The Ky Fan Norms and Beyond: Dual Norms and Combinations for Matrix Optimization

Alexey Kravatskiy KRAVTSKII.AIU@PHYSTECH.EDU
MIPT

Ivan Kozyrev KOZYREV.IN@PHYSTECH.EDU
MIPT, INM RAS

Nikolai Kozlov KOZLOV.NA@PHYSTECH.EDU
MIPT

Alexander Vinogradov VINOGRADOV.AM@PHYSTECH.EDU
MIPT

Daniil Merkulov DANIIL.MERKULOV@PHYSTECH.EDU
MIPT, Skoltech, HSE, AI4Science

Ivan Oseledets I.OSELEDETS@SKOLTECH.RU
AIRI, Skoltech, INM RAS

Abstract

In this article, we explore the use of various matrix norms for optimizing functions of weight matrices, a crucial problem in training large language models. Moving beyond the spectral norm underlying the Muon update, we leverage duals of the Ky Fan k-norms to introduce a family of Muon-like algorithms we name *Fanions*, which are closely related to Dion. By working with duals of convex combinations of the Ky Fan k-norms with either the Frobenius norm or the l_∞ norm, we construct the families of *F-Fanions* and *S-Fanions*, respectively. Their most prominent members are *F-Muon* and *S-Muon*. We complement our theoretical analysis with an extensive empirical study of these algorithms across a wide range of tasks and settings, demonstrating that F-Muon and S-Muon consistently match Muon's performance, while outperforming vanilla Muon on a synthetic linear least squares problem.

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 (Kingma and Ba, 2014) and AdamW (Loshchilov and Hutter, 2017), recently proposed Muon (Jordan et al., 2024b) has shown promising results on training very large models (Liu et al., 2025). Its key difference from Adam and AdamW is that it has been constructed specifically for optimizing functions of weight matrices, which are common in deep learning.

From a theoretical perspective, a key innovation of Muon was its principled derivation of the update rule, which emerged as the solution to an optimization problem constrained by the RMS-to-RMS norm (a scaled version of the spectral norm) (Bernstein, 2025).

Motivated by the success of Muon, many generalizations and variations of it were proposed. Among the notable ones are Scion (Pethick et al., 2025c), Dion (Ahn et al., 2025) and Gluon (Riabinin et al., 2025). Those works try to explain Muon's efficiency and establish convergence bounds. One central question, however, remains unanswered:

In deriving Muon's update step, why should one constrain by the spectral or any other operator norm? How would alternative norms affect performance and computational cost?

In this article, we tackle this question by showing that there are many viable non-operator norms. We leverage the family of norms dual to Ky Fan k-norms to derive a new family of *Fanions*, algorithms with low-rank updates. This approach theoretically explains the backbone of Dion's update (Ahn et al., 2025) and generalizes the memory-motivated application of the nuclear norm to Sharpness-Aware Minimization (Pethick et al., 2025b). As was done with Muon, we develop an effective procedure for computing Fanions' updates. The Lanczos algorithm, which is described and compared with its competitors in Section 5, is the most operation-efficient algorithm, though it currently lacks an effective GPU- and PyTorch-friendly implementation.

Working with dual norms and various convex combinations of norms, we construct the families of *F-Fanions* and *S-Fanions*, which are hybrids of Muon with NormalizedSGD and SignSGD, respectively.

After that in Section 6 compare the performance of these algorithm families on various model and real-world problems:

- Synthetic least squares experiment
- CIFAR-10 airbench (Keller, 2023)
- Pre-training NanoGPT and GPT-2 Medium on FineWeb dataset (Jordan et al., 2024a)
- Fine-tuning NanoGPT on Tiny Stories dataset (Eldan and Li, 2023)

Our experiments reveal important insights into the role of matrix norms in optimization. First, using the example of Neon (the rank-one Fanion), we show that not every LMO-based algorithm is effective, despite sharing the same asymptotics in the general bounds of (Kovalev, 2025) and (Riabinin et al., 2025). This suggests that existing theoretical guarantees should be refined to better explain empirical performance.

Most notably, our experiments on real-world tasks demonstrate that the choice of underlying matrix norm is remarkably flexible. On CIFAR-10 airbench, properly-tuned F-Muon and S-Muon achieve $94.02 \pm 0.13\%$ and $94.03 \pm 0.13\%$ accuracy, matching Muon's $94.01 \pm 0.13\%$ performance. On pre-training of NanoGPT F-Muon achieves 3.281 cross-entropy loss, while fully-tuned Muon achieves 3.279; of GPT-2 Medium - F-Muon 2.9215 and Muon 2.9198. Finally, S-Muon matches Muon on fine-tuning NanoGPT on Tiny Stories, while F-Muon is far more resistant to learning rate choice than Muon. These results show that Muon-like algorithms can maintain competitive performance even when the underlying norm constraint is significantly modified, providing an affirmative answer to the central question posed above. Moreover, the tools from Section 4 give researchers considerable flexibility in designing algorithms that need not be direct modifications of Muon.

2. Preliminaries: Linear Minimization Oracle Framework

Training a neural network is essentially an optimization of a function of several weight matrices and a few vectors. Let us start by considering the problem of minimizing a differentiable function of a *single* matrix (the connection to the general case is presented in Section A):

$$F(\cdot)\colon \mathbb{R}^{m\times n} \to \mathbb{R}\,, \qquad F(\boldsymbol{X}) \to \min_{\boldsymbol{X}\in\mathbb{R}^{m\times n}} \tag{1}$$

We equip the matrix space $\mathbb{R}^{m\times n}$ with a standard dot product $\langle\cdot,\cdot\rangle \to \mathbb{R}$ and a norm $\|\cdot\|\colon \mathbb{R}^{m\times n} \to \mathbb{R}_+$, which does not have to coincide with the Frobenius norm $\|\cdot\|_{\mathrm{F}} = \langle\cdot,\cdot\rangle$. The dual norm $\|\cdot\|^\dagger\colon \mathbb{R}^{m\times n} \to \mathbb{R}_+$ that is associated with $\|\cdot\|$ is defined as

$$\|\boldsymbol{M}\|^\dagger = \sup_{\boldsymbol{D}\in\mathbb{R}^{m\times n}\colon \|\boldsymbol{D}\|\le 1} \langle \boldsymbol{M}, \boldsymbol{D}\rangle\,. \tag{2}$$

Such problems can be solved with an iterative algorithm based on the Linear Minimization Oracle (LMO):

$$\mathrm{LMO}(\boldsymbol{M}^t) \in \arg\min_{\boldsymbol{D}\in\mathcal{S}} \langle \boldsymbol{M}^t, \boldsymbol{D}\rangle\,, \tag{3}$$

where $\boldsymbol{M}^t$ is the effective update direction and $\mathcal{S}\subset\mathbb{R}^{m\times n}$ is a constraint set. The algorithm proceeds as:

$$\begin{aligned} \boldsymbol{B}^t &= \beta\boldsymbol{B}^{t-1} + (1-\beta)\nabla F(\boldsymbol{X}^t) \quad \text{(momentum buffer)},\\ \boldsymbol{M}^t &= \begin{cases} \nabla F(\boldsymbol{X}^t) & \text{(no momentum)},\\ \boldsymbol{B}^t & \text{(heavy ball)},\\ \nabla F(\boldsymbol{X}^t) + \beta\boldsymbol{B}^t & \text{(approximate Nesterov)}, \end{cases}\\ \boldsymbol{X}^{t+1} &= \boldsymbol{X}^t + \gamma_t \mathrm{LMO}\left(\boldsymbol{M}^t\right)\,, \end{aligned} \tag{4}$$

where $\beta\in[0,1)$ is the momentum coefficient. Throughout our experiments, we employ approximate Nesterov momentum, which matches the implementation in Muon and PyTorch SGD (`nesterov=True`). It uses the current gradient rather than the true lookahead gradient, trading theoretical guarantees for computational efficiency.

We are particularly interested in the case when $\mathcal{S}$ is a unit ball in some norm $\|\cdot\|$:

$$\mathcal{S} = \mathcal{B}_{\|\cdot\|} = \{\boldsymbol{D}\in\mathbb{R}^{m\times n} \mid \|\boldsymbol{D}\|\le 1\}\,.$$

In this case,

$$\arg\min_{\boldsymbol{D}\in\mathcal{S}} \langle \boldsymbol{M}^t, \boldsymbol{D}\rangle = -\{\boldsymbol{D}\in\mathcal{B}_{\|\cdot\|} \mid \langle \boldsymbol{M}^t, \boldsymbol{D}\rangle = \|\boldsymbol{M}^t\|^\dagger\}\,,$$

and the update for $\boldsymbol{X}^{t+1}$ in Equation (4) simplifies to

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t\{\boldsymbol{D}\in\mathcal{B}_{\|\cdot\|} \mid \langle \boldsymbol{M}^t, \boldsymbol{D}\rangle = \|\boldsymbol{M}^t\|^\dagger\}\,. \tag{5}$$

Using this formula, it is easy to compute updates for algorithms induced by various norms $\|\cdot\|$:

Frobenius norm and Normalized SGD When the norm $\|\cdot\|$ is the Frobenius norm $\|\cdot\|_{\mathrm{F}}$, Equation (5) turns into

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t \frac{\boldsymbol{M}^t}{\|\boldsymbol{M}^t\|_{\mathrm{F}}}\,, \tag{6}$$

which recovers Normalized SGD (NSGD).

Spectral norm and Muon When the norm is the spectral norm $\|\cdot\|_2$, we get

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t \boldsymbol{U}\boldsymbol{V}^\top\,, \tag{7}$$

which is Muon without the $\sqrt{m/n}$ factor. Here, $\boldsymbol{M}^t = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ is the Singular Value Decomposition (SVD) of $\boldsymbol{M}^t$ ($\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_r]$, $\boldsymbol{\Sigma} = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$, and $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r]$). Muon can be recovered by taking the RMS-to-RMS norm: $\sqrt{n/m}\|\cdot\|_2$.

Chebyshev norm and SignSGD When the norm is the Chebyshev norm $\|\cdot\|_{\mathrm{C}}$, we get

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t \operatorname{sign}(\boldsymbol{M}^t)\,, \tag{8}$$

which recovers SignSGD (Bernstein et al., 2018). Here, $\operatorname{sign}(\boldsymbol{M}^t)$ denotes the element-wise sign function. SignSGD is particularly notable for its communication efficiency in distributed training, as it compresses gradients to 1-bit per parameter.

3. Beyond the Spectral Norm: Fanions

3.1. The Rank Gap Problem

Having examined the Frobenius norm (NSGD), spectral norm (Muon), and Chebyshev norm (SignSGD), all of which produce full-rank updates, we turn to the nuclear norm $\|\cdot\|_*$.

Lemma 1 *When $\|\cdot\| = \|\cdot\|_*$, Equation (5) becomes*

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t \boldsymbol{u}_1 \boldsymbol{v}_1^\top\,. \tag{9}$$

Proof Since $\|\cdot\|_*^\dagger = \|\cdot\|_2$, the goal is to reach $\langle \boldsymbol{M}^t, \boldsymbol{D}\rangle = \sigma_1$ in Equation (5). Note that $\boldsymbol{D} = \boldsymbol{u}_1\boldsymbol{v}_1^\top$ delivers this value. Indeed, $\|\boldsymbol{D}\|_* = 1$ and by the trace property and orthogonality of the singular vectors,

$$\langle \boldsymbol{M}^t, \boldsymbol{D}\rangle = \langle \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \boldsymbol{u}_1\boldsymbol{v}_1^\top\rangle = \operatorname{tr}\operatorname{diag}(\sigma_1, 0, \ldots, 0) = \|\boldsymbol{M}^t\|_2\,,$$

which completes the proof. ■

We call this algorithm *Neon*. The nuclear norm thus yields rank-one updates, in stark contrast to the full-rank updates of Muon, NSGD, and SignSGD. This raises a natural question: can we derive algorithms with updates of intermediate ranks?

