HTMuon: Improving Muon via Heavy-Tailed Spectral Correction

Tianyu Pang[*1], **Yujie Fang**[*2], **Zihang Liu**[3,4], **Shenyang Deng**[1], **Lei Hsiung**[1], **Shuhua Yu**[5], **Yaoqing Yang**[1]
[1]Dartmouth College
[2]Microsoft
[3]International Computer Science Institute
[4]University of California, Berkeley
[5]Meta

Abstract

Muon has recently shown promising results in LLM training. In this work, we study how to further improve Muon. We argue that Muon's orthogonalized update rule suppresses the emergence of heavy-tailed weight spectra and overemphasizes the training along noise-dominated directions. Motivated by the Heavy-Tailed Self-Regularization (HT-SR) theory, we propose HTMuon. HTMuon preserves Muon's ability to capture parameter interdependencies while producing heavier-tailed updates and inducing heavier-tailed weight spectra. Experiments on LLM pretraining and image classification show that HTMuon consistently improves performance over state-of-the-art baselines and can also serve as a plug-in on top of existing Muon variants. For example, on LLaMA pretraining on the C4 dataset, HTMuon reduces perplexity by up to 0.98 compared to Muon. We further theoretically show that HTMuon corresponds to steepest descent under the Schatten-q norm constraint and provide convergence analysis in smooth non-convex settings. The implementation of HTMuon is available at https://github.com/TDCSZ327/HTmuon.

1 Introduction

Optimizers play a central role in training Large language models (LLMs). A well-designed optimizer can help LLMs learn more effectively from large-scale datasets (Semenov et al., 2025), leading not only to lower training loss but also to improved generalization (Foret et al., 2020; Kaddour et al., 2022). Over the decades, optimizers have often been developed from a *vector-based* view, where gradients are treated as flattened vectors during updates. Within this framework, Adam (Kingma, 2014) and AdamW (Loshchilov and Hutter, 2017) have become the default choices for training LLMs (Groeneveld et al., 2024; OLMo et al., 2024). A key reason is that they incorporate first- and second-order moment information in an element-wise manner, allowing them to adapt to the different statistical patterns (e.g., local geometry of the loss landscape) across individual coordinates. However, an overly element-wise update scheme often ignores interdependencies among coordinates (e.g., geometric correlations among parameters), which can limit the effectiveness of the optimizer. Although recent work (Zhou et al., 2023; Liu et al., 2024b; Zhang et al., 2024b; Wang et al., 2025a) has gone beyond Adam and its variants by using layer-wise or module-wise learning rates to capture parameters' relationships, whether these adjustments are sufficient to capture the complex coupling across these parameters remains to be seen.

Recently, Muon (Jordan et al., 2024) has been proposed as a representative *matrix-based* optimizer[*]. Building on earlier matrix-based optimizers such as AdaGrad (Duchi et al., 2011) and Shampoo (Gupta et al., 2018), Muon performs preconditioning on the momentum matrix, and its update can be interpreted as an orthogonalization step, which effectively captures geometry interdependencies among parameters. Moreover, the orthogonalization update can be viewed as the steepest descent under the Schatten$-\infty$ norm constraint (Bernstein and Newhouse, 2024; Pethick et al., 2025), which leads to a competitive convergence rate and improved stability (Shen et al., 2025; Chen et al., 2025; Ma et al., 2026). Muon has shown promising results (Liu et al., 2025a; Shah et al., 2025; Wen et al., 2025; Wang et al., 2025b) and has been adopted in large-scale LLM training, including Moonshot's Kimi K2 (Team et al., 2025) and GLM-4.5 (Zeng et al., 2025).

However, some studies have found that the improved performance of Muon is inversely proportional to the model scale and the number of training steps (Wen et al., 2025; Semenov et al., 2025). The orthogonalization step in Muon update sets all

[*]Equal contribution.

[*]*Matrix-based* optimizers use matrix-valued preconditioners, which can better capture geometry-induced dependencies among different parameters.

Findings of the Association for Computational Linguistics: ACL 2026, pages 36504–36535
July 2-7, 2026

singular values of the momentum matrix to one, which means it assigns the same weight to each singular-vector direction of the updates. Although this may help Muon decrease the loss faster in the early phase of training (Shen et al., 2025), it is well known that directions associated with small singular values tend to be more noise-dominated (Sharma et al., 2023; Chen et al., 2024; Liu et al., 2025d; Defilippis et al., 2025). In Section 2.2, we conduct experiments to show that using uniform weights on all singular vector directions may be suboptimal.

More importantly, the orthogonalization step makes the spectrum of the momentum update matrix light-tailed. This, in turn, makes the spectrum of learned weight matrices light-tailed. Here, the spectrum refers to the *empirical spectral density* (ESD) of the weight matrices. However, a line of work by Martin and Mahoney (2021); Martin et al. (2021) shows that *well-trained* neural networks—namely, networks that have extracted strong correlations in their weights through learning from data—tend to have heavy-tailed ESDs in their weight matrices (Kothapalli et al., 2025; Hodgkinson et al., 2025). Within a reasonable range, the degree of heavy-tailedness is strongly correlated with model quality (Martin et al., 2021). Motivated by these observations, they proposed the Heavy-Tailed Self-Regularization theory (HT-SR), which has been supported by extensive empirical studies and further theoretical analyses (Yang et al., 2023; Martin and Hinrichs, 2025; Defilippis et al., 2025). In Section 2.3, we empirically show that the Muon optimizer leads to a less heavy-tailed weight ESD, and it performs worse than variants of Muon that induce more heavy-tailed spectra. Based on this, we claim that the Muon update rule, which sets all singular values of the momentum matrix to one, limits the final quality that the model can achieve.

Motivated by this observation, we propose a matrix-based optimizer HTMuon. HTMuon aims to make Muon's momentum update more heavy-tailed, which can lead to more heavy-tailed weight ESDs, while preserving Muon's advantage in capturing interdependencies among parameters. The design of HTMuon is simple: we raise the singular values of the momentum matrix to the power of p, where $p \in (0, 1)$. See Algorithm 1. Setting $p = 1$ reduces the method to SGDM (Sutskever et al., 2013), whose updates are independent across parameters. In contrast, $p = 0$ recovers Muon, which, as discussed earlier, leads to less heavy-tailed updates. Therefore, we choose $p \in (0, 1)$ in HTMuon to maintain the *matrix-based* ability to model parameter coupling, while producing updates that are more heavy-tailed than those of Muon. Unless otherwise specified, we use $p = 0.125$ for HTMuon throughout. We discuss the specific choice of p in Section 4.7. To evaluate the empirical performance of HTMuon, we study a range of models and tasks, including training LLaMA models (Touvron et al., 2023) on the C4 dataset, GPT-2 models on the OpenWebText dataset and training image classification models using the ResNet (He et al., 2016) and ViT (Dosovitskiy, 2020) architectures. All of these tasks are standard and have been widely used in prior literature (He et al., 2025; Zhao et al., 2024; Yuan et al., 2024). We compare HTMuon against commonly used baseline optimizers such as Adam, Muon, Cautious (Liang et al., 2024), GaLore (Zhao et al., 2024), Sophia (Liu et al., 2024a), Mars (Yuan et al., 2024), SOAP (Vyas et al., 2024), COSMOS (Liu et al., 2025b), and show that HTMuon not only achieves superior performance but can also serve as a plug-in module on top of existing Muon variants like NorMuon (Li et al., 2025), AdaMuon (Si et al., 2025) to further improve test performance. Moreover, our analysis reveals that HTMuon produces update and weight matrices that are more heavy-tailed than those of Muon during training, consistent with prior work on HT-SR theory. Finally, we present theoretical results showing that HTMuon is equivalent to the steepest-descent method under the Schatten$-q$ norm constraint, generalizing Muon's equivalence to steepest descent under a Schatten$-\infty$ norm constraint, this provides a matrix-version of pbSGD (Zhou et al., 2020) and also places HTMuon within the linear minimization oracle framework (Pethick et al., 2025; Bernstein and Newhouse, 2024). We also provide a convergence analysis, showing that HTMuon matches the sample complexity upper bound of Muon and SGDM in smooth non-convex settings.

To summarize, the main contributions of our work are the following:

- We argue that Muon's orthogonalized update rule biases updates toward noise-dominated directions and suppresses the emergence of heavy-tailed ESDs in the model's weight matrices, thereby limiting generalization performance according to HT-SR theory.

- We propose HTMuon, an optimizer that makes Muon updates more heavy-tailed while preserving Muon's strength in modeling parameter interdependence. We demonstrate HTMuon's

effectiveness in improving performance on both LLM pretraining and image classification tasks relative to state-of-the-art optimizers including Cautious (Liang et al., 2024), Mars (Yuan et al., 2024), SOAP (Vyas et al., 2024), NorMuon (Li et al., 2025), AdaMuon (Si et al., 2025), COSMOS (Liu et al., 2025b), etc. For example, HTMuon reduces perplexity by 0.92 compared to Muon when training LLaMA-60M on the C4 dataset, and by 0.98 when training LLaMA-135M on C4. Moreover, HTMuon can be used as an add-on method in combination with existing Muon variants to achieve further improvements. We also design two accelerated implementations of HTMuon. Using these implementations, we show that HTMuon outperforms Muon on LLaMA-1B, highlighting its potential for training large-scale models.

- We present theoretical results analyzing HTMuon. Specifically, we show that it is equivalent to steepest descent under a Schatten$-q$ norm constraint, and we provide a convergence analysis demonstrating that HTMuon matches the sample-complexity upper bounds of Muon and SGDM in nonconvex smooth settings. These results support HTMuon's competitive convergence rate and improved training stability.

2 Motivation for HTMuon

In this section, we first briefly introduce HT-SR theory and then use two examples to motivate the importance of heavy-tailed spectra for the design of HTMuon.

2.1 Background on HT-SR Theory

The HT-SR theory (Martin and Mahoney, 2021; Martin et al., 2021) offers a principled lens for analyzing the ESD of neural network weight matrices. Empirically, well-trained models tend to exhibit more heavy-tailed ESDs, and the degree of heavy-tailedness correlates with training quality. Based on prior work (Martin and Mahoney, 2021; Zhou et al., 2023), heavy-tailedness is quantified by fitting a power law (PL) to the ESD and using the resulting PL exponent α as the metric.

Given a network with L layers and weight matrices $\{\boldsymbol{W}_l\}_{l=1}^{L}$of shape $n \times m$, we compute the ESD by extracting the eigenvalues of the correlation matrix $\boldsymbol{X}_l = \boldsymbol{W}_l^\top \boldsymbol{W}_l$ for each module. We fit a power law (PL) to the ESD in the form $p(\lambda) \propto \lambda^{-\alpha}, \quad \lambda_{\min} < \lambda < \lambda_{\max}$, where $p(\lambda)$ denotes the eigenvalue density within the specified range. The PL exponent α serves as a proxy for the degree of heavy-tailedness, smaller α indicates a more heavy-tailed weight ESD, and more heavy-tailed spectra are often associated with better model quality.

