becomes identical to that of Muon Jordan et al. (2024).

Convergence analysis is presented in the original article Gupta et al. (2018). Shampoo achieves $O(1/T^{\frac{1}{2}})$ convergence for the loss function value in the L-smooth convex case, where T is the number of iterations.

10.3 Synthesis and Neon's position

Recent developments in optimization techniques show that utilizing the matrix structure of weights in neural networks can be very beneficial. Optimizers following this path converge faster in terms of iterations or epochs and often even FLOPs, but have a high iteration cost. Neon seeks to decrease the iteration cost while preserving fast convergence.

While the uniqie advantages and effects introduced by Neon are yet to be discovered, we can already say that our new optimizer introduces additional overhead of $O(mn)$ FLOPS on average per $m \times n$ weight matrix, which is much better then $O(mn \min\{m, n\})$ for Muon optimizer and $O(n^3 + m^3)$ for Shampoo.

10.4 Optimization Strategies for Efficient Large Language Model Training

The unprecedented scale of modern Large Language Models (LLMs) has pushed traditional optimizers like AdamW Loshchilov & Hutter (2017) to their limits in terms of computational efficiency and convergence speed Liu et al. (2025); Chen et al. (2025). This challenge has catalyzed research into more sophisticated optimization approaches that can maintain or improve performance while reducing training costs.

The Muon optimizer has emerged as a promising alternative based on matrix orthogonalization principles. Scaling Muon to billion-parameter LLMs required two key adaptations: the integration of L2 weight decay for stability and the implementation of per-parameter update scaling to handle diverse parameter distributions efficiently . Empirical evaluations demonstrate that Muon can match or exceed AdamW's model quality while requiring only about 52 of the training FLOPs. The successful training of the Moonlight model series, including a 16B-parameter Mixture-of-Experts model, validates Muon's practicality for production-scale applications Liu et al. (2025).

Building on this foundation, hybrid approaches like COSMOS Chen et al. (2025) further enhance efficiency by combining optimization techniques based on gradient structure. COSMOS applies computationally intensive updates to a low-dimensional "leading eigensubspace" while using memory-efficient methods like Muon for the remaining parameters. This approach maintains convergence benefits while substantially reducing memory requirements. For distributed training environments, optimizers like Dion Ahn & Xu (2025) specifically target communication efficiency by minimizing data exchange between workers through distributed orthonormalization techniques.

11 Derivation of update rules

Lemma 1. *Let $G \in \mathbb{R}^{m \times n}$ and $\lambda > 0$. Then, the following optimization problem*

$$f(\delta W) = \langle G, \delta W \rangle + \lambda \|\delta W\|_*^2 \to \min_{\delta W}$$

has solution

$$\delta W = -\frac{1}{2\lambda} u_1 \sigma_1 v_1^T \,,$$

where σ_1 is the largest singular value of G, and u_1 and v_1 are the corresponding singular vectors.

Proof. Let us denote $r = \min\{m, n\}$. Then by Von Neumann's trace inequality,

$$|\langle G, \delta W\rangle| \leq \sum_{i=1}^{r} \sigma_i(G)\sigma_i(\delta W) \Rightarrow \langle G, \delta W\rangle \geq -\sum_{i=1}^{r} \sigma_i(G)\sigma_i(\delta W)\,.$$

Thus, expressing nuclear norm through singular values, we can write down

$$f(\delta W) \geq -\sum_{i=1}^{r} \sigma_i(G)\sigma_i(\delta W) + \lambda\left(\sum_{i=1}^{r}\sigma_i(\delta W)\right)^2 \geq \min_{d_1,\ldots,d_r\geq 0} -\sum_{i=1}^{r}\sigma_i(G)d_i + \lambda\left(\sum_{i=1}^{r} d_i\right)^2 .$$

By Karush-Kuhn-Tucker theorem, necessary conditions of minimum are

$$\left(d_i \geq 0 \text{ and } -\sigma_i(G) + 2\lambda\sum_{j=1}^{r} d_j = 0\right) \text{ or } \left(d_i = 0 \text{ and } -\sigma_i(G) + 2\lambda\sum_{j=1}^{r} d_j \geq 0\right) \quad i = 1,\ldots,r\,.$$

These conditions simplify to

$$\sum_{i\in S} d_i = \sigma_1(G)\,,\ \begin{cases} d_i \geq 0 & \text{if } \sigma_i(G) = \sigma_1(G)\,, \\ d_i = 0 & \text{otherwise}\,. \end{cases}$$