Schatten norms (Cesista, 2025) considered Schatten-p norms:

$$\|\boldsymbol{M}^t\|_{S_p} = \left(\sum_{i=1}^{\min(m,n)} \sigma_i^p\right)^{1/p},$$

which produce the update

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t \boldsymbol{U} \frac{\operatorname{diag}\left(\sigma_1^{q-1}, \ldots, \sigma_{\min(m,n)}^{q-1}\right)}{\left(\sum_{i=1}^{\min(m,n)} \sigma_i^q\right)^{\frac{q-1}{q}}} \boldsymbol{V}^\top$$

where p and q satisfy $p^{-1} + q^{-1} = 1$. This formula recovers Neon when $p \to 1$ (provided $\sigma_1 > \sigma_2$, which holds on real data), NSGD when $p = 2$, and Muon when $p \to \infty$.

However, Schatten norms do not fill the rank gap: when $p > 1$, the update has full rank, while when $p = 1$, it has rank one. Moreover, computing the update for $p \neq 1, 2, \infty$ appears to require knowing all σ_i, making the problem as hard as computing the full SVD.

Ky Fan norms Another family of matrix norms offers a potential solution: the Ky Fan norms. For $k \in \{1, \ldots, \min(m,n)\}$, the Ky Fan k-norm $\|\cdot\|_{\text{KF-}k}$ equals $\sum_{i=1}^k \sigma_i$, the sum of the k largest singular values. Notable special cases include the Ky Fan 1-norm (the spectral norm) and the Ky Fan $\min\{m,n\}$-norm (the nuclear norm).

To derive the update formula for arbitrary k, we use the expression for the norm dual to the Ky Fan k-norm (see, for instance (Bhatia, 2013), p. 96):

$$\|\cdot\|_{\text{KF-}k}^\dagger = \max\left\{\frac{1}{k}\|\cdot\|_*, \|\cdot\|_2\right\}.$$

According to Equation (5), we need to find $\boldsymbol{D}$ such that $\|\boldsymbol{D}\|_{\text{KF-}k} = 1$ and

$$\langle \boldsymbol{M}^t, \boldsymbol{D}\rangle = \max\left\{\frac{1}{k}\sum_{i=1}^{\min(m,n)} \sigma_i,\ \sigma_1\right\}.$$

The Neon update $\boldsymbol{D} = \boldsymbol{u}_1\boldsymbol{v}_1^\top$ achieves σ_1, while the scaled Muon update $\boldsymbol{D} = \frac{1}{k}\boldsymbol{U}\boldsymbol{V}^\top$ achieves $\frac{1}{k}\sum_{i=1}^{\min(m,n)} \sigma_i$. Thus, the update is either

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t \boldsymbol{u}_1\boldsymbol{v}_1^\top \text{ or } \boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \frac{\gamma_t}{k}\boldsymbol{U}\boldsymbol{V}^\top,$$

depending on which of those two expressions achieves smaller residual on the linear function $L(\boldsymbol{X}) := F(\boldsymbol{X}^t) + \langle \boldsymbol{M}^t, \boldsymbol{X} - \boldsymbol{X}^t\rangle$. The Ky Fan norms thus fail to close the rank gap: the resulting update is either rank-one or full-rank.

3.2. Solution: Duals to Ky Fan Norms

Unlike Schatten norms, which satisfy $\|\boldsymbol{M}^t\|_{S_p}^\dagger = \|\boldsymbol{M}^t\|_{S_q}$ (for $p^{-1} + q^{-1} = 1$) and are thus closed under dualization, Ky Fan norms are generally not. Only two exceptional cases exist: the dual to the Ky Fan 1-norm $\|\cdot\|_{\text{KF-1}}$ (the spectral norm) is the Ky Fan $\min(m,n)$-norm $\|\cdot\|_{\text{KF-}\min(m,n)}$ (the nuclear norm), and vice versa. Thus working with duals of Ky Fan norms can open up new possibilities:

Lemma 2 *When* $\|\cdot\| = \|\boldsymbol{M}^t\|^\dagger_{KF\text{-}k}$*, Equation (5) turns into:*

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t \sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{v}_i^\top . \tag{10}$$

Proof Since $\|\cdot\|^\dagger = \|\cdot\|^{\dagger\dagger}_{\text{KF-}k} = \|\cdot\|_{\text{KF-}k}$, the goal is to reach $\langle \boldsymbol{M}^t, \boldsymbol{D}\rangle = \|\boldsymbol{M}^t\|_{\text{KF-}k}$ in Equation (5). Note that $\boldsymbol{D} = \sum_{i=1}^k \boldsymbol{u}_i \boldsymbol{v}_i^\top$ attains this value. Indeed,

$$\langle \boldsymbol{M}^t, \boldsymbol{D}\rangle = \langle \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{v}_i^\top\rangle = \sum_{i,j=1}^{r,k} \langle \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i^\top, \boldsymbol{u}_j \boldsymbol{v}_j^\top\rangle = \sum_{i=1}^{k} \sigma_i = \|\boldsymbol{M}^t\|_{\text{KF-}k}\,,$$

which completes the proof. ■

These updates define the *Fanion* family of LMO-based algorithms, each operating under a norm $\|\boldsymbol{M}^t\|^\dagger_{\text{KF-}k}$. We denote the algorithm for a particular k as *Fanion-k*. This family elegantly bridges the rank gap, providing updates of any intermediate rank k.

Connection to existing algorithms The Fanion family unifies several known algorithms:

- **Neon** is Fanion-1 (rank-one updates)
- **Muon** is Fanion-min$\{m, n\}$ (full-rank updates)
- **Dion** (unsharded): The rank-r Dion (Algorithm 1 from (Ahn et al., 2025)) without error feedback and without scaling of the update is actually Fanion-r (see (Pethick, 2025), where Dion is written in a notation more similar to ours)

In Section 5, we will discuss how to efficiently compute Fanion updates using the Lanczos algorithm.

4. Conic Combination of LMO-algorithms is an LMO-algorithm

Simply applying norms dual to Ky Fan k-norms, however, does not provide sufficient algorithmic diversity for our purposes. We now consider linear combinations of LMO-based algorithms and show that these combinations are themselves LMO-algorithms.

4.1. General Case

We begin with a well-known result on dual norms (see, e.g., (Yu, 2012, Table 1)).

Lemma 3 *Let* $\|\cdot\|_{(1)}, \ldots, \|\cdot\|_{(n)}$ *be norms on a finite-dimensional Euclidean space, and let* $\alpha_1, \ldots, \alpha_n$ *be non-negative reals. Define*

$$\|\cdot\| := \sum_{i=1}^{n} \alpha_i \|\cdot\|_{(i)}.$$

Then the dual unit ball of $\|\cdot\|$ *satisfies*

$$\mathcal{B}_{\|\cdot\|^\dagger} = \sum_{i=1}^{n} \alpha_i \mathcal{B}_{\|\cdot\|_{(i)}^\dagger}$$

where $\sum$ *denotes the Minkowski sum and* $\mathcal{B}_{\|\cdot\|_{(i)}^\dagger}$ *is the unit ball of the dual norm* $\|\cdot\|_{(i)}^\dagger$.

A proof is provided in the appendix (see Section B).

Lemma 4 *Let* $\|\cdot\|_{(1)}, \ldots, \|\cdot\|_{(n)}$ *be norms on a finite-dimensional Euclidean space, and let* $\alpha_1, \ldots, \alpha_n$ *be non-negative reals. Consider Linear Minimization Oracles* $\mathrm{LMO}_1, \ldots, \mathrm{LMO}_n$, *corresponding to the unit balls of these norms. Then,* $\sum_{i=1}^n \alpha_i \mathrm{LMO}_i$ *is the LMO corresponding to the norm* $\|\cdot\|$ *dual to the* $\sum_{i=1}^n \alpha_i \|\cdot\|_{(i)}^\dagger$.

Proof Using Lemma 3 and the biduality property $\|\cdot\|^{\dagger\dagger} = \|\cdot\|$, we obtain the unit ball representation: $\mathcal{B}_{\|\cdot\|} = \sum_{i=1}^n \alpha_i \mathcal{B}_{\|\cdot\|_{(i)}}$. The linear minimization problem over this ball can thus be transformed as follows:

$$\operatorname*{arg\,min}_{\boldsymbol{D} \in \mathcal{B}_{\|\cdot\|}} \langle \boldsymbol{M}, \boldsymbol{D} \rangle = \operatorname*{arg\,min}_{\boldsymbol{D}_1 \in \alpha_i \mathcal{B}_{\|\cdot\|_{(1)}}, \ldots, \boldsymbol{D}_n \in \alpha_n \mathcal{B}_{\|\cdot\|_{(n)}}} \langle \boldsymbol{M}, \sum_{i=1}^n \boldsymbol{D}_i \rangle = \sum_{i=1}^n \operatorname*{arg\,min}_{D_i \in \mathcal{B}_{\|\cdot\|_{(i)}}} \langle \boldsymbol{M}, \boldsymbol{D}_i \rangle ,$$

where the last summation denotes the Minkowski sum. This immediately implies

$$\sum_{i=1}^n \alpha_i \mathrm{LMO}_i \in \operatorname*{arg\,min}_{\boldsymbol{D} \in \mathcal{B}_{\|\cdot\|}} \langle \boldsymbol{M}, \boldsymbol{D} \rangle ,$$

completing the proof. ■

Applying this result to optimization algorithms yields the following corollary.

Corollary 5 *Let there be a finite family of LMO based algorithms indexed by* $i = 1, \ldots, n$, *where the* $\boldsymbol{X}^{t+1}$ *update in the* i*-th algorithm is defined by*

$$\boldsymbol{X}^{t+1} - \boldsymbol{X}^t = \gamma_t \mathrm{LMO}_i(\boldsymbol{M}^t) ,$$

and LMO_i *corresponds to the unit ball of norm* $\|\cdot\|_i$. *For arbitrary non-negative* $\alpha_1, \ldots, \alpha_n$, *the algorithm with the update given by*

$$\boldsymbol{X}^{t+1} - \boldsymbol{X}^t = \gamma_t \sum_{i=1}^n \alpha_i \mathrm{LMO}_{\|\cdot\|_{(i)}}(\boldsymbol{M}^t)$$

is an LMO-algorithm itself, with LMO corresponding to the unit ball of the norm $\|\cdot\|$ *dual to the norm given by* $\sum_{i=1}^n \alpha_i \|\cdot\|_{(i)}^\dagger$.

4.2. Frobeniusize Them: F-Muon and F-Neon

We now construct concrete examples of algorithms obtained via linear combinations of LMO-algorithms. By Corollary 5, these combinations are themselves LMO-algorithms.

Combining Fanion-k with NSGD yields a family of algorithms with updates

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t \left(\alpha \sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{v}_i^\top + (1-\alpha) \frac{\boldsymbol{M}^t}{\|\boldsymbol{M}^t\|_{\mathrm{F}}} \right) . \tag{11}$$

Recall that Fanion-k operates under the dual to the Ky Fan k-norm, while NSGD operates under the self-dual Frobenius norm. By Corollary 5, this combination defines an LMO-algorithm with norm $\|\cdot\|^\dagger_{\mathrm{F-KF-k}}$, where

$$\|\cdot\|_{\mathrm{F-KF-k}} = \alpha \|\cdot\|_{\text{KF-}k} + (1-\alpha)\|\cdot\|_{\mathrm{F}} . \tag{12}$$

We call this family *F-Fanions*.

The extremal members of this family are *F-Neon* (with $k = 1$) and *F-Muon* (with $k = \min\{m, n\}$). Additional information and visualizations for the $\|\cdot\|_{\mathrm{F}*} = \|\cdot\|_{\mathrm{F-KF-1}}$ norm appear in the appendix (see Equation (16), Figure 8(a)subfigure).

4.3. Add SignSGD: S-Muon and S-Neon

Similarly, combining Fanion-k with SignSGD yields *S-Fanion-k* with update

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t - \gamma_t \left(\alpha \sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{v}_i^\top + (1-\alpha)\eta \operatorname{sign}(\boldsymbol{M}^t) \right) , \tag{13}$$

where `sign_lr_coeff` η is a scaling coefficient specific to SignSGD.