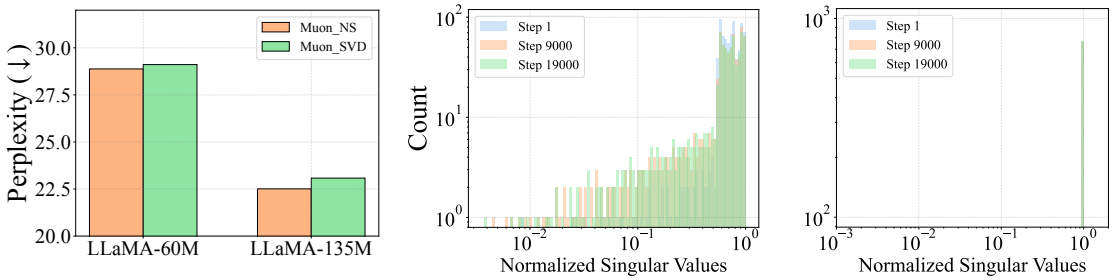
2.2 Unit Singular Values May Be Suboptimal

In this subsection, we introduce an intriguing empirical finding: for Muon implementation, the numerically accelerated version by Newton Schulz, which we call Muon_NS (see Algorithm 3) can outperform the theoretically exact implementation Muon_SVD (see Algorithm 4).

The theoretical Muon update in Algorithm 4 has a similar form to polar decomposition: $\boldsymbol{O}_t = \boldsymbol{U}_t\boldsymbol{V}_t^\top = (\boldsymbol{M}_t\boldsymbol{M}_t^\top)^{-\frac{1}{2}}\boldsymbol{M}_t$. This is equivalent to setting all singular values of the momentum matrix $\boldsymbol{M}_t$ to one. In practice, however, computing $\boldsymbol{U}_t$ and $\boldsymbol{V}_t$ exactly via SVD is expensive, so Muon uses a Newton_Schulz iteration (Higham and Schreiber, 1990) to approximate $\boldsymbol{O}_t$. In Figure 1, we train LLaMA-60M and LLaMA-135M on C4 dataset using Muon_SVD and Muon_NS, with identical hyperparameters for each model. Figure 1c visualizes the singular-value distributions of the Muon update matrix at different training steps. Although the Newton-Schulz iteration closely approximates the "all-ones" singular spectrum, many singular values still deviate noticeably from one. Meanwhile, as shown in Figure 1a, Muon_NS outperforms Muon_SVD on both LLaMA-60M and LLaMA-135M. This suggests that keeping the same weight for all singular-vector directions throughout training may not be the most helpful. In particular, directions associated with smaller singular values are often more noise-dominated (Sharma et al., 2023; Wang et al., 2023; Chen et al., 2024; Liu et al., 2025d; Defilippis et al., 2025); enforcing exactly equal weights across all directions yields a light-tailed update, which can make late-stage training more sensitive to noise and limit model capacity. In contrast, due to the polynomial used in the Newton-Schulz steps, Muon_NS implicitly assigns smaller weights to noise-dominated singular-vector directions, which improves performance. This observation also supports the motivation for making Muon updates more heavy-tailed.

2.3 From Updates to Weights: Muon Produces Lighter-Tailed Weight Spectra

In this section, we show that Muon's orthogonalization update makes weight matrices ESDs less

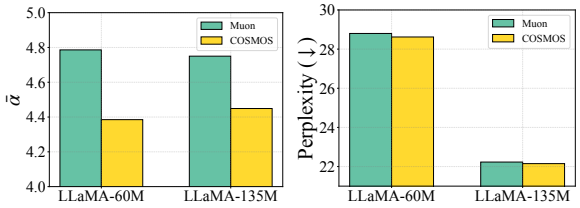


(a) Muon_NS vs Muon_SVD (b) Muon_NS Update Spectrum (c) Muon_SVD Update Spectrum

Figure 1: **Muon_NS vs. Muon_SVD on C4 dataset.** (a) Validation perplexity for LLaMA-60M/135M trained: Muon_NS consistently achieves lower perplexity than Muon_SVD. Both Learning rates for 60M is 0.03 and for 135M is 0.02. (b)(c) Spectra of update matrices at steps $1/9000/19000$ shown in different colors: Muon_SVD enforces an exactly "all-ones" spectrum in the update matrices; Muon_NS stays close to one but retains noticeable deviations, implicitly down-weighting noise-dominated singular-vector directions and correlating with improved performance.

heavy-tailed, which, based on HT-SR theory, may limit the model performance.

We train LLaMA-60M and LLaMA-135M on the C4 dataset using Muon_NS and COSMOS, a variant of Muon (Liu et al., 2025b). In Figure 2, we visualize the average fitted PL exponent $\bar{\alpha}$ for the trained LLaMA models across layers under the two optimizers. We find that $\bar{\alpha}$ across layers is higher for Muon, indicating that Muon makes the weight ESD less heavy-tailed. We further observe that Muon yields higher perplexity (PPL) than COSMOS on both models. Based on HT-SR theory, this indicates that Muon's orthogonalized update rule may limit the model's final quality, which motivates us to make Muon update more heavy-tailed, leading to more heavy-tailed weight matrices.



(a) Average PL $\bar{\alpha}$ (b) Perplexity

Figure 2: (a)Average PL $\bar{\alpha}$ of weight ESDs for LLaMA-60M and LLaMA-135M trained on C4 with Muon and COSMOS. Muon yields a higher mean $\bar{\alpha}$, indicating less heavy-tailed spectra than COSMOS; (b) COSMOS outperforms Muon for LLaMA-60M and LLaMA-135M models on C4 datatset.

3 Methodology

In this section, we introduce our method HTMuon. Our design goal is to preserve Muon's ability to capture parameter interdependencies while making its updates more heavy-tailed, which in turn leads to more heavy-tailed weight ESDs. We put HTMuon in Algorithm 1. HTMuon is similar to Muon, and the only difference is that we consider a different power transform of the momentum matrix's singular p in line 7. It is easy to know that when $p = 1$, HTMuon reduces to SGDM. SGDM can be viewed as a vector-based optimizer and has limited ability to capture interdependencies among parameters. When $p = 0$, HTMuon reduces to Muon; as discussed in Section 2.3, this update makes weight-matrix ESDs less heavy-tailed.

We therefore propose HTMuon with $p \in (0, 1)$. We believe HTMuon helps mitigate the design limitations of Muon and SGDM: On the one hand, in this regime, HTMuon remains a *matrix-based* optimizer and thus retains the ability to capture parameter interdependencies. On the other hand, HTMuon updates are more heavy-tailed than Muon updates, and we illustrate this with a simple example: suppose that during training, the singular values of the momentum matrix $\boldsymbol{M}_t$ follow $s_k = s_1 k^{-s}$, where s_1 is the largest singular value of $\boldsymbol{M}_t$. By Lemma 3.1, the PL exponent α of the HTMuon update matrix $\boldsymbol{O}_t$ is $1 + \frac{1}{2sp}$, it is easy to find that when $p \to 0$, the α will increase, this means HTMuon update is more heavy-tailed than Muon's, which will in turn lead to more heavy-tailed weight ESDs.

Lemma 3.1 (Proof in Appendix B.3) *Suppose the singular values of matrix $\boldsymbol{W} \in \mathbb{R}^{n\times m}$ follow $s_k = s_1 k^{-s}, 1 \le k \le m$, we have PL exponent α of $\boldsymbol{W}$ satisfies $\alpha = 1 + \frac{1}{2s}$.*

4 Experiments

In this section, we evaluate HTMuon on various pretraining tasks, and compare it with state-of-the-art pretraining optimizers, including Adam (Kingma,

Algorithm 1 HTMuon

Input: Initial weights $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, loss function L, learning rate η, momentum parameter β, weight decay λ, power $p \in (0, 1)$.
Initialize $\mathbf{M}_0 \in \mathbb{R}^{m\times n} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  $\mathbf{G}_t \leftarrow \nabla_{\mathbf{W}} L(\mathbf{W}_t)$
  $\mathbf{M}_t \leftarrow \beta\mathbf{M}_{t-1} + (1-\beta)\mathbf{G}_t$
  $\boldsymbol{U}_t, \boldsymbol{\Sigma}_t, \boldsymbol{V}_t^\top \leftarrow \mathrm{SVD}(\mathbf{M}_t)$
  $\boldsymbol{O}_t \leftarrow \boldsymbol{U}_t\boldsymbol{\Sigma}_t^p\boldsymbol{V}_t^\top$
  $s \leftarrow \sqrt{\max\left(1, \frac{m}{n}\right)}$
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\lambda\mathbf{W}_t - \eta s\boldsymbol{O}_t$
end for

2014), AdamW (Loshchilov and Hutter, 2017), SGDM (Sutskever et al., 2013), NorMuon (Li et al., 2025), AdaMuon (Si et al., 2025), Cautious (Liang et al., 2024), GaLore (Zhao et al., 2024), Sophia (Liu et al., 2024a), Mars (Yuan et al., 2024), SOAP (Vyas et al., 2024), COSMOS (Liu et al., 2025b). We provide detailed definitions of these optimizers in Appendix E. Unless otherwise specified, all Muon results in our experiments use the Muon_NS implementation. We demonstrate that HTMuon achieves superior performance than these optimizers across all datasets. In addition, we show that HTMuon can be used as an add-on method to combine with existing Muon variant optimizers to achieve further improvements. Furthermore, to reduce the overhead of SVD in HTMuon, we design two acceleration schemes that significantly cut the computational cost while still outperforming Muon. To ensure a fair comparison, we reproduce all results of the baseline methods with the codebases provided in previous papers.

4.1 Experimental Setup

Tasks. We conduct experiments on various pretraining tasks, including: 1) **LLM Pretraining**, in which we pretrain LLaMA-family models on the C4 dataset (Raffel et al., 2020), GPT-2 family models on the OpenWebText dataset (Gokaslan and Cohen, 2019). 2) **Image Classification**, where we train ResNet models on the CIFAR-100 and CIFAR-10 datasets (Krizhevsky and Hinton, 2009) and train ViT-tiny on the ImageNet-1K dataset (Deng et al., 2009).

Models. For LLM pretraining, we evaluate HTMuon on LLaMA-60M, LLaMA-135M, LLaMA-350M, LLaMA-1B (Touvron et al., 2023) and GPT-2 small (125M) (Radford et al., 2019). For image classification, we use ResNet18, ResNet50 (He et al., 2016) and ViT-tiny (Dosovitskiy, 2020).

Table 1: Comparison with dominant Pretraining Optimizers on LLaMA Models of Varying Sizes on the C4 dataset. Lower perplexity indicates better performance. Detailed hyperparameters are provides in Table 18, 19 and 20 in Appendix D.

	LLaMa-60M	LLaMa-135M	LLaMa-350M
Adam	32.21	23.01	17.11
AdamW	31.85	23.33	16.96
Muon	28.80	22.23	16.81
HTMuon	**27.88**	**21.25**	**16.79**

Unless otherwise specified, we use $p = 0.125$ for HTMuon throughout. We discuss the specific choice of p in Section 4.7. All baseline methods and HTMuon are carefully tuned for a fair comparison. For detailed hyperparameter settings, please refer to Appendix D.

4.2 LLM Pretraining

In this section, we evaluate HTMuon against several baseline optimizers on C4 and OpenWebText datasets.