All points satisfying those conditions deliver minimum, and

$$f(\delta W) \geq -\frac{\sigma_1^2(G)}{4\lambda}\,.$$

Now let

$$\delta W^* = -\frac{1}{2\lambda} u_1\sigma_1(G)v_1^T\,.$$

Inserting it to $f(\delta W)$ gives

$$f\left(\delta W^*\right) = -\frac{\sigma_1(G)^2}{2\lambda} + \frac{\sigma_1(G)^2}{4\lambda} = -\frac{\sigma_1(G)^2}{4\lambda}$$

This matches the derived lower bound. Thus, δW^* minimizes $f(\delta W)$. □

Lemma 2. *Let $G \in \mathbb{R}^{m\times n}$, $r = \min\{m.n\}$ and $\lambda > 0$. Then, the following optimization problem*

$$f(\delta W) = \langle G, \delta W\rangle + \lambda\|\delta W\|_{F*}^2 \to \min_{\delta W}\,,$$

where $\|\cdot\|_{F*}$ is defined in equation 16 has solution

$$\delta W = -\frac{1}{\lambda} U D V^T \tag{20}$$

with $D = diag(d_i)$, where $d_i = [\sigma_i - \tau]_+$, and τ is given by

$$\sum_{i=1}^{r}[\sigma_i - \tau]_+ = \tau\,. \tag{21}$$

Proof. Analogously to the proof of Lemma 1, we can use Von Neumann's trace inequality to write down

$$f(\delta W) \geq -\sum_{i=1}^{r}\sigma_i(G)\sigma_i(\delta W) + \frac{\lambda}{2}\left(\sum_{i=1}^{r}\sigma_i(\delta W)\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{r}\sigma_i^2(\delta W)\,,$$

$$f(\delta W) \geq \frac{1}{\lambda}\min_{d_1,\ldots,d_r\geq 0} -\sum_{i=1}^{r}\sigma_i(G)d_i + \frac{1}{2}\left(\sum_{i=1}^{r} d_i\right)^2 + \frac{1}{2}\left(\sum_{i=1}^{r} d_i^2\right)\,, \tag{22}$$

By Karush-Kuhn-Tucker theorem, neccesary conditions of minimum are

$$\left(d_i \geq 0 \text{ and } -\sigma_i(G) + \sum_{j=1}^{r} d_j + d_i = 0\right) \text{ or } \left(d_i = 0 \text{ and } -\sigma_i(G) + \sum_{j=1}^{r} d_j + d_i \geq 0\right) \quad i = 1,\ldots,r\,.$$

Denoting $\tau = \sum_{i=1}^{r} d_i$ gives $d_i = [\sigma_i(G) - \tau]_+$, where τ satisfies

$$\sum_{i=1}^{n} [\sigma_i(G) - \tau]_+ = \tau \,. \tag{23}$$

Inserting found minimum point into equation 22 yields

$$f(\delta W) \geq -\sum_{i=1}^{r} d_i(\tau + d_i) + \frac{\tau^2}{2\lambda} + \frac{1}{2\lambda}\sum_{i=1}^{r} d_i^2 = -\frac{1}{2\lambda}\left(\tau^2 + \sum_{i=1}^{r} d_i^2\right) .$$

Now let

$$\delta W^* = -\frac{1}{\lambda} U D V^T \tag{24}$$

with $D = \text{diag}(d_i)$. Inserting it to $f(\delta W)$ gives

$$f(\delta W^*) = -\sum_{i=1}^{r} d_i(\tau + d_i) + \frac{\tau^2}{2\lambda} + \frac{1}{2\lambda}\sum_{i=1}^{r} d_i^2 = -\frac{1}{2\lambda}\left(\tau^2 + \sum_{i=1}^{r} d_i^2\right) .$$

This matches the derived lower bound. Thus, δW^* minimizes $f(\delta W)$. □

12 QUALITY METRICS

1. The derivation is theoretically solid
2. The numerical procedure used to compute a step is grounded and has estimated time overhead (say, in FLOPS)
3. The code with Neon trains MLP and CNN (and NanoGPT, but it's a bonus) less than 3 times slower than Adam
4. Instruction of setting the parameters of the algorithm are presented and justified
5. The announced article has full structure (Abstract, Introduction, Theory, Experiments, Conclusion, Appendix)
6. If results are positive, it is written with NeurIPS template.

13 PRELIMINARY PLAN

Week April 28 - May 4

- For Alexey: solve how to tune the algorithms for MLP and CNN, try formulating theory (and an appropriate model of the problem) why Muon and Neon are so successful, and create the drafts of the proofs. Register at NeurIPS site.
- For Ivan: write the theory for an update from the algebra point of view (as for an article)
- For Nikolay: write the theory for computing an update, and implement the method, if required
- For Alexander: reproduce results of Jordan on NanoGPT and ResNet (CIFAR-10), learn to train both models with Neon.