By Corollary 5, this defines an LMO-algorithm with norm $\|\cdot\|^\dagger_{\mathrm{C-KF-k}}$, where

$$\|\cdot\|_{\mathrm{C-KF-k}} = \alpha \|\cdot\|_{\text{KF-}k} + \frac{1-\alpha}{\eta} \|\cdot\|^\dagger_{\mathrm{C}} . \tag{14}$$

The extremal members of this family are *S-Neon* (with $k = 1$) and *S-Muon* (with $k = \min\{m, n\}$).

5. Computing the Updates

We employ the thick-restart Lanczos method for the symmetric eigenvalue problem (TRLan) to compute the low-rank updates of Fanions. We apply TRLan to either $\boldsymbol{M}^{t\top}\boldsymbol{M}^t$ or $\boldsymbol{M}^t\boldsymbol{M}^{t\top}$, selecting whichever matrix is smaller. We use the CuPy implementation of `cupy.sparse.linalg.svds` (Preferred Infrastructure and Developers, 2025), which internally relies on TRLan (Simonz, 1998).

TRLan is specifically designed for efficiently approximating the largest singular values and corresponding singular vectors of very large matrices. The thick-restart strategy retains the most informative Ritz vectors across restarts, which dramatically accelerates convergence while maintaining moderate memory consumption. TRLan is particularly well-suited to our GPU setting because its dominant computational cost consists of a modest number of

highly parallelizable matrix-vector multiplications (matvecs), and it avoids full reorthogonalization against the entire Krylov basis by employing short recurrence relations combined with thick restarting.

The per-cycle complexity is $\mathcal{O}(mn^2 + n^2k + nk^2)$, where $m \geq n$ are the dimensions of the target matrix and k is the size of the retained subspace (typically $k \ll n$).

In Table 1, we compare TRLan against randomized SVD (RSVD) and simple power iterations for computing the rank-k update used in Fanion-k and related algorithms. Experiments are performed on dense random matrices with i.i.d. $\mathcal{N}(0,1)$ entries using CPU implementations for fair comparison. We report:

- err_1: relative error in the Frobenius norm of $\sum_i^k \boldsymbol{u}_i \boldsymbol{v}_i^T$,
- err_2: relative error in the Frobenius norm of $\sum_i^k \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T$.

On 500×500 matrices, TRLan and RSVD require comparable wall-clock time, but TRLan delivers orders-of-magnitude lower error with far fewer matvecs. On larger 5000×5000 matrices, this advantage becomes even more pronounced: TRLan is 3-4 times faster than RSVD while using $\sim$30 times fewer matvecs at comparable or superior accuracy.

An interesting empirical observation is that RSVD tends to approximate the *singular values* themselves reasonably well, but the reconstructed low-rank matrix exhibits noticeable deviation from the truncated SVD. In contrast, TRLan provides an excellent approximation to the truncated SVD matrix itself (low err_2), albeit at the cost of occasionally less accurate individual singular values. This makes TRLan the preferred choice for algorithms like Neon/Fanion-k that only require the low-rank term $\sum \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$, but less suitable for methods (e.g., Dion) that explicitly require accurate σ_i for error feedback or step-size control.

A current practical limitation is the absence of a native PyTorch implementation of thick-restart Lanczos; existing PyTorch-based randomized SVD routines cannot match TRLan's accuracy-efficiency combination for the matrix reconstruction task.

For reference, Table 2 presents results for the Newton-Schulz polar decomposition iteration on the same matrices, where err_1 is the relative error of $\boldsymbol{U}\boldsymbol{V}^T$ (29-30 iterations to converge, resulting in a significantly higher matvec count than TRLan).

6. Experiments

6.1. Linear Least Squares

We begin by evaluating F-Fanions on the following convex L-smooth problem:

$$F(\boldsymbol{X}) = \frac{1}{2}\langle(\boldsymbol{X} - \boldsymbol{S}), \boldsymbol{M}(\boldsymbol{X} - \boldsymbol{S})\boldsymbol{N}\rangle \to \min_{\boldsymbol{X} \in \mathbb{R}^{m \times n}} \tag{15}$$

where $\boldsymbol{X} \in \mathbb{R}^{m \times n}$, $m = 500$ and $n = 500$ represent typical dimensions of a neural network weight matrix, $\boldsymbol{S} \in \mathbb{R}^{m \times n}$, and $\boldsymbol{M} \in \mathbb{S}_+^m$ and $\boldsymbol{N} \in \mathbb{S}_+^n$ are positive-semidefinite matrices. The spectra of $\boldsymbol{M}$ and $\boldsymbol{N}$ are uniformly distributed on the interval $(0, 1)$. We set $\boldsymbol{S} = 0$ and initialize $\boldsymbol{X}^0$ with entries drawn independently from $\mathcal{N}(0, 0.01)$.

We evaluate several algorithms from the Fanion family and their convex combinations, using the update rule from Equation (4) with approximate Nesterov momentum:

Table 1: Comparison of methods for computing rank-k updates on dense random matrices (CPU, double precision). Lower is better in all columns.

Matrix sizes	k	Method	Time (s)	Matvecs	Iterations	err_1	err_2
500×500	5	Power Iterations	0.062	2005	200	$9.2{\cdot}10^{-3}$	$9.1{\cdot}10^{-3}$
500×500	5	RSVD	0.017	1170	38	$9.8{\cdot}10^{-3}$	$9.6{\cdot}10^{-3}$
500×500	5	TRLan	0.018	131	65	$9.6{\cdot}10^{-5}$	$9.4{\cdot}10^{-5}$
500×500	50	Power Iterations	0.44	43750	437	$9.9{\cdot}10^{-3}$	$9.0{\cdot}10^{-3}$
500×500	50	RSVD	0.61	6120	50	$9.9{\cdot}10^{-3}$	$9.1{\cdot}10^{-3}$
500×500	50	TRLan	0.16	462	231	$3.3{\cdot}10^{-7}$	$3.0{\cdot}10^{-7}$
5000×5000	5	Power Iterations	9.6	9065	906	$8.6{\cdot}10^{-3}$	$8.6{\cdot}10^{-3}$
5000×5000	5	RSVD	2.1	5640	187	$9.7{\cdot}10^{-3}$	$9.7{\cdot}10^{-3}$
5000×5000	5	TRLan	0.70	205	102	$7.7{\cdot}10^{-3}$	$7.7{\cdot}10^{-3}$

Table 2: Newton-Schulz iterations on the same matrices (for reference).

Matrix size	Time (s)	Matvecs	Iterations	err_1
500×500	0.041	27 000	27	4.8e-3
5000×5000	26.4	290 000	29	6.5e-3

- **Fanions:** Neon (Fanion-1), Fanion-2, Fanion-10, Fanion-100, and Muon (Fanion-500).
- **F-Fanions:** F-Neon (F-Fanion-1), F-Fanion-2, F-Fanion-10, F-Fanion-100, and F-Muon (F-Fanion-500) with $\alpha = 1/2$.
- **S-Fanions:** S-Neon (S-Fanion-1), S-Fanion-2, S-Fanion-10, S-Fanion-100, and S-Muon (S-Fanion-500) with $\alpha = 1/2$ and `sign_lr_coeff=0.01`.

We also include baselines Normalized SGD and SignSGD, which correspond to F-Fanion and S-Fanion respectively with $\alpha = 0$ and arbitrary k.

Since theoretical bounds (Kovalev, 2025; Riabinin et al., 2025) rely on a loose norm bound $\|\cdot\| \leq \rho\|\cdot\|_{\mathrm{F}}$, we do not derive the learning rate or Nesterov momentum from the smoothness constants, which also depend on the choice of norm. Instead, we identify the `(lr, momentum)` pair that reaches the loss threshold of 0.001 in the minimal number of iterations. This setting is both realistic and consistent with the corollaries of convergence theorems that propose constant learning rate and momentum coefficients.

We perform a grid search over the following hyperparameter ranges:

- `momentum` $\in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95\}$ for all algorithms.
- `lr` $\in [0.005, 0.020]$ with step 0.001 for Muon, F-Muon, and S-Muon.
- `lr` $\in$ [5e−5, 2e−4] with step 1e−5.
- `lr` $\in [0.01, 0.10]$ with step 0.001 for NSGD.

The tuned parameters and the number of iterations required to converge to 0.001 are presented in Table 3. For intermediate-rank Fanions, F-Fanions, and S-Fanions, the hyperparameters `lr`, `momentum`, and `sign_lr_coeff` are set equal to those of Muon, F-Muon, and S-Muon respectively, as their loss decreases too slowly to reach 0.001 within a reasonable tuning budget.

Table 3: Tuning learning rates and momentum coefficients

Algorithm	`lr`	`momentum`	Iterations to 0.001 loss
Muon	0.007	0.5	1060
NSGD	0.08	0.95	1020
F-Muon	0.015	0.7	910
SignSGD	0.016×0.01	0.95	2650
S-Muon	0.011	0.9	890

The results are presented in Figure 1 with additional details in Section E.

Both F-Muon and S-Muon converge faster to lower loss values and achieve lower Frobenius norms of the gradient than NSGD, Muon, or SignSGD.

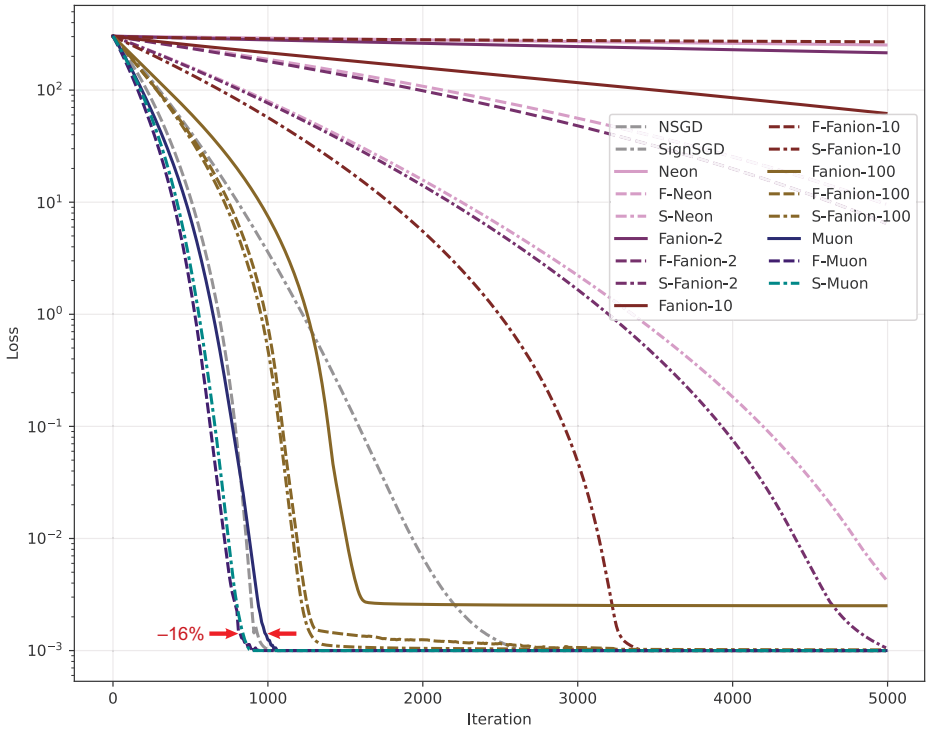
6.2. CIFAR-10 airbench

We evaluate the algorithms on the CIFAR-10 airbench (Keller, 2023). To assess the impact of the mixing parameter α, we first run F-Muon for different values of α using hyperparameters tuned for vanilla Muon by Keller Jordan: `lr=0.24(1 - step/total_steps), momentum=0.65, nesterov=True` with weight normalization. We perform 10 repetitions for each α value and record the accuracy after 8 epochs of training (Figure 2(a)subfigure).