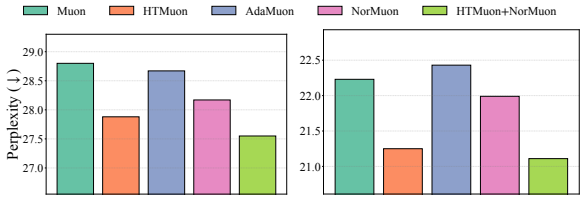
HTMuon consistently outperforms Muon. In Table 1, We find that HTMuon consistently outperforms Muon across three model scales (60M, 135M, and 350M) on C4 dataset. In particular, on LLaMA-60M, HTMuon achieves a PPL that is 0.92 lower than Muon and 4.33 lower than Adam; on LLaMA-135M, HTMuon is 0.98 lower than Muon and 2.08 lower than AdamW. In Table 2, we show that on GPT-2 small, HTMuon achieves a PPL that is 0.26 lower than Muon and 2.99 lower than AdamW. In Figure 8 in Appendix C, we provide the training loss curves of HTMuon and Muon. These results suggest that HTMuon's update rule effectively mitigates the limitations of Muon updates.

HTMuon consistently outperforms other variants of Muon. In Figure 3, we find that HTMuon consistently outperforms Muon variant optimizers like AdaMuon and NorMuon. Specifically, on LLaMA-60M, our PPL is 0.29 lower than that of NorMuon, the strongest baseline after HTMuon; on LLaMA-135M, we outperform NorMuon by 0.74 PPL and AdaMuon by 1.18 PPL. Moreover, we observe that **combining HTMuon with NorMuon yields further gains**, reducing PPL by an additional 0.33 on LLaMA-60M and 0.14 on LLaMA-135M.

HTMuon consistently outperforms state-of-the-art pretraining optimizers. In Figure 4, we compare HTMuon with state-of-the-art optimizers on

Table 2: Comparison with dominant Pretraining Optimizers on GPT-2 small on the OpenWebText dataset. Lower perplexity indicates better performance. Hyperparameter settings are provided in Appendix D.

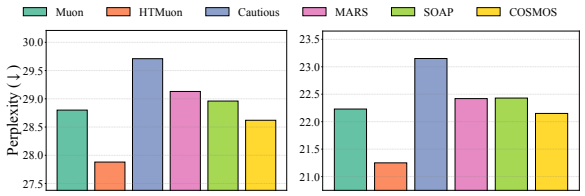
	AdamW	Muon	HTMuon
GPT-2 small	25.19	22.46	**22.20**



(a) LLaMA-60M (b) LLaMA-135M

Figure 3: Comparison with Muon variant optimizers on LLaMA-60M and 135M on C4 dataset. All optimizers are carefully tuned via grid search; detailed results and hyperparameter settings are provided in Table 7 and 19 in Appendix C and D.

LLaMA-60M and LLaMA-135M. HTMuon consistently outperforms all competing methods at both scales. For example, compared to COSMOS (the second-best method), HTMuon effectively reduces PPL by 1.07 on LLaMA-60M and by 1.04 on LLaMA-135M. In Appendix C, we include more recent optimizers, including GaLore and Sophia.



(a) LLaMA-60M (b) LLaMA-135M

Figure 4: Comparison with state-of-the-art pretraining optimizers on LLaMA-60M and 135M on C4 dataset. All optimizers are carefully tuned via grid search; detailed results and hyperparameter settings are provided in Table 7 and 19 in Appendix C and D.

4.3 Image Classification

In this section, we compare HTMuon against several baseline optimizers on CIFAR and ImageNet-1K datasets.

In Table 3, we find that, when training ResNet18 and ResNet50 on CIFAR-10 and CIFAR-100, HTMuon consistently achieves higher accuracy than SGDM, Muon, and NorMuon. For example, relative to Muon, HTMuon improves accuracy by up to 0.31 % on CIFAR-100 and up to 0.24 % on CIFAR-10. Moreover, on CIFAR-100, combining HTMuon with NorMuon further improves the accuracy of both ResNet18 and ResNet50. In Table 4, we show that HTMuon also outperforms Muon and Adam when training ViT-tiny on ImageNet-1K.

Table 3: Results on CIFAR-100 and CIFAR-10 dataset. Higher accuracy indicates better performance.

Optimizer	CIFAR-100		CIFAR-10	
	ResNet18	ResNet50	ResNet18	ResNet50
SGDM	77.82	78.57	95.15	95.16
Muon	77.95	79.85	95.39	95.97
NorMuon	77.82	79.78	95.57	96.03
HTMuon	78.24	80.16	**95.63**	**96.13**
HTMuon + NorMuon	**78.51**	**80.22**	95.40	96.04

Table 4: Results on ImageNet-1K. Hyperparameter settings are provided in Appendix D.

	Adam	Muon	HTMuon
ViT-Tiny	67.67	71.02	**71.16**

4.4 More Efficient Implementations of HTMuon

In this section, to reduce the overhead introduced by SVD in HTMuon, we consider two acceleration strategies:

(i) Apply HTMuon every fixed number of training steps and use Muon otherwise. In Figure 5a and 5c, we evaluate applying HTMuon on LLaMA-60M and LLaMA-135M every 5, 10, and 25 steps. Compared to using HTMuon at every step, this strategy greatly reduces runtime overhead, making it competitive with Muon, while still outperforming Muon. We also observe that smaller intervals lead to better performance, further supporting the effectiveness of our optimizer.

Based on these findings, we train LLaMA-1B on C4 using HTMuon with interval $= 5$. As shown in Table 5, HTMuon still outperforms Muon, highlighting its potential for large-scale training.

(ii) Replace the SVD with a numerical iterative method for faster computation. In Algorithm 1, We note that the HTMuon update $\boldsymbol{O}_t = \boldsymbol{U}_t\boldsymbol{\Sigma}_t^p\boldsymbol{V}_t^\top$ admits the factorization $\boldsymbol{O}_t = (\boldsymbol{U}_t\boldsymbol{V}_t^\top)(\boldsymbol{V}_t\boldsymbol{\Sigma}_t^p\boldsymbol{V}_t^\top)$. For the $\boldsymbol{U}_t\boldsymbol{V}_t^\top$ part, this is Muon's update; we can consider Newton-Schulz with 5 steps. To compute $\boldsymbol{V}_t\boldsymbol{\Sigma}_t^p\boldsymbol{V}_t^\top$ efficiently, we apply the NS_root routine to $\boldsymbol{M}_t^\top\boldsymbol{M}_t$ in Algorithm 6, since $\boldsymbol{M}_t^\top\boldsymbol{M}_t = \boldsymbol{V}_t\boldsymbol{\Sigma}_t^2\boldsymbol{V}_t^\top$, we have $\boldsymbol{V}_t\boldsymbol{\Sigma}_t^p\boldsymbol{V}_t^\top = (\boldsymbol{M}_t^\top\boldsymbol{M}_t)^{\frac{p}{2}}$. Accordingly, NS_root uses Newton-Schulz iterations to approximate matrix square roots and applies successive square-root operations for $[1 - \log_2 p]$ rounds. Combining these two, we obtain HTMuon_NS in Algorithm 5.

Table 5: We train LLaMA-1B with interval= 5 on C4. Hyperparameter settings are provided in Table 20 in Appendix D.

	Adam	AdamW	Muon	HTMuon
LLaMA-1B	15.22	15.11	14.33	**14.17**

In Figure 5b and 5d, comparing HTMuon_NS with HTMuon with interval= 1, we find HTMuon_NS substantially reduces runtime overhead, while only slightly increases PPL. We also apply HTMuon_NS on LLaMA-60M and LLaMA-135M every 5, 10, and 25 steps. We find that this further reduces runtime overhead while still maintaining better performance than Muon, for example, HTMuon_NS (interval $= 5$) achieves 0.59 s/step compared to Muon's 0.51 s/step in Table 9(which only introduces a minor increase in overhead), Table 8 shows that HTMuon_NS (interval $= 5$) still outperforms Muon by 0.38 PPL.

We also evaluate HTMuon + NorMuon with different intervals. As observed earlier, using larger intervals reduces runtime overhead while still outperforming NorMuon. Please refer to Figure 9 in Appendix C.

Besides reporting per-step runtime overhead, we also report the wall-clock time for Muon, HTMuon and HTMuon_NS with intervals. We summarize the PPL and wall-clock time of Muon, HTMuon, HTMuon_NS, HTMuon (Interval $= 5$), and HTMuon_NS (Interval $= 5$) in the Table 10 in Appendix C. We also give the FLOPs analysis for Muon and HTMuon_NS in Appendix C. Please refer to more discussions in Appendix C.3.

4.5 Weight Spectrum Analysis

HTMuon successfully produces more heavy-tailed weight matrices. In Figure 6, we fit the layer-wise PL α for models trained with Muon and HTMuon. Across both LLaMA and ResNet models, we find that HTMuon yields smaller α than Muon for most layers, leading to a lower average $\bar{\alpha}$. Meanwhile, HTMuon achieves better performance, consistent with the HT-SR observation that better-trained models tend to have lower α. For more PL α plots, please refer to Figure 10a and 10b in Appendix C.

Additional generalization metrics. In Figure 11 in Appendix C, we visualize the layer-wise spectral norm and Frobenius norm for LLaMA models trained with Muon and HTMuon. We find that, for these models, HTMuon consistently yields smaller norms than Muon. This is consistent with prior findings (Miyato et al., 2018) that smaller spectral and Frobenius norms are often associated with better generalization.

4.6 Downstream Tasks

To further evaluate HTMuon's applicability to broader NLP scenarios, in this section, we evaluate LLaMA-1B model trained with HTMuon on 7 commonsense tasks using the lm_eval_harness framework. For all tasks we use the default prompt and perform zero-shot evaluation.

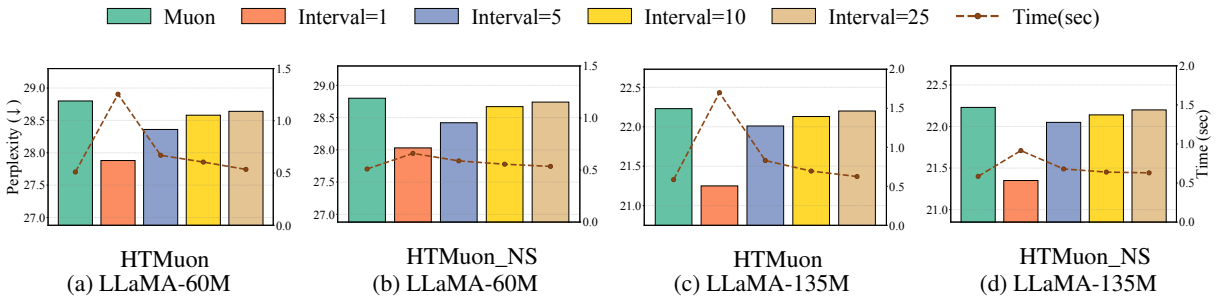
As shown in the Table 6, HTMuon achieves the best test accuracy for 4 out of 7 commonsense tasks, and achieves the best average score across 7 tasks, significantly outperforming the second best optimizer (Muon) by 1.05. This demonstrates the effectiveness of HTMuon on wider NLP scenarios.