Week May 5 - May 11

- For Alexey: finalize the proofs. Verify them via small experiments on MLP and CNN. Write with Alexander Experiments for the article.
- For Ivan: join Nikolay to finalize algebra part of the article. Estimate FLOPS, memory and other overheads (produce O bounds)
- For Nikolay: write a draft of the poster (before May 6), and work with Ivan
- For Alexander: agressively test algorithms, prove that Neon outperforms competitors and prepare the results for the article.

- For everybody: write and edit the article
- May 11: submit an abstract to NeurIPS.
- May 12-14: the article is being polished.
- May 15: the article must be sent.

14 PROTOTYPING PHASE REPORT

1. Update rule is derived, see idea
2. Update rule methods are tested: power iteration vs Lanczos (see 1)
3. Recorded the distribution of singular values of gradients during NanoGPT training (see Firgures 1, 1).

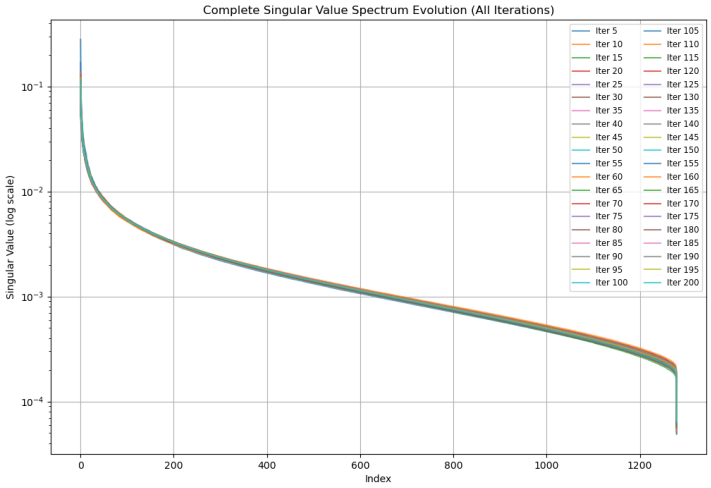

Figure 1: Singular values of 50257×1280 layer via 200 iterations

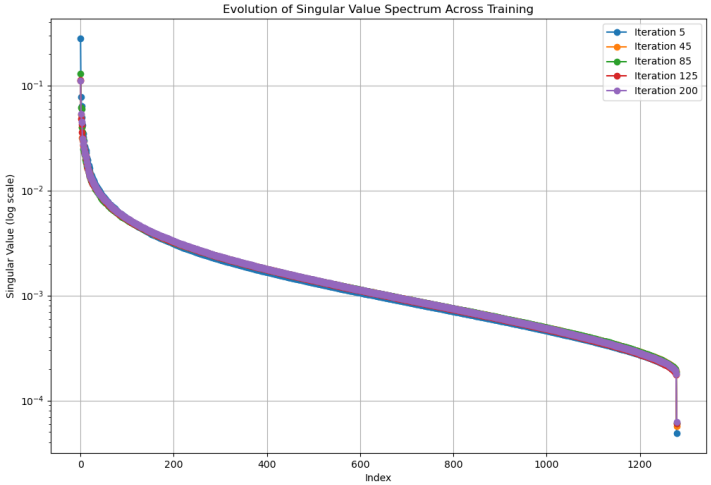

Figure 2: Singular values of of 50257×1280 layer for 5-th,45-th, 65-th,175-th and 200-th iteration