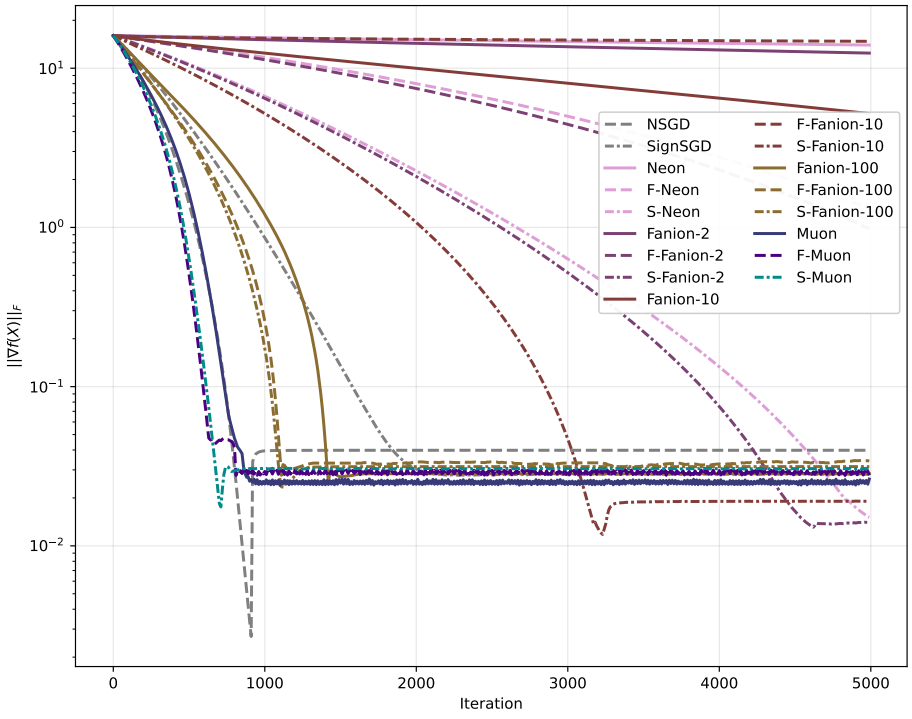
Next, we tune F-Muon specifically with $\alpha = 0.5$, obtaining optimal hyperparameters `lr=0.4(1 - step/total_steps)`, `momentum=0.6`, `nesterov = True`. Tuned F-Muon achieves $94.02 \pm 0.13\%$ validation accuracy after 8 epochs (averaged over 200 runs), matching Muon's accuracy and variance. We again measure the validation accuracy as a function of α (Figure 2(b)subfigure) and observe that even at $\alpha = 0.1$, the accuracy substantially exceeds that of vanilla NSGD.

For S-Muon with $\alpha = 0.5$, we tune all hyperparameters to find the optimal configuration `lr=0.42(1 - step/total_steps)`, `momentum=0.63`, `nesterov = True`, `sign_lr_coeff = 0.003`, achieving $94.03 \pm 0.13\%$ validation accuracy, which slightly exceeds vanilla Muon's performance.

The Muon-level performance is remarkable given that F-Muon and S-Muon operate with substantially different constraint geometries. Figure 3 visualizes F-Muon by plotting the LMO balls of Muon and F-Muon (scaled by actual learning rates) in the 2D space of singular values. S-Muon cannot be visualized in this manner because the l_∞ norm is not a function of singular values alone, and visualizing $\|\cdot\|_2$ for $\mathbb{R}^{m\times n}$ with $m, n > 2$ requires at least 4D space. Nevertheless, the LMO ball of S-Muon clearly differs substantially from Muon's, as the effective SignSGD learning rate $\texttt{lr}(1-\alpha)\texttt{sign\_lr\_coeff} = 6.3 \times 10^{-4}$ is comparable to typical SignSGD learning rates in deep learning.

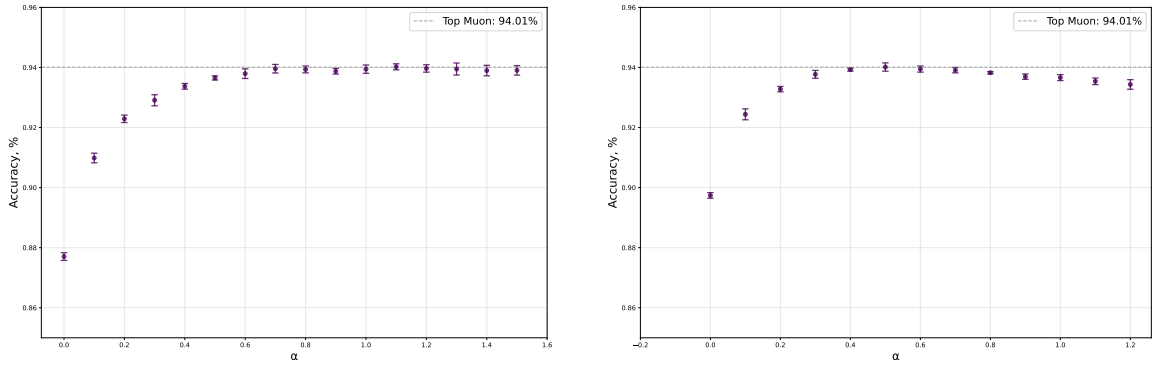


(*a*) The loss



(*b*) The Frobenius norm of the gradient

Figure 1: Linear least squares problem for a 500x500 matrix



(*a*) With parameters tuned for Muon (*b*) With parameters tuned for F-Muon

Figure 2: Mean validation accuracies for F-Muon with different α

Notably, even the pathological case $\alpha > 1$—which corresponds to an LMO constraint region that is not a norm ball—achieves accuracy nearly identical to vanilla Muon.

These observations raise fundamental questions about the sensitivity of LMO-based algorithms to the shape of the constraint region.

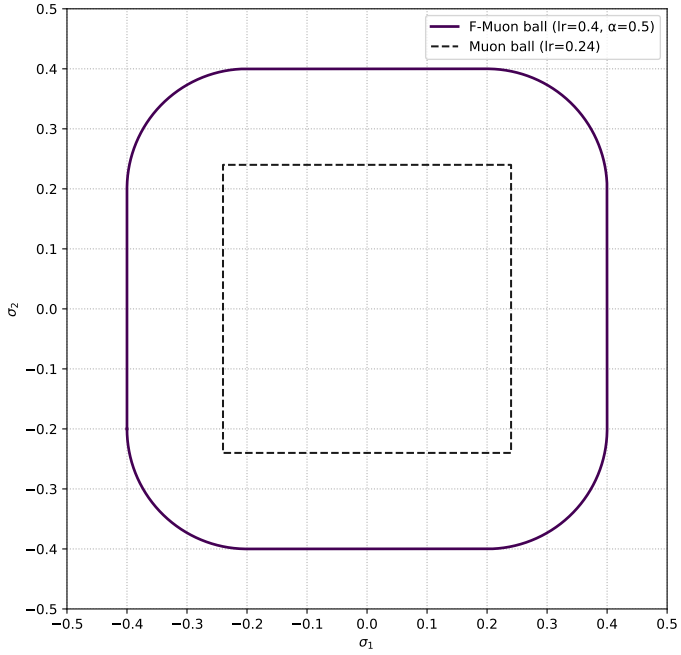


Figure 3: Visualization of the LMO balls for Muon and F-Muon for CNN training.

6.3. NanoGPT Speedrun

We evaluate F-Muon and S-Muon with $\alpha = 0.5$ against Muon, SignSGD, and NSGD on the NanoGPT speedrun benchmark (Jordan et al., 2024a). The optimal hyperparameters are shown in the legend of Figure 4, with `sign_sgd_coeff` values of 3e-4/0.07 for S-Muon. After 1750 training steps, F-Muon achieves a cross-entropy loss of 3.281, S-Muon achieves 3.287, and Muon achieves 3.279, falling below the target threshold of 3.280. As illustrated in Figure 4, these differences are negligible. Notably, F-Muon represents a convex combination of Muon and NSGD, despite NSGD performing poorly in isolation. The same observation holds for S-Muon.

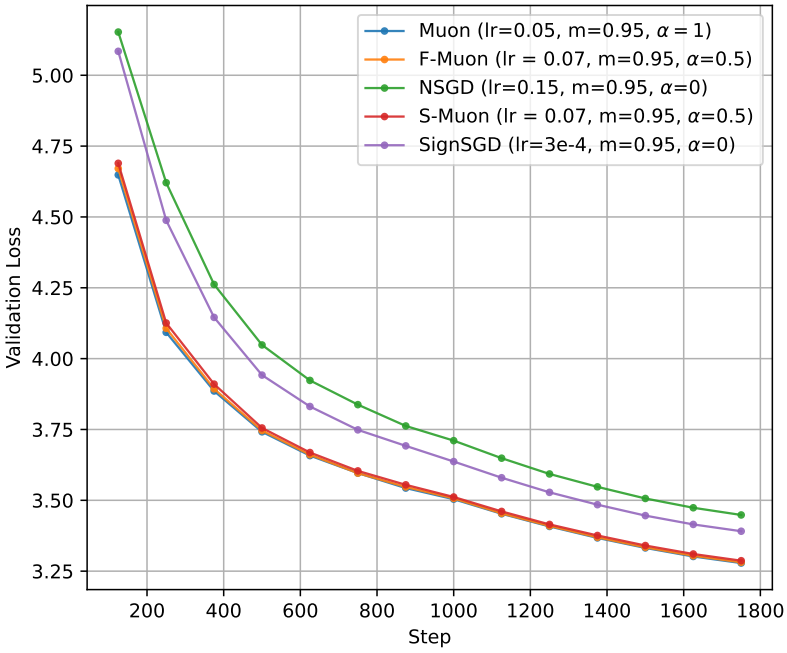


Figure 4: The validation loss for NanoGPT

As observed in the CIFAR airbench experiments, setting $\alpha = 1.1$ for F-Muon (corresponding to the no-ball configuration) yields a loss of 3.2818, representing only a marginal difference from the baseline.

Notably, F-Muon with the NSGD component not scaled by Muon's rule yields a loss of 3.288 (an increase of 0.007), while S-Muon scaled with this rule achieves a loss of 3.292 (an increase of 0.005).

6.4. GPT-2 Medium Speedrun

We scale from NanoGPT to GPT-2 Medium. After testing for 5960 steps we get 2.9215 loss for F-Muon and 2.9235 for S-Muon, while for Muon it is 2.9198 (the speedrun threshold is 2.92).

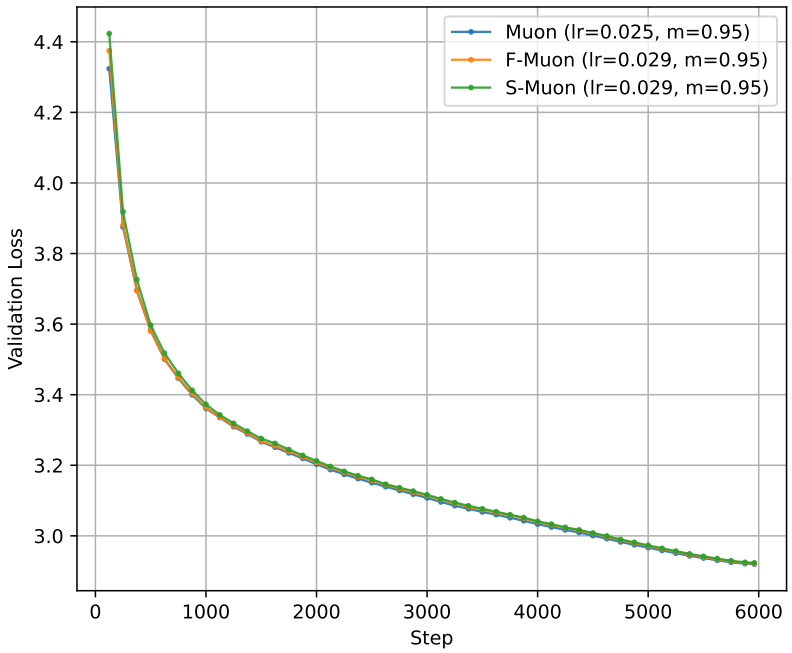


Figure 5: The validation loss for GPT-2 Medium

6.5. NanoGPT Fine-Tuning

We fine-tuned the NanoGPT framework by Karpathy (Karpathy, 2022) using the standard *NanoGPT-medium* configuration, which corresponds to GPT-2 Medium. This model consists of 24 transformer layers, a hidden dimension of 1024, 16 attention heads, and approximately 345 million parameters. We selected the *Tiny Stories* corpus (Eldan and Li, 2023) as the training dataset for its high entropy and structural diversity, which amplify the differences in optimization dynamics. All training runs were initialized with the same random seed, weight initialization, and learning rate schedule to ensure that performance differences arose solely from the choice of optimizer.