4.7 Ablation Study

Varying different p. Since p is an additional hyperparameter introduced by HTMuon, we perform a grid search over p on LLaMA models in Figure 7. We find that $p = 0.125$ is a good choice, which is why we use $p = 0.125$ in most experiments. In Appendix C.4.2, we discuss more about the hyperparameter sensitivity for p, we show that although introducing p adds a tuning dimension and add hyperparameter search burden, $p = 0.125$ generalizes well across tasks and architectures and often can be optimal or near-optimal across tasks and architectures. We suggest that $p = 0.125$ can serve as a recommended default for other tasks.

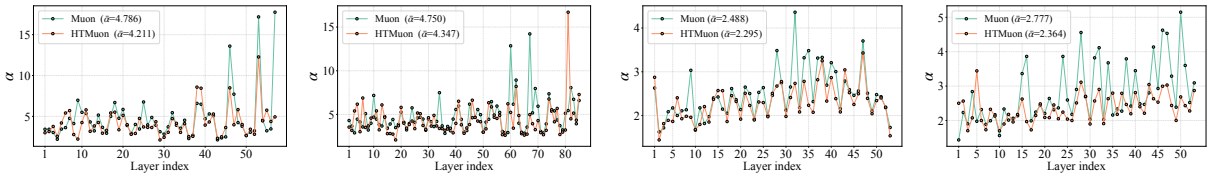
Varying learning rates. In Figure 12, we visualize performance across learning-rate grids for LLaMA models on C4 and ResNet models on CIFAR datasets. We find that, over most learning rates, HTMuon consistently outperforms Muon and NorMuon, demonstrating the stability of HTMuon. For more details, please refer to Table 11, 12 and 13 in Appendix C.

Varying different ways to make Muon update more heavy-tailed. To better prove making Muon update more heavy-tailed can improve performance, we design another optimizer HTMuon_HT in Algorithm 7. Specifically, at each step, HTMuon_HT directly replaces the singular values of the momentum matrix with a heavy-tailed sequence $\{i^{-\alpha}\}_{i=1}^{m}$. As shown in Table 17, although HTMuon_HT is not as strong as HTMuon, it still outperforms Muon on LLaMA models. This shows that merely inducing a heavy-tailed spectrum already yields non-trival gains, and our algorithm HTMuon further show that heavy tail spectral correction on top of preserved



(a) LLaMA-60M HTMuon (b) LLaMA-60M HTMuon_NS (c) LLaMA-135M HTMuon (d) LLaMA-135M HTMuon_NS

Figure 5: we evaluate applying HTMuon and HTMuon_NS on LLaMA-60M and LLaMA-135M every 1, 5, 10, and 25 steps. We report the average per-step runtime overhead for all methods. Detailed results and hyperparameter settings are provided in Table 8 and 9 in Appendix C.

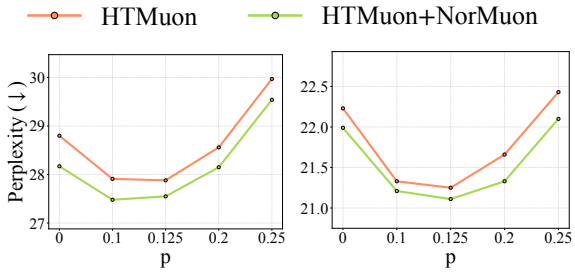


(a) LLaMA-60M on C4 (b) LLaMA-135M on C4 (c) ResNet50 on CIFAR-100 (d) ResNet50 on CIFAR-10

Figure 6: Layer-wise PL α for LLaMA and ResNet model weights trained with Muon and HTMuon. All models used for visualization are trained using each optimizer's best-performing hyperparameter configuration. For hyperparameter configurations, please refer to Appendix D.

Table 6: Zero-shot evaluation results (↑) on seven commonsense reasoning benchmarks for the LLaMA-1B model pretrained with different methods.

Optimizer	ARC-c	ARC-e	PIQA	Hellaswag	OBQA	Winogrande	BOOLQ	Avg.
Adam	21.93	31.27	65.18	27.49	17.40	**53.83**	62.17	39.90
AdamW	21.16	32.37	64.25	27.29	**17.80**	52.49	**62.23**	39.66
Muon	20.56	32.66	65.13	31.53	17.00	51.22	62.14	40.03
HTMuon	**22.18**	**33.16**	**66.59**	**33.76**	17.20	52.72	61.93	**41.08**



(a) LLaMA-60M (b) LLaMA-135M

Figure 7: We conduct grid search on p. We find $p = 0.125$ is a strong choice. Note that $p = 0$ reduces to Muon. Detailed results provided in Table 14 in Appendix D.

spectral information is more effective, we put more discussion about the performance improvement attribution in Appendix C.4.3. In Figure 10c and 10d, we visualize the layer-wise PL α for HTMuon_HT, and as intended, it yields lower α than Muon across layers.

5 Theoretical Analysis

In this section, we provide theoretical understandings of HTMuon.

Notation. For a vector $\boldsymbol{v}$, we denote its l_2 norm as $\|\boldsymbol{v}\|_2$. For a matrix $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m\times n}$, we denote its spectral norm as $\|\boldsymbol{A}\|$, Frobenius norm as $\|\boldsymbol{A}\|_{\mathrm{F}}$, nuclear norm as $\|\boldsymbol{A}\|_*$, where $\|\boldsymbol{A}\|_* = \sum_{i=1}^{r} \sigma_i$. Here, $\{\sigma_i\}_{i=1}^{r}$ are singular values. For $p > 1$, we denote Schatten$-p$ norm as $\|\boldsymbol{A}\|_p$, where $\|\boldsymbol{A}\|_p = (\sum_{i=1}^{r} \sigma_i^p)^{\frac{1}{p}}$. We use $\langle \boldsymbol{A}, \boldsymbol{B}\rangle$ to denote the inner product between $\boldsymbol{A}$ and $\boldsymbol{B}$, i.e., $\langle \boldsymbol{A}, \boldsymbol{B}\rangle = \mathrm{Tr}\left(\boldsymbol{A}^\top \boldsymbol{B}\right)$.

5.1 Update Under the Schatten$-q$ Norm Constraint

We are here to provide a view of Schatten norm constrained optimization for HTMuon. To determine the descent direction at step t, we formulate the

update as the following constrained minimization problem:

$$\min_{\Delta \boldsymbol{W} \in \mathbb{R}^{m \times n}} \quad f(\boldsymbol{W}_t + \Delta \boldsymbol{W}) \qquad (1)$$
$$\text{subject to} \quad \|\Delta \boldsymbol{W}\|_q \leq \delta$$

where $\|\cdot\|_q$ denotes the Schatten$-q$ norm. When we employ the first-order approximation for our optimization objective

$$f(\boldsymbol{W}_t + \Delta \boldsymbol{W}) \approx f(\boldsymbol{W}_t) + \mathrm{Tr}(\nabla f(\boldsymbol{W}_t)^\top \Delta \boldsymbol{W}) \qquad (2)$$

It is straightforward to derive that optimizing this approximation is equivalent to the following objective:

$$\max_{\Delta \boldsymbol{W}} \quad \mathrm{Tr}(\widetilde{\boldsymbol{G}_t}^\top \Delta \boldsymbol{W}) \qquad (3)$$
$$\text{subject to} \quad \|\Delta \boldsymbol{W}\|_q \leq \delta,$$

where $\widetilde{\boldsymbol{G}_t} = -\nabla f(\boldsymbol{W}_t)$ denotes the negative gradient.

Theorem 5.1 *Let $\widetilde{\boldsymbol{G}_t} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be the* SVD *of the negative gradient $\widetilde{\boldsymbol{G}_t} = -\nabla f(\boldsymbol{W}_t)$, where $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$. Let p be the conjugate exponent of q, satisfying $1/p' + 1/q = 1$. The explicit solution $\Delta \boldsymbol{W}^*$ to the optimization problem 3 is given by:*

$$\Delta \boldsymbol{W}^* = \delta \frac{1}{\|\widetilde{\boldsymbol{G}_t}\|_{p'}^{p'/q}} \boldsymbol{U}\boldsymbol{\Sigma}^{p'-1}\boldsymbol{V}^\top, \qquad (4)$$

where $\boldsymbol{\Sigma}^{p'-1} = \mathrm{diag}(\sigma_1^{p'-1}, \ldots, \sigma_r^{p'-1})$.

We put proof in Appendix B.1. Based on Theorem 5.1, when $q \in (2, \infty), p' \in (1, 2)$, we can assume $p = p' - 1$ and find the $\Delta \boldsymbol{W}^*$ is exactly the update by HTMuon, which means HTMuon is equivalent to steepest descent under a Schatten$-q$ norm constraint. It is well known that Muon can be viewed as the steepest descent under the Schatten$-\infty$ norm constraint (Bernstein and Newhouse, 2024; Pethick et al., 2025), so HTMuon generalizes Muon's equivalence to steepest descent under a Schatten$-\infty$ norm constraint and this will also benefit training (Davis and Drusvyatskiy, 2025).

5.2 Convergence Analysis

Here we give the convergence analysis under a smooth non-convex setting for finding ϵ-stationary points, i.e. $\|\nabla f(\boldsymbol{W})\|_* \leq \epsilon$. We consider the following stochastic optimization problem:

$$\min_{\boldsymbol{W} \in \mathbb{R}^{m \times n}} f(\boldsymbol{W}) = \mathbb{E}_{\boldsymbol{\xi}}[f(\boldsymbol{W}; \boldsymbol{\xi})], \qquad (5)$$

where $\boldsymbol{\xi}$ is stochastic noise. We denote $f^* = \inf_{\boldsymbol{W}} f(\boldsymbol{W})$, $r = \min\{m, n\}$. We assume $f^* > -\infty$.

For ease of theoretical analysis, we rewrite HTMuon in a stochastic setting and present it in Algorithm 2.

We assume f is an L Frobenius norm Lipchitz smooth function. We assume that $\nabla f(\boldsymbol{W}; \xi)$ is an unbiased stochastic estimator of the true gradient $\nabla f(\boldsymbol{W})$ and has a bounded variance. We also assume that $\boldsymbol{M}_t$'s singular values are bounded by l. We give assumptions in Appendix B.2, some assumptions and definitions are adapted from Shen et al. (2025).

Theorem 5.2 (HTMuon) *Under Assumptions B.3, B.4 and B.5, if we apply* HTMuon *in Algorithm 2 with adaptive learning rate $\eta_t = \frac{\langle \boldsymbol{M}_t, \rho(\boldsymbol{M}_t) \rangle}{L\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2}$, $\rho(\boldsymbol{M}_t) = \boldsymbol{U}_t \boldsymbol{\Sigma}_t^p \boldsymbol{V}_t^\top$, $\eta = \max\{\eta_t\}_{t=1}^T$, batch size $B = T$ and $\beta = \frac{1}{\sqrt{T}}$, $\Delta = f(\boldsymbol{W}_0) - f^*$, we have*

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\big[\|\nabla f(\boldsymbol{W}_t)\|_{1+p}^2\big] \leq \frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\left(\frac{1-\varepsilon'}{L} + \frac{1}{L\varepsilon'}\right)$$
$$\big[\frac{2\sigma}{T} + \frac{4\sigma^2}{T^3} + \frac{4rl^{\frac{2}{q}}L^2\eta^2}{T}\big] + \frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\frac{\Delta}{T},$$

where $s = \frac{1+p}{1-p}$ and $0 < \varepsilon' < 1$.