4. NanoGPT is tested on Muon and Adam. For now, Neon (rank-1 version) does not converge (see Figures 3 4)

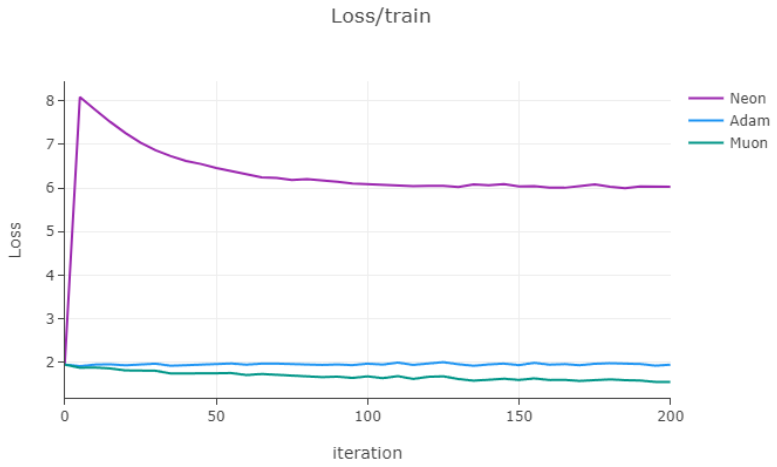

Figure 3: Train loss

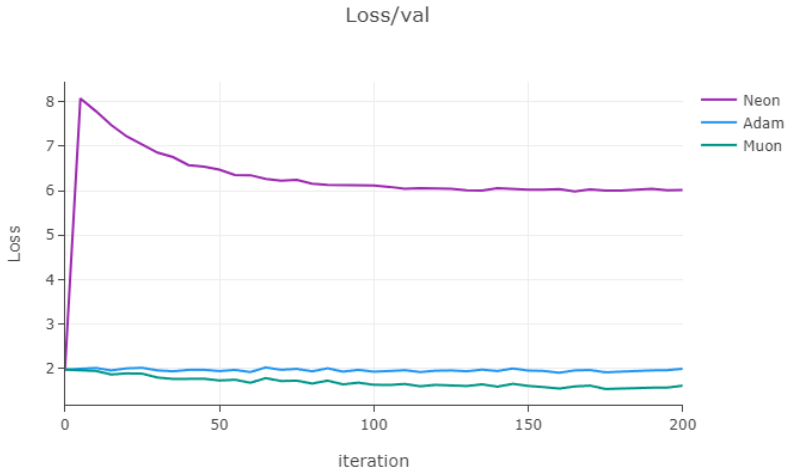

Figure 4: Validation loss

The pictures show the best results achieved so far. The experiments were conducted with two 4090 24GB GPUs for nanotgpt-large on the tiny stories dataset.

5. Neon (rank-1 version), Muon, AdamW and SGD are compared on MLP and CNN (see Figures 5, 6, 7, and 8). All methods work correctly, but again there is the problem with which one is the fastest (for now, it's SGD).

REFERENCES

Kwangjun Ahn and Byron Xu. Dion: A communication-efficient optimizer for large models, 2025. URL https://arxiv.org/abs/2504.05295.

Jeremy Bernstein. Deriving muon, 2025. URL https://jeremybernste.in/writing/deriving-muon.

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

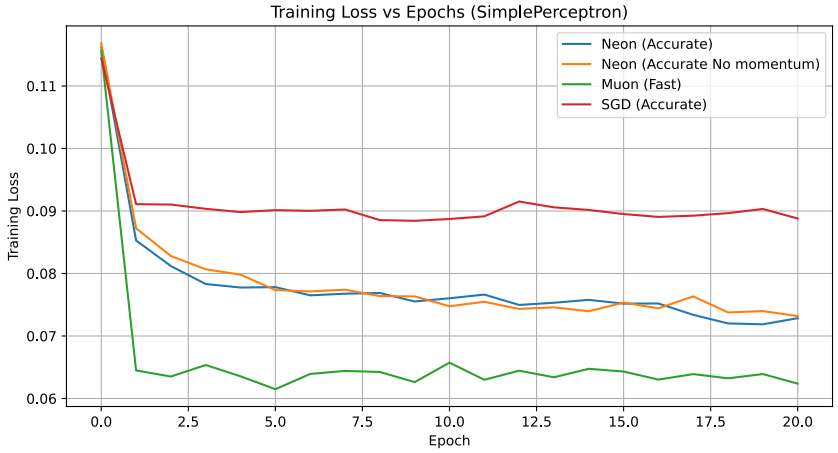

Figure 5: MLP: self.linear1 = nn.Linear(32*32*3, 512), self.linear2 = nn.Linear(512, 10), self.activ = nn.GELU()

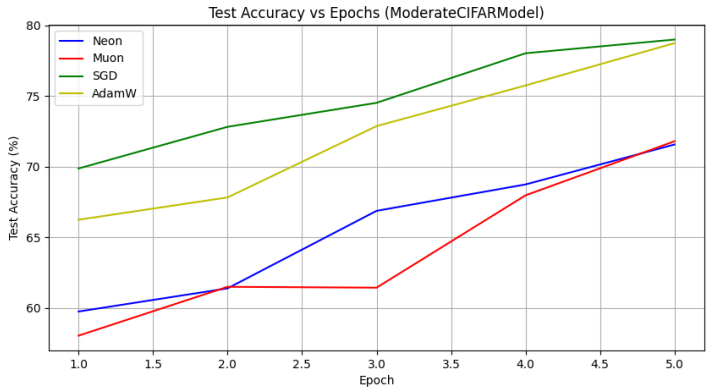

Figure 6: CNN: 2 convolutional blocks, 2 fully connected layers, activation + dropout