For all one-dimensional layers, we used AdamW with a learning rate of 1×10^{-3}. Since momentum exhibited negligible influence on training dynamics, it was held constant across all experiments. Figure 6 presents a comparative analysis of optimization performance.

Figure 7 presents the training and validation loss curves at the optimal learning rate for each optimizer. While F-Muon and S-Muon maintain consistent behavior, vanilla Muon achieves the lowest loss overall, outperforming other algorithms by a small margin.

7. Related Work and Discussion

7.1. Algorithms for Vectors $\rightarrow$ Algorithms for Matrices

The updates of LMO optimizers exhibit striking similarities between vector l_p norms and matrix Schatten S_p norms, as illustrated in Table 4. These parallels extend beyond the

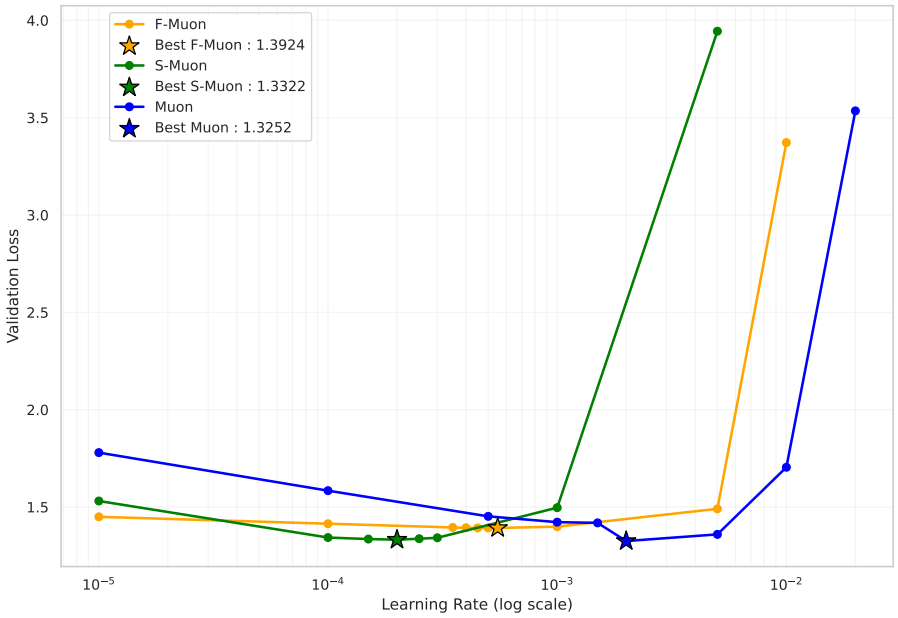


Figure 6: Comparison of validation loss for Muon, F-Muon, and S-Muon across a range of learning rates. Stars denote the best learning rate identified for each optimizer.

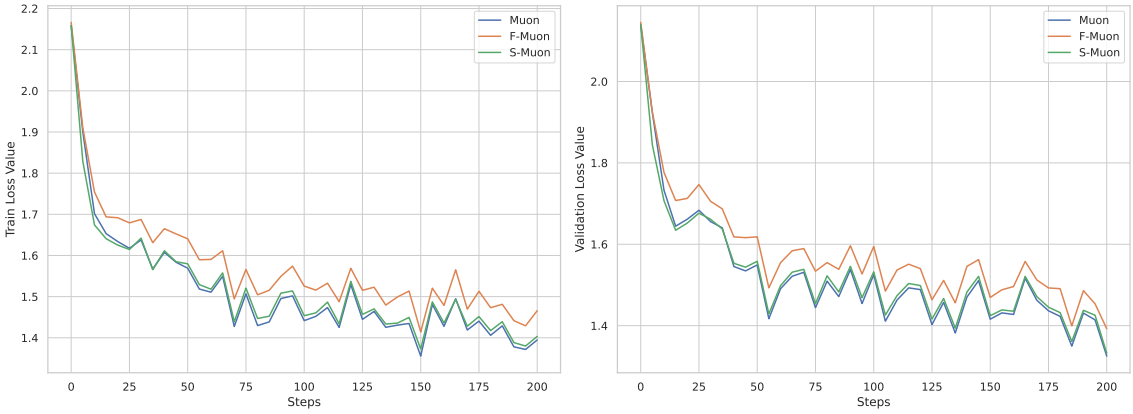


Figure 7: Train and validation loss at the optimal learning rate for each optimizer.

update formulas themselves to empirical performance characteristics. SignSGD closely resembles Adam, as noted in (Bernstein and Newhouse, 2024), and both Adam and Muon perform well during training large models (Zhao et al., 2025; Liu et al., 2025). NSGD maintains identical updates in both vector and matrix cases. Greedy Coordinate Descent is rarely applied to high-dimensional problems, which provides perspective on why one-rank Neon underperforms in such settings.

Table 4: LMO optimizers in Schatten S_p norms and in l_p norms. g is the gradient. When it is a matrix, $g = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$

Algorithm	LMO constraint set $\mathcal{D}$	LMO	Reference
Normalized SGD	l_2-ball, S_2-ball	$-\eta \frac{g}{\|g\|_2} = -\eta \frac{g}{\|g\|_{\mathrm{F}}}$	(Hazan et al., 2015)
Momentum Normalized SGD	Ball in l_2, or Ball in S_2	$-\eta \frac{g}{\|g\|_2} = -\eta \frac{g}{\|g\|_{\mathrm{F}}}$	(Cutkosky et al., 2020)
SignSGD	Ball in Max-norm l_∞	$-\eta\,\mathrm{sign}(g)$	(Bernstein et al., 2018, Thm. 1)
Signum	Ball in Max-norm l_∞	$-\eta\,\mathrm{sign}(g)$	(Bernstein et al., 2018, Thm. 3)
Muon	Ball in Spectral S_∞	$-\eta \boldsymbol{U}\boldsymbol{V}^\top$	(Jordan et al., 2024b)
Gauss-Southwell Coordinate Descent	Ball in l_1	$-\eta \sum_{i \in \arg\max \lvert g_i^t \rvert} \mathrm{sign}(g_i^t) e_i$	(Shi et al., 2016, p. 19)
Neon	Ball in Nuclear S_1	$-\eta \boldsymbol{u}_1 \boldsymbol{v}_1^\top$	This work

7.2. Improvements of Muon

Since Muon (Jordan et al., 2024b) is a highly efficient optimizer for functions of weight matrices, substantial research has focused on two objectives: further improving its performance and explaining its success.

A large number of applications and improvements of Muon have been proposed in less than a year. (Liu et al., 2025) adapted the algorithm for training language models larger than NanoGPT. (Shah et al., 2025) enabled efficient hyperparameter transfer by combining Muon with maximal update parametrization. To construct their COSMOS optimizer, (Chen et al., 2025) applied computationally intensive updates of the SOAP optimizer (Vyas et al., 2025) to a low-dimensional "leading eigensubspace" while using memory-efficient methods like Muon for the remaining parameters. (Amsel et al., 2025; Grishina et al., 2025) proposed more efficient alternatives to the Newton-Schulz algorithm. (Si et al., 2025) introduced AdaMuon, which combines element-wise adaptivity with orthogonal updates. We expect that similar techniques can be applied to our optimizers as well. For example, F-Muon and S-Muon also benefit from faster alternatives to Newton-Schulz iterations, and Fanions may serve as an effective substitute for Muon in COSMOS, as we have shown in Section 5 that the Lanczos algorithm is substantially faster than Newton-Schulz iterations on very large matrices.

7.3. Theory behind Muon

There has been a prolonged gap in the theoretical understanding of Muon, excluding the simplistic derivation of (Bernstein, 2025) based on (Bernstein and Newhouse, 2024). This gap, in our view, remains incompletely closed. For example, (Kovalev, 2025) provided convergence guarantees for Muon in various settings, from which, however, Muon's empirical

supremacy cannot be recovered. Indeed, although the obtained bounds depend on the norm choice, the convergence asymptotics remain the same as for NSGD and other optimizers: $K = \mathcal{O}(\varepsilon^{-4})$ in the L-smooth stochastic case.

A similar limitation affects (Riabinin et al., 2025), where the L-smoothness assumption is replaced with a more practical (L_0, L_1)-smoothness. By estimating smoothness and substituting it into their Theorem 1, the authors recovered the optimal fine-tuned step sizes reported by (Pethick et al., 2025c). However, they did not demonstrate the optimality of the spectral or RMS-to-RMS norm, which is observed in practice, as our comparison with NSGD highlights.

A common limitation of these analyses is their focus on convergence measured by the gradient norm. As we showed in our CIFAR experiments Section F, the gradient norm may decrease by only a factor of ten when the accuracy reaches 100%.

We hypothesize that the stark performance discrepancy between Neon and Muon, both of which are described by the Stochastic Conditional Gradient (Pethick et al., 2025c) or Gluon frameworks, lies in the structure of the norm ball or in the preconditioner interpretation (Pethick et al., 2025a), which warrants further investigation.

7.4. The LMO and Error Feedback

As previously mentioned, rank-k unsharded Dion without error feedback and update scaling is equivalent to Fanion-k. Since error feedback is crucial for Dion, as demonstrated by the ablation study in (Ahn et al., 2025), F-Fanions and S-Fanions would benefit from it as well. In federated learning, error feedback proves effective even for compressed Muon (Gruntkowska et al., 2025a). Fanions and S-Fanions offer a transmission advantage, requiring fewer bits: $\sum_{i=1}^{k} \boldsymbol{u}_i \boldsymbol{v}_i^\top$ can be efficiently transmitted as $\{\boldsymbol{u}_1, \boldsymbol{v}_1, \ldots, \boldsymbol{u}_k, \boldsymbol{v}_k\}$ ($(m+n)\times k$ floats), while the sign component can be encoded in $m \times n$ bits. Thus, compression is inherently built into the representation. Moreover, there is an intriguing possibility to construct differentially private Fanions and S-Fanions using more optimal non-Gaussian noise, as was done with DP-SignSGD (Jang et al., 2024). We leave this for future research.

7.5. The nuclear norm in the LMO

We discovered during the preparation of this article that the nuclear norm has already been explored in the context of the linear minimization oracle. (Pethick et al., 2025b) applied it to create νSAM, a novel sharpness-aware minimization technique. It would be interesting to substitute $\|\cdot\|_*$ with $\|\cdot\|^\dagger_{\text{KF-}k}$ in their approach. Since the SAM neighborhood becomes more diverse, using $k > 1$ might enhance the accuracy boost while preserving a small memory footprint and minimal time overhead when Dion-style power iterations are employed.

7.6. Orthogonal research

Fanions, F-Fanions, and S-Fanions benefit from the general theoretical description in (Riabinin et al., 2025), where better learning rates can be predicted by calculating the trajectory smoothness. They could be transformed into Drop-Fanions by updating only selected layers, as in (Gruntkowska et al., 2025b). They can be viewed as approximations of the Non-Euclidean Proximal Point Method for the corresponding norms (Gruntkowska and Richtárik,

2025). They can be clipped to produce ClippedScion-like algorithms (Pethick et al., 2025d). They could be made more memory-efficient through the zero-order techniques that proved effective for Muon (Petrov et al., 2025). Finally, the results from (Shulgin et al., 2025) can be used to explain the robustness of Muon to the norm changes observed in our experiments and to theoretically derive faster yet effective approximate schemes for calculating the LMO; power iterations with a limited number of iterations represent a promising direction for this analysis.

8. Conclusion

In this article, we addressed the central question of why one should constrain by the spectral or any other operator norm in deriving Muon-like updates, and how alternative norms affect performance and computational cost. Our answer is that the choice of matrix norm is remarkably flexible: properly-tuned variants based on alternative norms can match or even slightly exceed Muon's performance on real-world tasks, while offering additional benefits such as improved learning rate robustness.