We put proof in Appendix B.2. Based on Theorem 5.2 and Definition B.6, when $B = T$, the sample complexity upper bound is $O(\epsilon^{-4})$, which matches Muon's sample complexity in Appendix Theorem B.8 and SGDM's sample complexity in Garrigos and Gower (2023); Arjevani et al. (2023).

6 Conclusion

In this work, motivated by HT-SR theory, we introduce HTMuon optimizer. HTMuon maintains Muon's capacity to model parameter interdependencies, while yielding more heavy-tailed updates and promoting more heavy-tailed weight spectra. It delivers consistent gains in accuracy and training stability on LLM pretraining and image classification, and can be seamlessly integrated into existing Muon variants. To reduce the SVD overhead in HTMuon, we introduce two acceleration implementations that further lower runtime cost while still outperforming Muon. Finally, we establish a theoretical connection to steepest descent under a Schatten-q norm constraint and provide convergence analysis for smooth non-convex settings.

Limitations

Despite achieving improvements in LLM pretraining and image classification, HTMuon has some potential limitations. Due to limited computational resources, we have not evaluated HTMuon on larger models than 1B or larger-scale datasets; we leave this to future work. In addition, HTMuon and HTMuon_NS incur slightly higher per-step runtime overhead than Muon. While using HTMuon with larger update intervals can reduce this cost, achieving stronger performance typically requires more frequent HTMuon updates, which increases the overall runtime. Exploring better numerical methods to accelerate HTMuon is also part of future work.

Acknowledgments

We thank our colleagues and funding agencies. This work is supported by the DARPA AIQ and DARPA DIAL programs, the U.S. Department of Energy under Award Number DE-SC0025584, Dartmouth College, and Lambda AI.

References

Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, and Blake Woodworth. 2023. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, 199(1):165–214.

Jeremy Bernstein and Laker Newhouse. 2024. Modular duality in deep learning. *arXiv preprint arXiv:2410.21265*.

Da Chang, Yongxiang Liu, and Ganzhao Yuan. 2025. On the convergence of muon and beyond. *arXiv preprint arXiv:2509.15816*.

Lei Chen, Joan Bruna, and Alberto Bietti. 2024. Distributional associations vs in-context reasoning: A study of feed-forward and attention layers. *arXiv preprint arXiv:2406.03068*.

Lizhang Chen, Jonathan Li, and Qiang Liu. 2025. Muon optimizes under spectral norm constraints. *arXiv preprint arXiv:2506.15054*.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V Le. 2023. Symbolic discovery of optimization algorithms. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yatin Dandi, Luca Pesce, Hugo Cui, Florent Krzakala, Yue M Lu, and Bruno Loureiro. 2024. A random matrix theory perspective on the spectrum of learned features and asymptotic generalization capabilities. *arXiv preprint arXiv:2410.18938*.

Damek Davis and Dmitriy Drusvyatskiy. 2025. When do spectral gradient updates help in deep learning? *arXiv preprint arXiv:2512.04299*.

Leonardo Defilippis, Yizhou Xu, Julius Girardin, Emanuele Troiani, Vittorio Erba, Lenka Zdeborová, Bruno Loureiro, and Florent Krzakala. 2025. Scaling laws and spectra of shallow neural networks in the feature learning regime. *arXiv preprint arXiv:2509.24882*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Shenyang Deng, Boyao Liao, Zhuoli Ouyang, Tianyu Pang, Minhak Song, and Yaoqing Yang. 2026. Suspicious alignment of sgd: A fine-grained step size condition analysis. *arXiv preprint arXiv:2601.11789*.

Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*.

Guillaume Garrigos and Robert M Gower. 2023. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus.

Dirk Groeneveld, Iz Beltagy, Evan Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, and 24 others. 2024. Olmo: Accelerating the science of language models. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 15789–15809.

Vineet Gupta, Tomer Koren, and Yoram Singer. 2018. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR.

Di He, Songjun Tu, Ajay Jaiswal, Li Shen, Ganzhao Yuan, Shiwei Liu, and Lu Yin. 2025. Alphadecay: Module-wise weight decay for heavy-tailed balancing in llms. *arXiv preprint arXiv:2506.14562*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Nicholas J Higham and Robert S Schreiber. 1990. Fast polar decomposition of an arbitrary matrix. *SIAM Journal on Scientific and Statistical Computing*, 11(4):648–655.

Liam Hodgkinson, Umut Simsekli, Rajiv Khanna, and Michael Mahoney. 2022. Generalization bounds using lower tail exponents in stochastic optimizers. In *International Conference on Machine Learning*, pages 8774–8795. PMLR.

Liam Hodgkinson, Zhichao Wang, and Michael W Mahoney. 2025. Models of heavy-tailed mechanistic universality. *arXiv preprint arXiv:2506.03470*.

Yuanzhe Hu, Kinshuk Goel, Vlad Killiakov, and Yaoqing Yang. 2025. Eigenspectrum analysis of neural networks without aspect ratio bias. In *Forty-second International Conference on Machine Learning*.

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

Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J Kusner. 2022. When do flat minima optimizers work? *Advances in Neural Information Processing Systems*, 35:16577–16595.

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

Vignesh Kothapalli, Tianyu Pang, Shenyang Deng, Zongmin Liu, and Yaoqing Yang. 2025. From spikes to heavy tails: Unveiling the spectral evolution of neural networks. *Transactions on Machine Learning Research*.

Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. 2025. Normuon: Making muon more efficient and scalable. *arXiv preprint arXiv:2510.05491*.

Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. 2024. Cautious optimizers: Improving training with one line of code. *arXiv preprint arXiv:2411.16085*.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. 2024a. Sophia: A scalable stochastic second-order optimizer for language model pretraining. *Preprint*, arXiv:2305.14342.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, and 9 others. 2025a. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*.

Liming Liu, Zhenghao Xu, Zixuan Zhang, Hao Kang, Zichong Li, Chen Liang, Weizhu Chen, and Tuo Zhao. 2025b. Cosmos: A hybrid adaptive optimizer for memory-efficient training of llms. *arXiv preprint arXiv:2502.17410*.

Xuyuan Liu, Lei Hsiung, Yaoqing Yang, and Yujun Yan. 2025c. Spectral insights into data-oblivious critical layers in large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 4860–4877, Vienna, Austria. Association for Computational Linguistics.

Zihang Liu, Yuanzhe Hu, Tianyu Pang, Yefan Zhou, Pu Ren, and Yaoqing Yang. 2024b. Model balancing helps low-data training and fine-tuning. *arXiv preprint arXiv:2410.12178*.

Zihang Liu, Tianyu Pang, Oleg Balabanov, Chaoqun Yang, Tianjin Huang, Lu Yin, Yaoqing Yang, and Shiwei Liu. 2025d. Lift the veil for the truth: Principal weights emerge after rank reduction for reasoning-focused supervised fine-tuning. *arXiv preprint arXiv:2506.00772*.

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

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W Mahoney, and Yaoqing Yang. 2024. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *Advances in neural information processing systems*, 37:9117–9152.

Jianhao Ma, Yu Huang, Yuejie Chi, and Yuxin Chen. 2026. Preconditioning benefits of spectral orthogonalization in muon. *Preprint*, arXiv:2601.13474.

Charles H Martin and Christopher Hinrichs. 2025. Setol: A semi-empirical theory of (deep) learning. *arXiv preprint arXiv:2507.17912*.

Charles H Martin and Michael W Mahoney. 2021. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165):1–73.

Charles H Martin, Tongsu Peng, and Michael W Mahoney. 2021. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(1):4122.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, and 1 others. 2024. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. 2025. Benchmarking optimizers for large language model pretraining. *arXiv preprint arXiv:2509.01440*.

Ishaan Shah, Anthony M. Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, Khoi Nguyen, Kurt Smith, Michael Callahan, Michael Pust, Mohit Parmar, Peter Rushton, Platon Mazarakis, Ritvik Kapila, Saurabh Srivastava, and 4 others. 2025. Practical efficiency of muon for pretraining. *arXiv preprint arXiv:2505.02222*.

Pratyusha Sharma, Jordan T Ash, and Dipendra Misra. 2023. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. *arXiv preprint arXiv:2312.13558*.

Wei Shen, Ruichuan Huang, Minhui Huang, Cong Shen, and Jiawei Zhang. 2025. On the convergence analysis of muon. *arXiv preprint arXiv:2505.23737*.

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

James B Simon, Madeline Dickens, Dhruva Karkada, and Michael R DeWeese. 2023. The eigenlearning framework: A conservation law perspective on kernel ridge regression and wide neural networks. *Transactions on Machine Learning Research*.

Umut Simsekli, Ozan Sener, George Deligiannidis, and Murat A Erdogdu. 2020. Hausdorff dimension, heavy tails, and generalization in neural networks. *Advances in Neural Information Processing Systems*, 33:5138–5151.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. pmlr.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. 2024. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*.

Jinbo Wang, Mingze Wang, Zhanpeng Zhou, Junchi Yan, and Lei Wu. 2025a. The sharpness disparity principle in transformers for accelerating language model pre-training. *arXiv preprint arXiv:2502.19002*.

Shuche Wang, Fengzhuo Zhang, Jiaxiang Li, Cunxiao Du, Chao Du, Tianyu Pang, Zhuoran Yang, Mingyi Hong, and Vincent YF Tan. 2025b. Muon outperforms adam in tail-end associative memory learning. *arXiv preprint arXiv:2509.26030*.

Zhichao Wang, Andrew Engel, Anand D Sarwate, Ioana Dumitriu, and Tony Chiang. 2023. Spectral evolution and invariance in linear-width neural networks. *Advances in neural information processing systems*, 36:20695–20728.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. 2025. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*.

Yaoqing Yang, Ryan Theisen, Liam Hodgkinson, Joseph E Gonzalez, Kannan Ramchandran, Charles H Martin, and Michael W Mahoney. 2023. Test accuracy vs. generalization gap: Model selection in nlp without accessing training or testing data. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3011–3021.

Huizhuo Yuan, Yifeng Liu, Shuang Wu, Xun Zhou, and Quanquan Gu. 2024. Mars: Unleashing the power of variance reduction for training large models. *arXiv preprint arXiv:2411.10438*.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, and 1 others. 2025. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*.

Chunhui Zhang, Yiren Jian, Zhongyu Ouyang, and Soroush Vosoughi. 2024a. Working memory identifies reasoning limits in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16896–16922.

Chunhui Zhang, Yiren Jian, Zhongyu Ouyang, and Soroush Vosoughi. 2025. Pretrained image-text models are secretly video captioners. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*.

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

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.

Beitong Zhou, Jun Liu, Weigao Sun, Ruijuan Chen, Claire J Tomlin, and Ye Yuan. 2020. pbsgd: Powered stochastic gradient descent methods for accelerated non-convex optimization. In *IJCAI*, pages 3258–3266.

Yefan Zhou, Tianyu Pang, Keqin Liu, Michael W Mahoney, and Yaoqing Yang. 2023. Temperature balancing, layer-wise weight analysis, and neural network training. *Advances in Neural Information Processing Systems*, 36:63542–63572.