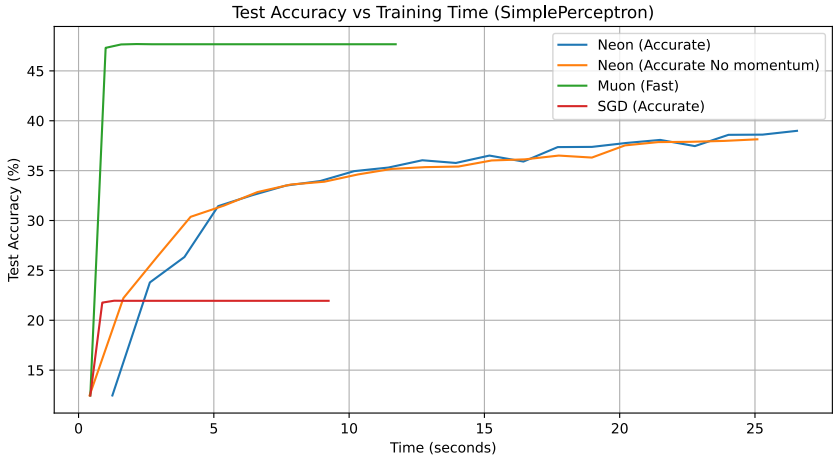

Figure 7: MLP: wallclock time measurements

Weizhu Chen, Chen Liang, Tuo Zhao, Zixuan Zhang, Hao Kang, Liming Liu, Zichong Li, and Zhenghao Xu. Cosmos: A hybrid adaptive optimizer for memory-efficient training of llms, 2025.

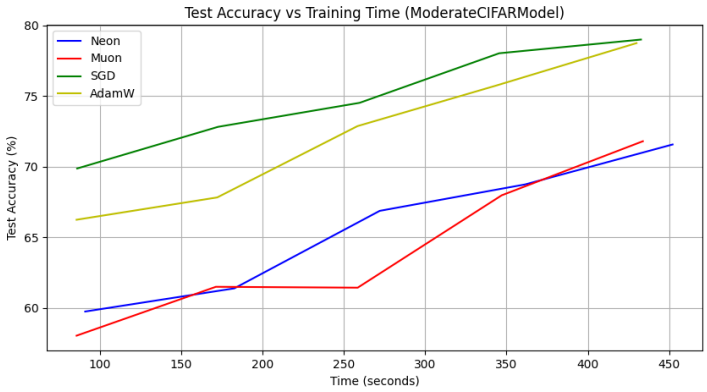

Figure 8: CNN: wallclock time measurements

Method	rtol	k	time,s
Power Iterations	0.01	1	7.7
SVDS (thick-restart Lanczos method)	0.01	1	0.18
PCA Low Rank (RSVD)	0.01	1	1.15
SVDS (thick-restart Lanczos method)	0.01	10	0.47
PCA Low Rank (RSVD)	0.01	10	19.4
SVDS (thick-restart Lanczos method)	0.01	100	1.96
PCA Low Rank (RSVD)	0.01	100	170

Table 1: k-rank updated comparison
Comparison of different numerical methods to calculate k-rank update on 5000×5000 matrix of real numbers, rtol is an error in Frobenius norm relative to the k-rank approximation of truncated svd. During the research it was noted that rsvd can give good and fast approximation for singular values, but the matrix of approximation is far from the one given by truncated svd, while Lanczos method gives good and fast approximation for a matrix, but not so good approximation for singular values.

URL https://arxiv.org/abs/2502.17410.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018. URL https://arxiv.org/abs/1802.09568.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Dmitry Kovalev. Understanding gradient orthogonalization for deep learning via non-euclidean trust-region optimization. *arXiv preprint arXiv:2503.12645*, 2025.

Jiaxiang Li and Mingyi Hong. A note on the convergence of muon and further, 2025. URL https://arxiv.org/abs/2502.02900.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *ArXiv*, abs/1711.05101, 2017.

Depen Morwani, Itai Shapira, Nikhil Vyas, eran malach, Sham M. Kakade, and Lucas Janson. A new perspective on shampoo's preconditioner. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=c6zI3Cp8c6.

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

Amund Tveit, Bjørn Remseth, and Arve Skogvold. Muon optimizer accelerates grokking, 2025. URL https://arxiv.org/abs/2504.16041.

A Old Appendix

You may include other additional sections here.