We generalized several successful algorithms, including Muon, Dion, and νSAM, to LMO-based algorithms using the family of norms dual to Ky Fan k-norms, yielding the Fanion family with low-rank updates. We further proposed the technique of regularizing these updates by combining them with NSGD or SignSGD, creating the F-Fanion and S-Fanion families through the $\|\cdot\|^{\dagger}_{\mathrm{F-KF-k}}$ and $\|\cdot\|^{\dagger}_{\mathrm{C^{\dagger}-KF-k}}$ norms. Our experiments demonstrate that F-Muon and S-Muon achieve competitive performance with Muon on CIFAR-10 airbench and NanoGPT tasks, confirming that the underlying norm constraint can be significantly modified without sacrificing effectiveness.

However, our results also reveal that not every LMO-based algorithm is effective: Neon (rank-one Fanion) underperforms despite sharing the same theoretical convergence asymptotics as Muon in existing bounds. We suggest that future work on non-Euclidean LMO algorithms should explain in the corollaries to their convergence theorems the empirical superiority of Muon over other Fanions, and ideally account for the robustness of F-Muon and S-Muon as well. Without such refinements, it is difficult to believe that current theoretical bounds are relevant for practitioners.

Author Contributions

IO suggested using the nuclear norm in the (Bernstein and Newhouse, 2024) framework. DM presented the problem at the MIPT optimization course. IK suggested using composite norms (though not the ones that induce Fanions, F-Fanions or S-Fanions) and helped to draft and rewise the manuscript. NK suggested Lanczos algorithm as the most precise means to compute Fanions' updates, conducted experiments to prove this, and wrote Section 5. AV conducted the finetuning of NanoGPT on Tiny Stories and wrote Section 6.5. All other work was done by AK: constructing Fanions, F-Fanions, S-Fanions, experimenting on the Linear Least Squares and CIFAR-airbench, pretraining NanoGPT and GPT-2 Medium, and writing the manuscript.

Acknowledgements

We thank Dmitry Kovalev and Egor Shulgin for helpful discussions.

References

Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates, 2025. URL https://arxiv.org/abs/2504.05295.

Noah Amsel, David Persson, Christopher Musco, and Robert M Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. *arXiv preprint arXiv:2505.16932*, 2025.

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.

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.

Rajendra Bhatia. *Matrix analysis*, volume 169. Springer Science & Business Media, 2013.

Franz Louis Cesista. Steepest Descent under Schatten-p Norms, February 2025. URL https://leloykun.github.io/ponder/steepest-descent-schatten-p/.

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

Ashok Cutkosky, Harsh Mehta, and Francesco Orabona. Momentum-based variance reduction in nonconvex sgd. *Advances in Neural Information Processing Systems*, 2020.

Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english?, 2023. URL https://arxiv.org/abs/2305.07759.

Ekaterina Grishina, Matvey Smirnov, and Maxim Rakhuba. Accelerating newton-schulz iteration for orthogonalization via chebyshev-type polynomials, 2025. URL https://arxiv.org/abs/2506.10935.

Kaja Gruntkowska and Peter Richtárik. Non-euclidean broximal point method: A blueprint for geometry-aware optimization, 2025. URL https://arxiv.org/abs/2510.00823.

Kaja Gruntkowska, Alexander Gaponov, Zhirayr Tovmasyan, and Peter Richtárik. Error feedback for muon and friends, 2025a. URL https://arxiv.org/abs/2510.00643.

Kaja Gruntkowska, Yassine Maziane, Zheng Qu, and Peter Richtárik. Drop-muon: Update less, converge faster, 2025b. URL https://arxiv.org/abs/2510.02239.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *Advances in neural information processing systems*, 28, 2015.

Jonggyu Jang, Seongjin Hwang, and Hyun Jong Yang. Rethinking DP-SGD in discrete domain: Exploring logistic distribution in the realm of signSGD. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 21241–21265. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/jang24a.html.

Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, @fernbear.bsky.social, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024a. URL https://github.com/KellerJordan/modded-nanogpt.

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

Andrej Karpathy. nanogpt: The simplest, fastest repository for training/finetuning medium-sized gpts. GitHub repository, 2022. URL https://github.com/karpathy/nanoGPT.

Jordan Keller. cifar10-airbench, 2023. URL https://github.com/KellerJordan/cifar10-airbench.

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

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

Dmitry Kovalev and Ekaterina Borodich. Non-euclidean sgd for structured optimization: Unified analysis and improved rates, 2025. URL https://arxiv.org/abs/2511.11466.

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.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Thomas Pethick. Understanding dion, 2025. URL https://pethick.dk/posts/2025-08-18-understanding-dion/.

Thomas Pethick, Kimon Antonakopoulos, Antonio Silveti-Falls, Leena Chennuru Vankadara, and Volkan Cevher. Training neural networks at any scale, 2025a. URL https://arxiv.org/abs/2511.11163.

Thomas Pethick, Parameswaran Raman, Lenon Minorics, Mingyi Hong, Shoham Sabach, and Volkan Cevher. νSAM: Memory-efficient sharpness-aware minimization via nuclear norm constraints. *Transactions on Machine Learning Research*, 2025b. ISSN 2835-8856. URL https://openreview.net/forum?id=V6ia5hWIMD.

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*, 2025c.

Thomas Pethick, Wanyun Xie, Mete Erdogan, Kimon Antonakopoulos, Tony Silveti-Falls, and Volkan Cevher. Generalized gradient norm clipping & non-euclidean (l_0, l_1)-smoothness, 2025d. URL https://arxiv.org/abs/2506.01913.

Egor Petrov, Grigoriy Evseev, Aleksey Antonov, Andrey Veprikov, Nikolay Bushkov, Stanislav Moiseev, and Aleksandr Beznosikov. Leveraging coordinate momentum in signsgd and muon: Memory-optimized zero-order, 2025. URL https://arxiv.org/abs/2506.04430.

Inc. Preferred Infrastructure and CuPy Developers. CuPy: `cupyx.scipy.sparse.linalg.svds` — api reference. https://docs.cupy.dev/en/stable/reference/generated/cupyx.scipy.sparse.linalg.svds.html, 2025. Accessed: 2025-08-24.

Artem Riabinin, Egor Shulgin, Kaja Gruntkowska, and Peter Richtárik. Gluon: Making muon & scion great again!(bridging theory and practice of lmo-based optimizers for llms). *arXiv preprint arXiv:2505.13416*, 2025.

Ishaan Shah, Anthony M Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, et al. Practical efficiency of muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

Hao-Jun Michael Shi, Shenyinying Tu, Yangyang Xu, and Wotao Yin. A primer on coordinate descent algorithms. *arXiv preprint arXiv:1610.00040*, 2016.

Egor Shulgin, Sultan AlRashed, Francesco Orabona, and Peter Richtárik. Beyond the ideal: Analyzing the inexact muon update, 2025. URL https://arxiv.org/abs/2510.19933.

Chongjie Si, Debing Zhang, and Wei Shen. Adamuon: Adaptive muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025.

Kesheng Wuzand Horst Simonz. Thick-restart lanczos method for symmetric eigenvalue problemsy. 1998.

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. URL https://arxiv.org/abs/2409.11321.

Yao-Liang Yu. Arithmetic duality for norms, 2012. URL https://cs.uwaterloo.ca/~y328yu/notes/normduality.pdf.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham M. Kakade. Deconstructing what makes a good optimizer for autoregressive language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zfeso8ceqr.

Appendix A. LMO for Neural Networks

In a typical neural network, the objective function F depends on a set of weight matrices $\{\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots\}$. The optimization framework we have described is applied in a layer-wise fashion. At each iteration t, a stochastic gradient $g(\boldsymbol{X}^t, \xi^t)$ is computed using a mini-batch of data with the ξ^t noise via backpropagation. This yields a separate gradient component, $\boldsymbol{G}_i^t$, for each matrix $\boldsymbol{X}_i$. The LMO-based update rule is then applied to each matrix $\boldsymbol{X}_i$ using its corresponding gradient component $\boldsymbol{G}_i^t$.

Appendix B. Proof of Lemma on the Dual to Convex Combinations

We provide here the proof of Lemma 3.

Proof Let us first prove the lemma for the case $n = 2$. Denote $f(x) = \alpha_1\|x\|_{(1)}$ and $g(x) = \alpha_2\|x\|_{(2)}$, such that $\|x\| = f(x) + g(x)$. Recall two standard facts:

1. For any norm $\|\cdot\|$ and $\lambda > 0$,

$$(\lambda\|\cdot\|)^*(y) = \sup_x\big(\langle y, x\rangle - \lambda\|x\|\big) = \delta_{\lambda\mathcal{B}_{\|\cdot\|^\dagger}}(y)\,,$$

   i.e., the indicator function of the scaled dual ball.

2. The Fenchel conjugate of a sum satisfies

$$(f+g)^*(y) = \inf_{u+v=y}\big(f^*(u) + g^*(v)\big)\,.$$

Applying these to f and g, we have

$$f^*(u) = \delta_{\alpha_1\mathcal{B}_{\|\cdot\|^\dagger_{(1)}}}(u)\,, \quad g^*(v) = \delta_{\alpha_2\mathcal{B}_{\|\cdot\|^\dagger_{(2)}}}(v)\,.$$

Thus,

$$\|\cdot\|^*(y) = (f+g)^*(y) = \inf_{u+v=y}\big(\delta_{\alpha_1\mathcal{B}_{\|\cdot\|^\dagger_{(1)}}}(u) + \delta_{\alpha_2\mathcal{B}_{\|\cdot\|^\dagger_{(2)}}}(v)\big) = \delta_{\alpha_1\mathcal{B}_{\|\cdot\|^\dagger_{(1)}} + \alpha_2\mathcal{B}_{\|\cdot\|^\dagger_{(2)}}}(y).$$

By definition, the conjugate of a norm is exactly the indicator of its dual unit ball:

$$\|\cdot\|^*(y) = \delta_{\mathcal{B}_{\|\cdot\|^\dagger}}(y)\,.$$

Therefore, $\mathcal{B}_{\|\cdot\|^\dagger} = \alpha_1\mathcal{B}_{\|\cdot\|^\dagger_{(1)}} + \alpha_2\mathcal{B}_{\|\cdot\|^\dagger_{(2)}}$.

Now we prove the general case by induction. The base case ($n = 2$) is already proven. Suppose that the assumption of the lemma holds for $n = k$. Then, for $n = k + 1$,

$$\|x\| = \sum_{i=1}^{k} \alpha_i \|x\|_{(i)} + \alpha_{k+1}\|x\|_{(k+1)} = \|x\|_{(1:k)} + \alpha_{k+1}\|x\|_{(k+1)}\,.$$

Applying the result for $n = 2$ combined with the induction assumption, we obtain

$$\mathcal{B}_{\|\cdot\|^\dagger} = \mathcal{B}_{\|\cdot\|^\dagger_{(1:k)}} + \alpha_{k+1}\mathcal{B}_{\|\cdot\|^\dagger_{(k+1)}} = \sum_{i=1}^{k+1} \alpha_i \mathcal{B}_{\|\cdot\|^\dagger_{(i)}}\,,$$

which proves the lemma. ■

Appendix C. Norms $\|\cdot\|^\dagger_{\mathrm{F}*}$ and $\|\cdot\|^\dagger_{\mathrm{F2}}$

Based on the Lemma 3, we immediately find $\|\cdot\|^\dagger_{\mathrm{F}*}$, which is related to F-Muon update. Indeed, after setting $\beta = 1 - \alpha$ and remembering that for smooth and bounded cases we can use min instead of inf, we get

$$\|\boldsymbol{Y}\|^\dagger_{\mathrm{F}*} = \min_{\boldsymbol{Z}} \min_{t} \{t, s.t. \|\boldsymbol{Z}\|_2 \le \alpha t, \|\boldsymbol{Y} - \boldsymbol{Z}\|_{\mathrm{F}} \le (1-\alpha)t\} \tag{16}$$

If $\alpha = 1$, then $\boldsymbol{Z} = \boldsymbol{Y}$, and we get $\|\boldsymbol{Y}\|^\dagger_{\mathrm{F}*} = \|\boldsymbol{Y}\|_2$. If $\alpha = 0$, then $\boldsymbol{Z} = 0$, and we get $\|\boldsymbol{Y}\|^\dagger_{\mathrm{F}*} = \|\boldsymbol{Y}\|_{\mathrm{F}}$.