A Related Work

Pretraining Optimizers. A strong optimizer is critical for successful LLM training(Zhang et al., 2024a, 2025). For a long time, Adam (Kingma, 2014) and AdamW (Loshchilov and Hutter, 2017) have been the dominant choices. In recent years, a number of methods have emerged that build on the Adam family: some focus on reducing gradient variance (e.g., MARS (Yuan et al., 2024) and Cautious (Liang et al., 2024)), while others target memory efficiency (e.g., Lion (Chen et al., 2023), Adam-mini (Zhang et al., 2024b) and GaLore (Zhao et al., 2024)). However, these Adam variants remain *vector-based* optimizers and have difficulty capturing interdependencies among parameters. Some works (Zhou et al., 2023; Wang et al., 2025a) extend Adam by using layer- or module-wise learning rates to capture relationships among parameters at a coarse level. Recently, Muon (Jordan et al., 2024) has been proposed as a representative *matrix-based* optimizer. It uses the momentum matrix as a preconditioner, which better captures interdependencies among parameters and improves training stability. Muon has shown promising results in LLM training (Shah et al., 2025; Liu et al., 2025a; Wen et al., 2025). Following this line, several Muon variants have emerged, including NorMuon (Li et al., 2025) and AdaMuon (Si et al., 2025).

Heavy-tailed Phenomenon. Heavy tailed weight spectrum in machine learning have been observed and studied in various forms. Among the most prominent empirical findings are a series of works by Martin and Mahoney (2021); Martin et al. (2021), which propose the HT-SR theory for deep neural networks. They observe that well-trained networks often exhibit heavy-tailed spectra in their weight matrices, and the degree of heavy-tailedness is strongly correlated with model quality. Furthermore, they quantify this behavior via power-law fits to the ESDs of weight matrices. From a theoretical perspective, a number of studies (Simsekli et al., 2020; Hodgkinson et al., 2022; Simon et al., 2023; Dandi et al., 2024; Kothapalli et al., 2025; Defilippis et al., 2025; Deng et al., 2026) have used tools such as stochastic differential equations, random matrix theory, eigenlearning framework, and approximate message passing to characterize power-law behavior in weight ESDs and its strong connections to generalization bounds and test loss. Empirically, treating HT-SR theory as a diagnostic tool, subsequent work has explored layer-wise hyperparameter scheduling (Liu et al., 2024b; He et al., 2025; Zhou et al., 2023) and layer-wise pruning (Lu et al., 2024; Hu et al., 2025) based on the power-law fit to improve training performance and efficiency. Similarly, Liu et al. (2025c) use spectral analysis of layer-wise representations to identify change-point layers whose top principal components undergo significant shifts, connecting spectral properties to layer criticality during fine-tuning. In this work, motivated by HT-SR theory, we use HTMuon to make Muon updates more heavy-tailed to improve training.

B Proofs

Notation. For a vector $\boldsymbol{v}$, we denote its l_2 norm as $\|\boldsymbol{v}\|_2$. For a matrix $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m\times n}$, we denote its spectral norm as $\|\boldsymbol{A}\|$, Frobenius norm as $\|\boldsymbol{A}\|_{\mathrm{F}}$, nuclear norm as $\|\boldsymbol{A}\|_*$, where $\|\boldsymbol{A}\|_* = \sum_{i=1}^r \sigma_i$, here $\{\sigma_i\}_{i=1}^r$ are singular values, for $p > 1$, Schatten$-p$ norm as $\|\boldsymbol{A}\|_p$, where $\|\boldsymbol{A}\|_p = (\sum_{i=1}^r \sigma_i^p)^{\frac{1}{p}}$. We denote $\langle \boldsymbol{A}, \boldsymbol{B}\rangle = \mathrm{Tr}\left(\boldsymbol{A}^\top \boldsymbol{B}\right)$.

B.1 Update Under the Schatten$-q$ Norm Constraint

Lemma B.1 (Von Neumann trace inequality) *For matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, let $\sigma_i(\boldsymbol{A})$ denote the i-th largest singular value of $\boldsymbol{A}$, and let $r = \min(m, n)$. Then for any $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m\times n}$, we have*

$$\langle \boldsymbol{A}, \boldsymbol{B}\rangle = \mathrm{Tr}\left(\boldsymbol{A}^\top \boldsymbol{B}\right) \le \sum_{i=1}^r \sigma_i(\boldsymbol{A})\, \sigma_i(\boldsymbol{B}).$$

Theorem B.2 *Let $\widetilde{\boldsymbol{G}}_t = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ be the* SVD *of the negative gradient $\widetilde{\boldsymbol{G}}_t = -\nabla f(\boldsymbol{W}_t)$, where $\boldsymbol{\Sigma} = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$. Let p be the conjugate exponent of q, satisfying $1/p' + 1/q = 1$. The explicit solution $\Delta\boldsymbol{W}^*$ to the optimization problem 3 is given by:*

$$\Delta\boldsymbol{W}^* = \delta \frac{1}{\|\widetilde{\boldsymbol{G}}_t\|_{p'}^{p'/q}} \boldsymbol{U}\boldsymbol{\Sigma}^{p'-1}\boldsymbol{V}^\top, \tag{6}$$

where $\boldsymbol{\Sigma}^{p'-1} = \mathrm{diag}(\sigma_1^{p'-1}, \ldots, \sigma_r^{p'-1})$.

Proof. First, by invoking Von Neumann's trace inequality, we have the upper bound

$$\mathrm{Tr}(\widetilde{\boldsymbol{G}}_t^\top \Delta\boldsymbol{W}) \le \sum_i \sigma_i(\widetilde{\boldsymbol{G}}_t)\sigma_i(\Delta\boldsymbol{W})$$

The equality holds if and only if $\Delta\boldsymbol{W}$ shares the same left and right singular vectors as $\widetilde{\boldsymbol{G}}_t$. Thus, the optimal $\Delta\boldsymbol{W}$ must take the form:

$$\Delta\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^\top, \quad \text{where } \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_r). \tag{7}$$

Here, $\lambda_i \geq 0$ represents the singular values of $\Delta \boldsymbol{W}$. The problem reduces to maximizing $\sum_i \sigma_i \lambda_i$ subject to $\|\lambda\|_q \leq \delta$. By Hölder's inequality, we have the upper bound:

$$\sum_i \sigma_i \lambda_i \leq \|\sigma\|_{p'} \|\lambda\|_q \leq \|\sigma\|_{p'} \delta. \tag{8}$$

Equality is achieved if and only if $\lambda_i^q \propto \sigma_i^{p'}$, which implies:

$$\lambda_i = c\sigma_i^{p'/q}, \tag{9}$$

for some constant $c \geq 0$. Using the conjugate property $1/p' + 1/q = 1$, the exponent simplifies to:

$$\frac{p'}{q} = p'\left(1 - \frac{1}{p'}\right) = p' - 1. \tag{10}$$

Substituting this back yields the optimal singular values:

$$\lambda_i = c \cdot \sigma_i^{p'-1}, \tag{11}$$

where c is a normalization constant. To determine c, we substitute these values into the active constraint $\|\Delta \boldsymbol{W}\|_q = \delta$:

$$\left(\sum_i (c\sigma_i^{p'-1})^q\right)^{1/q} = c\left(\sum_i \sigma_i^{(p'-1)q}\right)^{1/q} = \delta. \tag{12}$$

Using the algebraic relation $(p'-1)q = p'$, the term involving singular values simplifies to the Schatten$-p'$ norm of $\widetilde{\boldsymbol{G}}_t$. Solving for c, we obtain:

$$c\|\widetilde{\boldsymbol{G}}_t\|_{p'}^{p'/q} = \delta \implies c = \frac{\delta}{\|\widetilde{\boldsymbol{G}}_t\|_{p'}^{p'/q}}. \tag{13}$$

Substituting c back into the expression $\Delta \boldsymbol{W} = U\Lambda V^\top$ yields the stated closed-form solution. □

B.2 Convergence Analysis

We consider the following stochastic optimization problem:

$$\min_{\boldsymbol{W} \in \mathbb{R}^{m\times n}} f(\boldsymbol{W}) = \mathbb{E}_{\boldsymbol{\xi}}[f(\boldsymbol{W}; \boldsymbol{\xi})],$$

where $\boldsymbol{\xi}$ is stochastic noise. We denote $f^* = \inf_{\boldsymbol{W}} f(\boldsymbol{W})$, $r = \min\{m, n\}$. We assume $f^* > -\infty$.

Assumption B.3 (F-norm Lipschitz smooth) *Define $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is L Frobenius norm Lipschitz smooth if for any $\boldsymbol{W}, \boldsymbol{W}' \in \mathbb{R}^{m\times n}$, we have*

$$\|\nabla f(\boldsymbol{W}) - \nabla f(\boldsymbol{W}')\|_{\mathrm{F}} \leq L\|\boldsymbol{W} - \boldsymbol{W}'\|_{\mathrm{F}}.$$

Assumption B.4 (Bounded variance) *We assume $\nabla f(\boldsymbol{W}; \xi)$ is an unbiased stochastic estimator of the true gradient $\nabla f(\boldsymbol{W})$ and has a bounded variance, i.e.*

$$\mathbb{E}[\nabla f(\boldsymbol{W}; \xi)] = \nabla f(\boldsymbol{W}),$$
$$\mathbb{E}\|\nabla f(\boldsymbol{W}; \xi) - \nabla f(\boldsymbol{W})\|_{\mathrm{F}}^2] \leq \sigma^2.$$

Assumption B.5 (Bounded singular value) *We assume the largest singular value $\sigma_{\max}(\boldsymbol{M}_t) \leq l$.*

Definition B.6 *We say $\boldsymbol{W}$ is an ϵ-nuclear norm stationary point of f if $\|\nabla f(\boldsymbol{W})\|_* \leq \epsilon$.*

We give the convergence analysis under a smooth non-convex setting for finding ϵ-stationary points, i.e. $\|\nabla f(x)\|_* \leq \epsilon$.

To better understand the proof, we give a stochastic version of HTMuon in Algorithm 2 for convergence analysis.