Similarly, we find $\|\cdot\|^\dagger_{\mathrm{F2}}$, which is related to F-Neon update:

$$\|\boldsymbol{Y}\|^\dagger_{\mathrm{F2}} = \min_{\boldsymbol{Z}} \min_{t} \{t, s.t. \|\boldsymbol{Z}\|_* \le \alpha t, \|\boldsymbol{Y} - \boldsymbol{Z}\|_{\mathrm{F}} \le (1-\alpha)t\} \tag{17}$$

If $\alpha = 1$, then $\boldsymbol{Z} = \boldsymbol{Y}$, and we get $\|\boldsymbol{Y}\|^\dagger_{\mathrm{F2}} = \|\boldsymbol{Y}\|_*$. If $\alpha = 0$, then $\boldsymbol{Z} = 0$, and we get $\|\boldsymbol{Y}\|^\dagger_{\mathrm{F2}} = \|\boldsymbol{Y}\|_{\mathrm{F}}$.

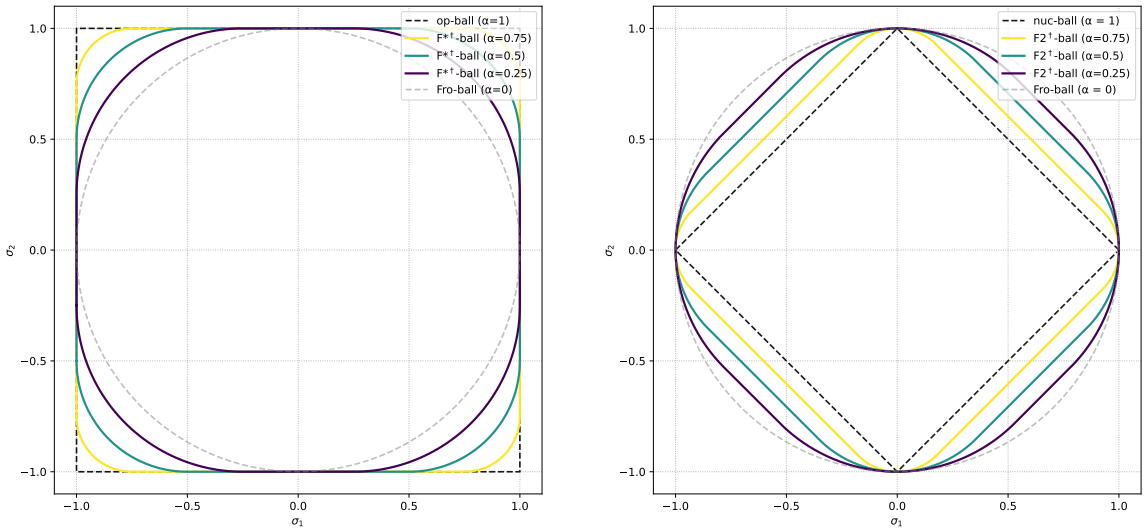
Unfortunately, $\|\cdot\|^\dagger_{\mathrm{F}*}$ and $\|\cdot\|^\dagger_{\mathrm{F2}}$ do not have a closed-form expression.

Appendix D. Visualization of different matrix norms

D.1. Duals to F* and F2 norms

It follows from Lemma 3 that the norm ball in $\|\cdot\|^\dagger_{\mathrm{F}*}$ is the Minkowski sum of the norm ball in $\alpha\|\cdot\|_*$ and $(1-\alpha)\|\cdot\|_{\mathrm{F}}$ and the norm ball in $\|\cdot\|^\dagger_{\mathrm{F2}}$ is the Minkowski sum of the norm ball in $\alpha\|\cdot\|_2$ and $(1-\alpha)\|\cdot\|_{\mathrm{F}}$.

In Figure 8 we plot these norms. On x-axis and y-axis, there are singular values σ_1, σ_2 respectively of a matrix from $\mathbb{R}^{m\times n}$ with $\min\{m, n\} = 2$.



(*a*) lmo balls for F-Muon for different α (*b*) lmo ball for F-Neon for different α

Figure 8: Balls in the duals to F* and F2 norms for different α

D.2. The Ky Fan norm and its dual

1-balls in l_∞, l_1 and l_2 norms are well-known from textbooks. But what about the Ky Fan k-norm? How can it be represented?

To showcase the complex structure of the Ky Fan k-norm and its dual, we suggest the illustrations Figure 9 with the ball in the Ky Fan 2-norm in Figure 9(*a*)subfigure and its dual in Figure 9(*b*)subfigure. On x-, y-, and z-axes, there are singular values σ_1, σ_2, and σ_3 respectively of a matrix from $\mathbb{R}^{m\times n}$ with $\min\{m,n\}=3$. In this particular case, we do not sort the singular values. In the proposed representation, we actually plot balls in the Top-2 norm $\max\{|x|+|y|, |x|+|z|, |y|+|z|\}$ and its dual norm $\max\{\max(|(|x), |y|, |z|), \frac{1}{2}(|x|+|y|+|z|)\}$. The resulting balls are much more complex than balls in l_∞, l_1 and l_2 norms.

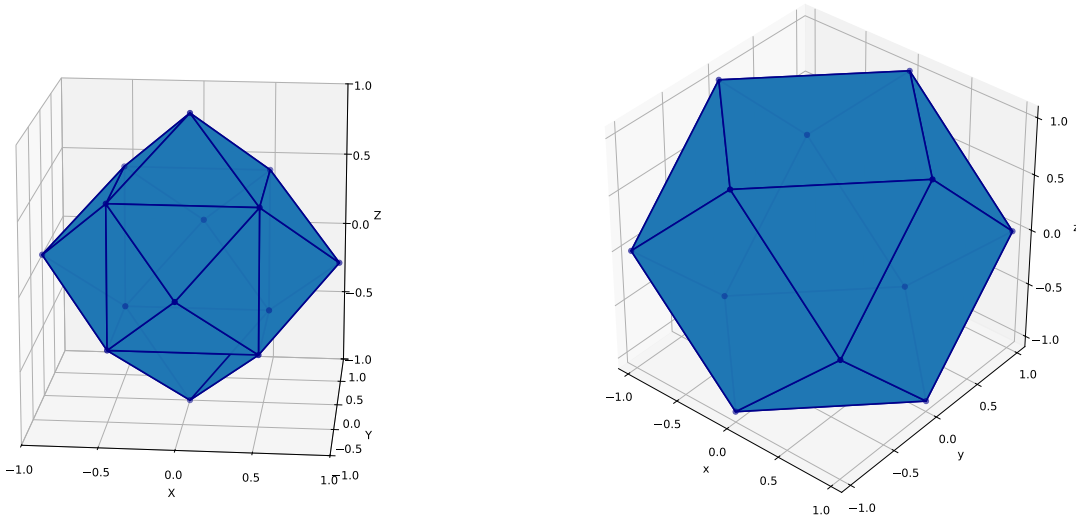
In fact, those balls can be described easier if we use the results from (Yu, 2012). The Ky Fan 2-norm ball is an intersection of three l_1 balls in (x,y), (x,z), and (y,z) spaces. The 1-ball in the dual Ky Fan 2-norm is an intersection of 1-ball the in l_∞ norm and $\frac{1}{2}$-ball in the l_1 norm.

Appendix E. More data for Linear Least Squares

Appendix F. Under the hood of CNN on CIFAR-10

Most bounds for Muon and other LMO-algorithms are given for the norm of the gradient (**?**Kovalev, 2025; Pethick et al., 2025c; Riabinin et al., 2025; Kovalev and Borodich, 2025). Accordingly, it is natural to measure those norms on a real deep learning problem.

We compare Muon, F-Muon, S-Muon, NSGD, SignSGD, Neon, F-Neon, Fanion-5, and F-Fanion-5 on CIFAR-airbench. NSGD and SignSGD are not heavily tuned. The hyperparameters of Neon, F-Neon, Fanion-5, and F-Fanion-5 are taken from Muon and F-Muon (see



(a) The Ky Fan 2-norm

(b) Dual to the Ky Fan 2-norm

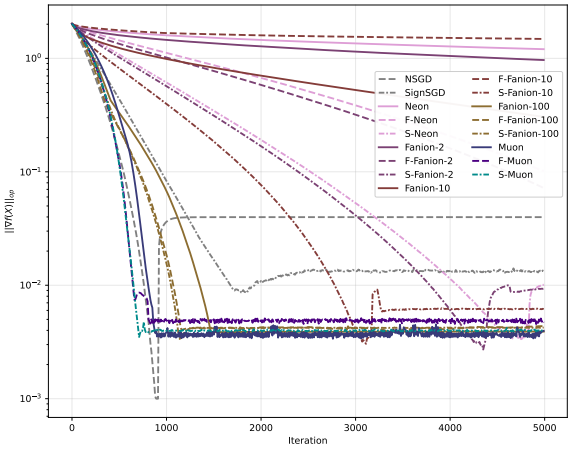
Figure 9: Ky Fan 2-norm and its dual

Table 5: Tuning lrs and momentum coefficients

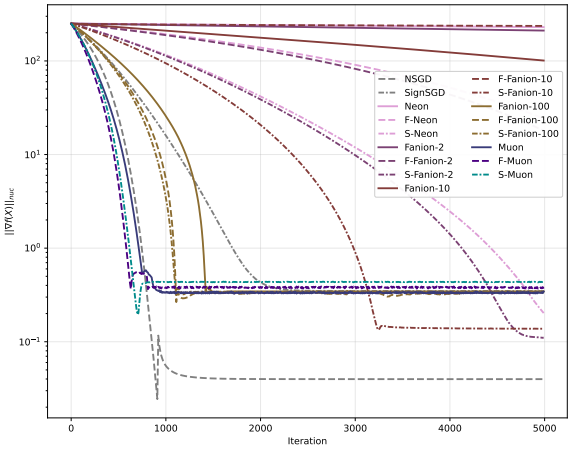
Algorithm	`lr`	`momentum`	Iterations to 0.001 loss
Muon	0.007	0.5	1060
NSGD	0.08	0.95	1020
F-Muon	0.015	0.7	910
SignSGD	0.016×0.01	0.95	2650
S-Muon	0.011	0.9	890

Table 6: Parameters for CIFAR-airbench. `sign_lr_mult` for S-Muon is 0.003

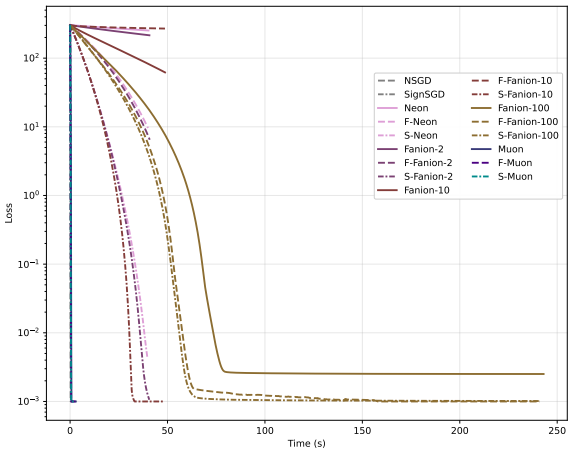
Method	`lr`	`momentum`	val_accuracy, %
NSGD	0.5	0.95	91.6 ± 0.52
SignSGD	0.003	0.95	91.54 ± 0.26
Muon	0.24	0.6	94.01 ± 0.10
F-Muon	0.40	0.6	94.01 ± 0.13
S-Muon	0.42	0.63	94.03 ± 0.13
Neon	0.24	0.6	69.8 ± 0.5
F-Neon	0.40	0.6	87.15 ± 0.24
Fanion-5	0.24	0.6	80.69 ± 1.25
F-Fanion-5	0.40	0.6	86.66 ± 0.65