Algorithm 2 HTMuon_Stochastic

Input: Initial weights $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, loss function L, learning rate η_t, momentum parameter β, batch size B, power $p \in (0, 1)$.
Initialize $\mathbf{M}_0 \in \mathbb{R}^{m\times n} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  Sample batch $\{\xi_{t,i}\}_{i=1}^B$ uniformly
  $\mathbf{G}_t = \frac{1}{B}\sum_{i=1}^B \nabla_{\mathbf{W}} L(W_t; \xi_{t,i})$
  $\mathbf{M}_t \leftarrow \beta \mathbf{M}_{t-1} + (1-\beta)\mathbf{G}_t$
  $\boldsymbol{U}_t, \boldsymbol{\Sigma}_t, \boldsymbol{V}_t^\top \leftarrow \mathtt{SVD}(\mathbf{M}_t)$
  $\boldsymbol{O}_t \leftarrow \boldsymbol{U}_t \boldsymbol{\Sigma}_t^p \boldsymbol{V}_t^\top$
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta_t \boldsymbol{O}_t$
end for

Lemma B.7 *For $t = 0, 1, \cdots, T$, $\boldsymbol{M}_t$ and $\boldsymbol{W}_t$ are generated by Algorithm 2. Consider that $\boldsymbol{M}_0 = \boldsymbol{G}_0$, $\boldsymbol{M}_t = \beta \boldsymbol{M}_{t-1} + (1-\beta)\boldsymbol{G}_t$, under Assumptions B.3 and B.4, if we assume $\eta = \max\{\eta_t\}_{t=1}^T$, under Assumption B.5, we have*

$$\mathbb{E}\|\nabla f(\boldsymbol{W}_t) - \boldsymbol{M}_t\|_{\mathrm{F}} \leq \sqrt{\frac{1-\beta}{1+\beta}}\frac{\sigma}{\sqrt{B}} + \frac{\beta^t \sigma}{\sqrt{B}} + \frac{\sqrt{r} l^p \beta L \eta}{1-\beta},$$

where $\boldsymbol{G}_t = \frac{1}{B}\sum_{i=1}^B \nabla f(\boldsymbol{W}_t; \xi_{t,i})$ and B is batch size.

Proof. We define $\boldsymbol{C}_0 = \nabla f(\boldsymbol{W}_0), \boldsymbol{C}_t = \beta \boldsymbol{C}_{t-1} + (1-\beta)\nabla f(\boldsymbol{W}_t) = (1-\beta)\sum_{i=1}^t \beta^{t-i}\nabla f(\boldsymbol{W}_i) + \beta^t \nabla f(\boldsymbol{W}_0)$, under

Assumptions B.3 and B.4 we note that

$$\begin{aligned}
&\mathbb{E}[\|\nabla f(\boldsymbol{W}_t-\boldsymbol{C}_t)\|_{\mathrm{F}}]\\
=&\mathbb{E}[\|\nabla f(\boldsymbol{W}_t)-(\beta\boldsymbol{C}_{t-1}+(1-\beta)\nabla f(\boldsymbol{W}_t))\|_{\mathrm{F}}]\\
=&\mathbb{E}[\beta\|\nabla f(\boldsymbol{W}_t-\boldsymbol{C}_{t-1})\|_{\mathrm{F}}]\\
\le&\mathbb{E}\big[\beta\|\nabla f(\boldsymbol{W}_{t-1}-\boldsymbol{C}_{t-1})\|_{\mathrm{F}}\big]\\
+&\mathbb{E}\big[\beta\|\nabla f(\boldsymbol{W}_{t-1}))-\nabla f(\boldsymbol{W}_t))\|_{\mathrm{F}}\big]\\
\le&\mathbb{E}\big[\beta\|\nabla f(\boldsymbol{W}_{t-1}-\boldsymbol{C}_{t-1})\|_{\mathrm{F}}\big]\\
+&\mathbb{E}\big[\beta L\|\boldsymbol{W}_{t-1}-\boldsymbol{W}_t\|_{\mathrm{F}}\big]\\
\le&\mathbb{E}\big[\beta\|\nabla f(\boldsymbol{W}_{t-1}-\boldsymbol{C}_{t-1})\|_{\mathrm{F}}\big]\\
+&\mathbb{E}\big[\beta L\eta\|\boldsymbol{U}_{t-1}\boldsymbol{\Sigma}_{t-1}^p\boldsymbol{V}_{t-1}^\top\|_{\mathrm{F}}\big]\\
\le&\mathbb{E}\big[\beta\|\nabla f(\boldsymbol{W}_{t-1}-\boldsymbol{C}_{t-1})\|_{\mathrm{F}}+\beta L\eta\sqrt{r}l^p\big]\\
\le&\beta^t\|\nabla f(\boldsymbol{W}_0)-\boldsymbol{C}_0\|_{\mathrm{F}}+\sum_{i=1}^{t}\beta^i L\eta\sqrt{r}l^p\\
\le&\frac{\sqrt{r}l^p\beta L\eta}{1-\beta}.
\end{aligned}$$

And follow the same proof in Section B.1 in Shen et al. (2025), we have

$$\mathbb{E}[\|\boldsymbol{C}_t-\boldsymbol{M}_t\|_{\mathrm{F}}]\le\sqrt{\frac{1-\beta}{1+\beta}}\frac{\sigma}{\sqrt{B}}+\beta^t\frac{\sigma}{\sqrt{B}}.$$

So we have

$$\begin{aligned}
&\mathbb{E}\|\nabla f(\boldsymbol{W}_t)-\boldsymbol{M}_t\|_{\mathrm{F}}\\
\le&\mathbb{E}[\|\nabla f(\boldsymbol{W}_t-\boldsymbol{C}_t)\|_{\mathrm{F}}]+\mathbb{E}[\|\boldsymbol{C}_t-\boldsymbol{M}_t\|_{\mathrm{F}}]\\
\le&\sqrt{\frac{1-\beta}{1+\beta}}\frac{\sigma}{\sqrt{B}}+\frac{\beta^t\sigma}{\sqrt{B}}+\frac{\sqrt{r}l^p\beta L\eta}{1-\beta}
\end{aligned}$$

□

Theorem B.8 (Muon (Shen et al., 2025)) *Under Assumptions B.3 and B.4, if we apply Muon with $\eta_t=\eta$, then*

$$\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\boldsymbol{W}_t)\|_*\big]\le&\frac{\mathbb{E}\big[f(\boldsymbol{W}_0)-f(\boldsymbol{W}_T)\big]}{T\eta}\\
&+\frac{Lr\eta}{2}+\frac{2\sigma\sqrt{r(1-\beta)}}{\sqrt{B(1+\beta)}}+\frac{2\beta\sigma\sqrt{r}}{(1-\beta)T\sqrt{B}}+\frac{2r\eta\beta L}{1-\beta}.
\end{aligned}$$

Where B is batch size. Denote $\Delta=f(\boldsymbol{W}_0)-f^*$. If we set $B=1$, $\eta=\sqrt{\frac{(1-\beta)\Delta}{rTL}}$, $1-\beta=\min\left\{\frac{\sqrt{L\Delta}}{\sigma\sqrt{T}},1\right\}$, then

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(\boldsymbol{W}_t)\|_*\big]\le\\
&O\left(\sqrt[4]{\frac{r^2L\Delta\sigma^2}{T}}+\sqrt{\frac{rL\Delta}{T}}+\frac{\sqrt{r}\sigma^2}{\sqrt{L\Delta T}}\right).
\end{aligned}$$

Thus, Muon *can find an ϵ-nuclear norm stationary point of f with a complexity of $O(r^2L\sigma^2\Delta\epsilon^{-4})$.*

Theorem B.9 (HTMuon: Theorem 5.2) *Under Assumptions B.3, B.4 and B.5, if we apply* HTMuon *in Algorithm 2 with adaptive learning rate $\eta_t=\frac{\langle\boldsymbol{M}_t,\rho(\boldsymbol{M}_t)\rangle}{L\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2}$, $\rho(\boldsymbol{M}_t)=\boldsymbol{U}_t\boldsymbol{\Sigma}_t^p\boldsymbol{V}_t^\top$ and batch size $B=T$, then*

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\nabla f(\boldsymbol{W}_t)\|_{1+p}^2\big]\le&\frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\left(\frac{1-\varepsilon'}{L}+\frac{1}{L\varepsilon'}\right)\\
&\Big[\frac{2(1-\beta)\sigma}{B(1+\beta)}+\frac{2\beta^2\sigma^2}{(1-\beta^2)BT}+\frac{2rl^{2p}\beta^2L^2\eta^2}{(1-\beta)^2}\Big]\\
&+\frac{4Lr^{\frac{1}{s}}}{1-\varepsilon'}\frac{\big[f(\boldsymbol{W}_0)-f(\boldsymbol{W}_T)\big]}{T}.
\end{aligned}$$

Where $s=\frac{1+p}{1-p}$ and $0<\varepsilon'<1$. Furthermore, if we define $\beta=\frac{1}{\sqrt{T}}$, $\Delta=f(\boldsymbol{W}_0)-f^*$, thus we have

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\nabla f(\boldsymbol{W}_t)\|_{1+p}^2\big]\le&\frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\left(\frac{1-\varepsilon'}{L}+\frac{1}{L\varepsilon'}\right)\\
&\Big[\frac{2\sigma}{T}+\frac{4\sigma^2}{T}+\frac{4rl^{2p}L^2\eta^2}{T}\Big]+\frac{4Lr^{\frac{1}{s}}}{1-\varepsilon'}\frac{\Delta}{T}.
\end{aligned}$$

Proof. Consider $\boldsymbol{M}_t=\boldsymbol{U}_t\boldsymbol{\Sigma}_t\boldsymbol{V}_t^\top$, we first define $\rho(\boldsymbol{M}_t)=\boldsymbol{U}_t\boldsymbol{\Sigma}_t^p\boldsymbol{V}_t^\top$. By Assumption B.3, we have

$$\begin{aligned}
&\mathbb{E}\big[f(\boldsymbol{W}_t)-f(\boldsymbol{W}_{t+1})\big]\\
&\ge\mathbb{E}\big[\eta_t\langle\nabla f(\boldsymbol{W}_t),\rho(\boldsymbol{M}_t)\rangle-\frac{L\eta_t^2}{2}\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2\big]\\
&\ge\mathbb{E}\big[\eta_t\langle\boldsymbol{M}_t,\rho(\boldsymbol{M}_t)\rangle-\frac{L\eta_t^2}{2}\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2\\
&\qquad-\eta_t\langle\boldsymbol{M}_t-\nabla f(\boldsymbol{W}_t),\rho(\boldsymbol{M}_t)\rangle\big].
\end{aligned}$$

Here we consider adaptive learning rate $\eta_t=\frac{\langle\boldsymbol{M}_t,\rho(\boldsymbol{M}_t)\rangle}{L\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2}>0$, thus

$$\begin{aligned}
&\mathbb{E}\big[f(\boldsymbol{W}_t)-f(\boldsymbol{W}_{t+1})\big]\\
&\ge\mathbb{E}\Big[\Big(\frac{\langle\boldsymbol{M}_t,\rho(\boldsymbol{M}_t)\rangle^2}{L\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2}-\frac{\langle\boldsymbol{M}_t,\rho(\boldsymbol{M}_t)\rangle^2}{2L\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2}\Big)\\
&\qquad-\eta_t\|\boldsymbol{M}_t-\nabla f(\boldsymbol{W}_t)\|_{\mathrm{F}}\|\rho(\boldsymbol{M}_t)\|_F\Big]\\
&\ge\mathbb{E}\Big[\Big(\frac{\langle\boldsymbol{M}_t,\rho(\boldsymbol{M}_t)\rangle^2}{L\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2}-\frac{\langle\boldsymbol{M}_t,\rho(\boldsymbol{M}_t)\rangle^2}{2L\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2}\Big)\\
&\qquad-\frac{\langle\boldsymbol{M}_t,\rho(\boldsymbol{M}_t)\rangle}{L\|\rho(\boldsymbol{M}_t)\|_F}\|\boldsymbol{M}_t-\nabla f(\boldsymbol{W}_t)\|_{\mathrm{F}}\Big]
\end{aligned}$$