(a) The spectral norm of the gradient



(b) The nuclear norm of the gradient

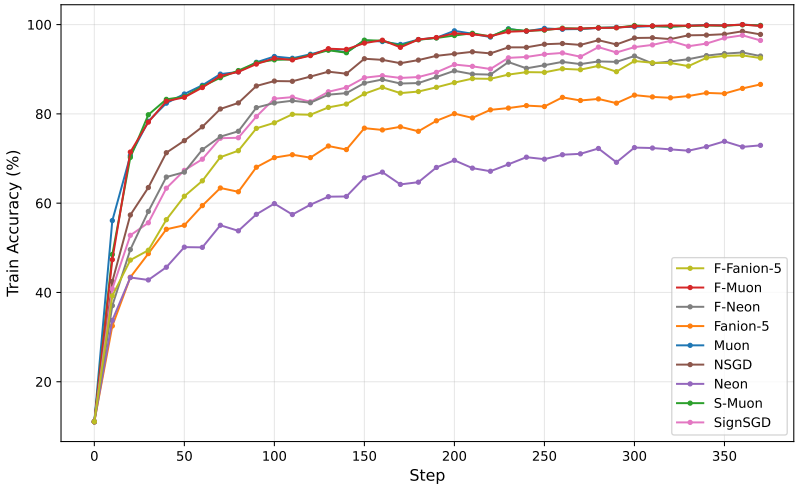


(c) The loss over time

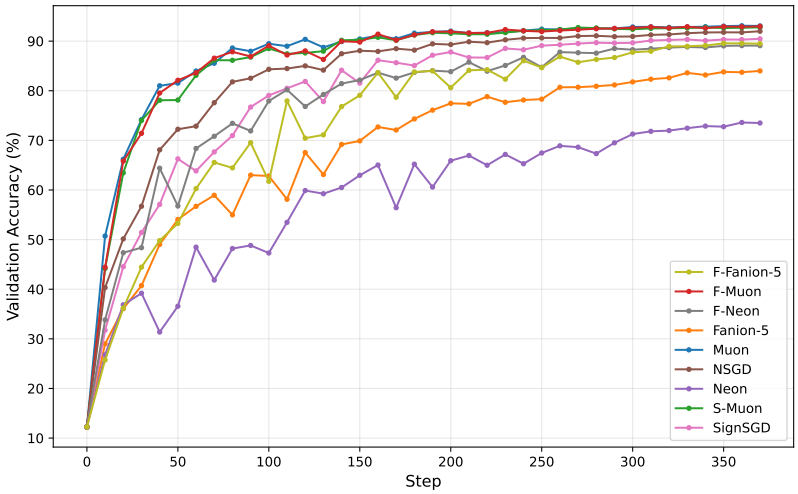
Figure 10: More images for Linear least squares problem for a 500x500 matrix

Table 6). In the table, val_accuracy is for the the weighted variant. However, to preserve the algorithms, we do not normalize the weights of the network each iteration.

We log train and validation accuracy, and nuclear, gradient, and spectral norms of the gradients of conv1.weight and conv2.weight for all the layers. The gradient norms decrease only by the order of ten maximum during training, while train accuracy reaches full 100% for Muon, S-Muon, and F-Muon.



(a) Train accuracy

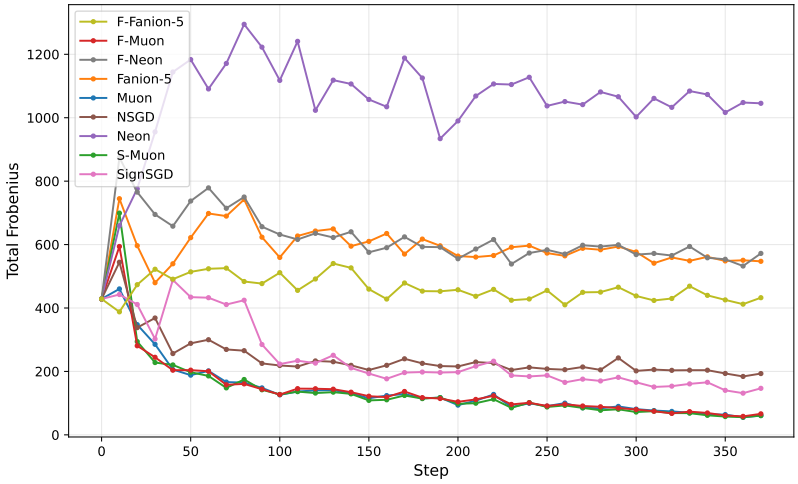


(b) Validation accuracy

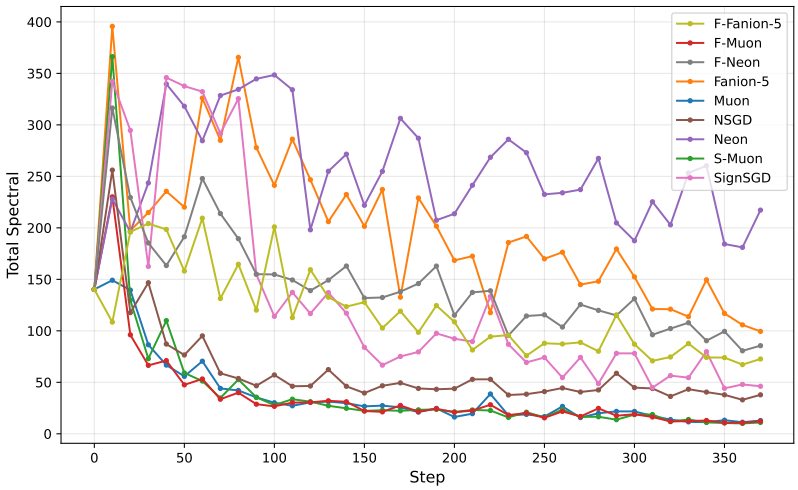
Figure 11: Different optimizers on CIFAR airbench without weight normalization

Appendix G. Technical details of the experiments

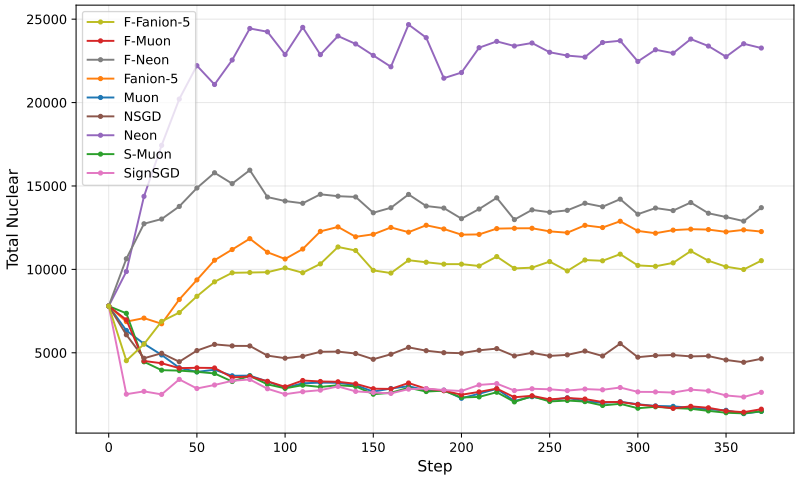
CIFAR airbench We used NVIDIA RTX A4000.



(a) Total Frobenius norm

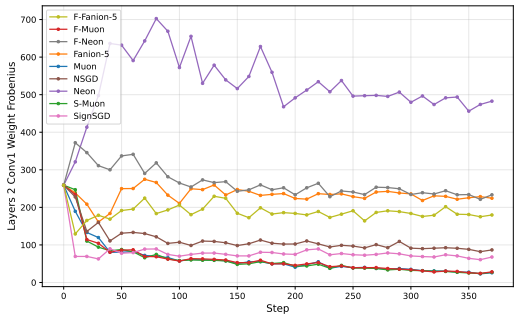


(b) Total spectral norm

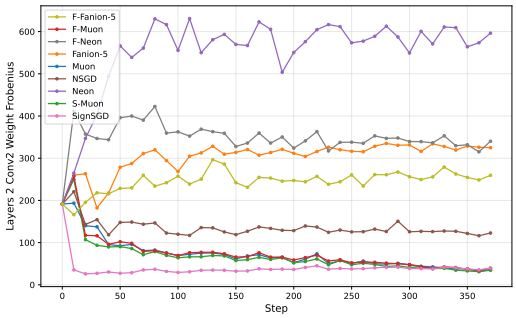


(c) Total nuclear norm

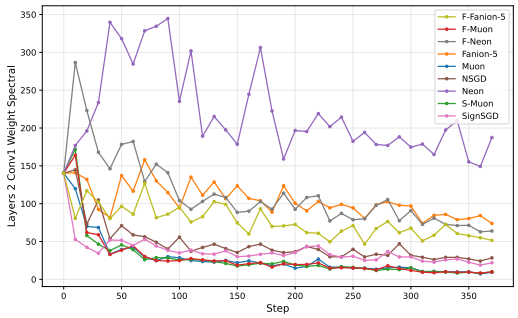
Figure 12: Matrix norms of the whole gradient of CIFAR CNN



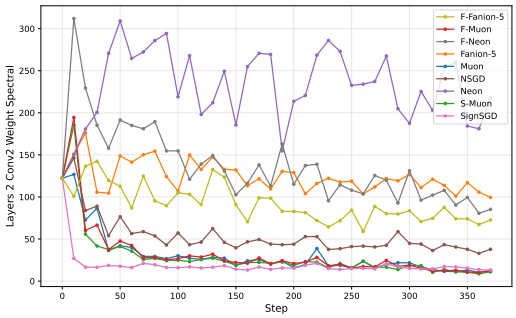
(*a*) Frobenius norm of layer2.conv1



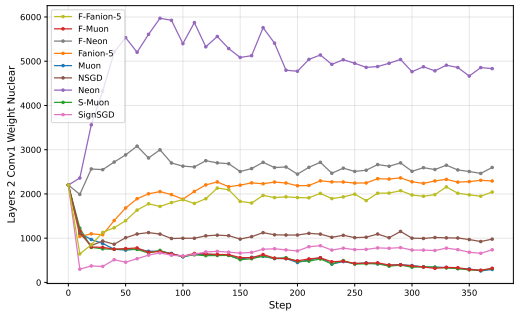
(*b*) Frobenius norm of layer2.conv2



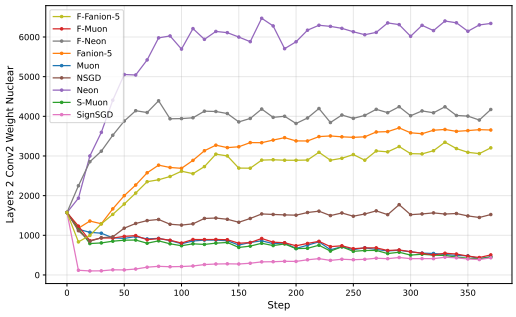
(*c*) Spectral norm of layer2.conv1



(*d*) Spectral norm of layer2.conv2



(*e*) Nuclear norm of layer2.conv1



(*f*) Nuclear norm of layer2.conv2

Figure 13: Matrix norms of layer2.conv1 (left) and layer2.conv2 (right) gradients of CIFAR CNN

NanoGPT pretrain We used the standard setting of 8 × H100 as documented in (Jordan et al., 2024a).

NanoGPT Fine-tuning. Experiments were conducted on a single NVIDIA RTX 4090 (24GB) GPU using PyTorch 2.8 with CUDA 12.8 and cuDNN 9.0, running on a workstation equipped with an Intel Core i9-14900KS CPU and 128 GB of RAM. No distributed or mixed-hardware training setups were used, ensuring a strictly controlled and unbiased comparison across optimizers.

A uniform training protocol was applied to all runs, including identical batch size, warm-up and cosine decay learning rate schedule, gradient clipping strategy, and validation split. Our optimization methods were integrated into the NanoGPT training loop without modifying the model architecture or preprocessing pipeline. Model quality was primarily evaluated using validation loss tracked over 200 training steps.