By $2ab \leq \varepsilon' a^2 + \frac{b^2}{\varepsilon'}$, where $0 < \varepsilon' < 1$, we have

$$\begin{aligned}
&\mathbb{E}\big[f(\boldsymbol{W}_t) - f(\boldsymbol{W}_{t+1})\big] \\
&\geq \mathbb{E}\big[\frac{\langle \boldsymbol{M}_t, \rho(\boldsymbol{M}_t)\rangle^2}{L\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2} - \frac{\langle \boldsymbol{M}_t, \rho(\boldsymbol{M}_t)\rangle^2}{2L\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2} \\
&- \frac{\varepsilon'}{2L}\frac{\langle \boldsymbol{M}_t, \rho(\boldsymbol{M}_t)\rangle^2}{\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2} - \frac{1}{2L\varepsilon'}\|\boldsymbol{M}_t - \nabla f(\boldsymbol{W}_t)\|_{\mathrm{F}}^2\big] \\
&\geq \mathbb{E}\big[\frac{1-\varepsilon'}{2L}\frac{\langle \boldsymbol{M}_t, \rho(\boldsymbol{M}_t)\rangle^2}{\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2} - \frac{\|\boldsymbol{M}_t - \nabla f(\boldsymbol{W}_t)\|_{\mathrm{F}}^2}{2L\varepsilon'}\big]
\end{aligned}$$

By Holder's Inequality, for $p \in (0,1)$ and we set $s = \frac{1+p}{1-p}$ and $m = \frac{1+p}{2p}$, we have

$$\begin{aligned}
\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2 = \sum_{i=1}^{r}\sigma_i^{2p} &\leq r^{\frac{1}{s}}\left(\sum_{i=1}^{r}(\sigma_i^{2p})^m\right)^{\frac{1}{m}} \\
&\leq r^{\frac{1}{s}}\left(\sum_{i=1}^{r}\sigma_i^{1+p}\right)^{\frac{2p}{p+1}}.
\end{aligned}$$

Thus we have

$$\begin{aligned}
\frac{\langle \boldsymbol{M}_t, \rho(\boldsymbol{M}_t)\rangle^2}{\|\rho(\boldsymbol{M}_t)\|_{\mathrm{F}}^2} &\geq \frac{\left(\sum_{i=1}^{r}\sigma_i^{1+p}\right)^2}{r^{\frac{1}{s}}\left(\sum_{i=1}^{r}\sigma_i^{1+p}\right)^{\frac{2p}{p+1}}} \\
&= \frac{\|\boldsymbol{M}_t\|_{1+p}^2}{r^{\frac{1}{s}}}.
\end{aligned}$$

Hence, we have

$$\begin{aligned}
&\mathbb{E}\big[f(\boldsymbol{W}_t) - f(\boldsymbol{W}_{t+1})\big] \geq \\
&\mathbb{E}\big[\frac{1-\varepsilon'}{2Lr^{\frac{1}{s}}}\|\boldsymbol{M}_t\|_{1+p}^2 - \frac{\|\boldsymbol{M}_t - \nabla f(\boldsymbol{W}_t)\|_{\mathrm{F}}^2}{2L\varepsilon'}\big].
\end{aligned}$$

By Lemma B.7, we let $\eta = \max\{\eta_t\}_{t=1}^T$, we have

$$\begin{aligned}
&\mathbb{E}\|\nabla f(\boldsymbol{W}_t) - \boldsymbol{M}_t\|_{\mathrm{F}}^2 \\
&\leq \left(\sqrt{\frac{1-\beta}{1+\beta}}\frac{\sigma}{\sqrt{B}} + \frac{\beta^t\sigma}{\sqrt{B}} + \frac{\sqrt{r}l^p\beta L\eta}{1-\beta}\right)^2 \\
&\leq \frac{2(1-\beta)\sigma}{B(1+\beta)} + \frac{2\beta^{2t}\sigma^2}{B} + \frac{2rl^{2p}\beta^2L^2\eta^2}{(1-\beta)^2}
\end{aligned}$$

Thus we have

$$\begin{aligned}
&\mathbb{E}\big[\frac{1-\varepsilon'}{2Lr^{\frac{1}{s}}}\|\nabla f(\boldsymbol{W}_t)\|_{1+p}^2\big] \\
&\leq \mathbb{E}\big[\frac{1-\varepsilon'}{Lr^{\frac{1}{s}}}\|\nabla f(\boldsymbol{W}_t) - \boldsymbol{M}_t\|_{1+p}^2\big] \\
&+ \mathbb{E}\big[\frac{1-\varepsilon'}{Lr^{\frac{1}{s}}}\|\boldsymbol{M}_t\|_{1+p}^2\big] \\
&\leq \mathbb{E}\big[\frac{1-\varepsilon'}{L}\|\nabla f(\boldsymbol{W}_t) - \boldsymbol{M}_t\|_{\mathrm{F}}^2\big] \\
&+ \mathbb{E}\big[\frac{1-\varepsilon'}{Lr^{\frac{1}{s}}}\|\boldsymbol{M}_t\|_{1+p}^2\big] \\
&\leq \mathbb{E}\big[\left(\frac{1-\varepsilon'}{L} + \frac{1}{L\varepsilon'}\right)\|\nabla f(\boldsymbol{W}_t) - \boldsymbol{M}_t\|_{\mathrm{F}}^2 \\
&\quad + f(\boldsymbol{W}_t) - f(\boldsymbol{W}_{t+1})\big]
\end{aligned}$$

Thus we have

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\nabla f(\boldsymbol{W}_t)\|_{1+p}^2\big] \leq \\
&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\left(\frac{1-\varepsilon'}{L} + \frac{1}{L\varepsilon'}\right)\|\nabla f(\boldsymbol{W}_t) - \boldsymbol{M}_t\|_{\mathrm{F}}^2\big] \\
&+ \frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\frac{\big[f(\boldsymbol{W}_0) - f(\boldsymbol{W}_T)\big]}{T} \\
&\leq \frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\left(\frac{1-\varepsilon'}{L} + \frac{1}{L\varepsilon'}\right) \\
&\times \big[\frac{2(1-\beta)\sigma}{B(1+\beta)} + \frac{2\beta^2\sigma^2}{(1-\beta^2)BT} + \frac{2rl^{2p}\beta^2L^2\eta^2}{(1-\beta)^2}\big] \\
&+ \frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\frac{\big[f(\boldsymbol{W}_0) - f(\boldsymbol{W}_T)\big]}{T}
\end{aligned}$$

We define $B = T, \beta = \frac{1}{\sqrt{T}}$, $\Delta = f(\boldsymbol{W}_0) - f^*$, thus we have

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\nabla f(\boldsymbol{W}_t)\|_{1+p}^2\big] \leq \frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\left(\frac{1-\varepsilon'}{L} + \frac{1}{L\varepsilon'}\right) \\
&\big[\frac{2\sigma}{T} + \frac{2\sigma^2}{(1-\beta^2)T^3} + \frac{2rl^{2p}L^2\eta^2}{T(1-\beta)^2}\big] + \frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\frac{\Delta}{T}.
\end{aligned}$$

When $\beta = \frac{1}{\sqrt{T}}$, it is easy to get $1 - \beta \geq \frac{1}{2}$, thus we have

$$\begin{aligned}
&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\big[\|\nabla f(\boldsymbol{W}_t)\|_{1+p}^2\big] \leq \frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\left(\frac{1-\varepsilon'}{L} + \frac{1}{L\varepsilon'}\right) \\
&\big[\frac{2\sigma}{T} + \frac{4\sigma^2}{T^3} + \frac{4rl^{2p}L^2\eta^2}{T}\big] + \frac{2Lr^{\frac{1}{s}}}{1-\varepsilon'}\frac{\Delta}{T}.
\end{aligned}$$

Based on Theorem 5.2 and Definition B.6, when $B = T$, the sample complexity upper bound is

$O(\epsilon^{-4})$, which matches Muon's sample complexity in Theorem B.8 and (Chang et al., 2025) and SGDM's sample complexity in Garrigos and Gower (2023); Arjevani et al. (2023). . □

B.3 Lemma for PL exponent α

Lemma B.10 *Suppose the singular values of matrix $\boldsymbol{W} \in \mathbb{R}^{n\times m}$ follow $s_k = s_1 k^{-s}, 1 \le k \le n$, we have PL exponent α of $\boldsymbol{W}$ satisfies $\alpha = 1 + \frac{1}{2s}$.*

Proof. Since we have $s_k = s_1 k^{-s}, 1 \le k \le n$, which means eigenvalues $\lambda_k = \lambda_1 k^{-2s}, 1 \le k \le n$. Here we suppose Λ is a random variable distributed according to the empirical distribution from these eigenvalues, we can see that the distribution function takes the following form:

$$\mathbb{P}(\Lambda > \lambda_1 k^{-2s}) = \frac{k}{n}.$$

By changing variables $\lambda_1 k^{-s} = \lambda$, we get the cumulative distribution function of Λ:

$$\mathbb{P}(\Lambda > \lambda) \sim \lambda^{-\frac{1}{2s}}.$$

After that, we take the derivative with respect to λ, and we get the ESD:

$$p(\lambda) \sim \lambda^{-\left(\frac{1}{2s}+1\right)}.$$

So we have $\alpha = 1 + \frac{1}{2s}$. □

C More Experimental Details

In this section, we present more experimental details.

C.1 Muon and HTMuon Algorithms

In Algorithm 3 and 4, we give the implementations of Muon_NS and Muon_SVD. Muon_NS corresponds to the commonly used five-step Newton Schulz implementation of Muon, whereas Muon_SVD represents the theoretically exact SVD-based implementation. Empirically, we find that due to the numerical approximation in Muon_NS, the singular values of the update matrix do not always equal 1 exactly. Such deviations may suppress weights along noise-dominated eigenvector directions, which could explain why Muon_NS performs better than Muon_SVD in practice.

In Algorithm 5, we give the implementations of HTMuon_NS. As discussed in Section 4.4, we note that the HTMuon update $\boldsymbol{O}_t = \boldsymbol{U}_t \boldsymbol{\Sigma}_t^p \boldsymbol{V}_t^\top$ admits the factorization

$$\boldsymbol{O}_t = (\boldsymbol{U}_t \boldsymbol{V}_t^\top)\,(\boldsymbol{V}_t \boldsymbol{\Sigma}_t^p \boldsymbol{V}_t^\top).$$

For $\boldsymbol{U}_t \boldsymbol{V}_t^\top$ part, this is Muon's update, we can consider Newton-Schulz with 5 steps. To compute the symmetric factor $\boldsymbol{V}_t \boldsymbol{\Sigma}_t^p \boldsymbol{V}_t^\top$ efficiently, we apply the NS_root routine to $\boldsymbol{M}_t^\top \boldsymbol{M}_t$ in Algorithm 6, since

$$\boldsymbol{M}_t^\top \boldsymbol{M}_t = \boldsymbol{V}_t \boldsymbol{\Sigma}_t^2 \boldsymbol{V}_t^\top,$$

we have $\boldsymbol{V}_t \boldsymbol{\Sigma}_t^p \boldsymbol{V}_t^\top = (\boldsymbol{M}_t^\top \boldsymbol{M}_t)^{\frac{p}{2}}$. Accordingly, NS_root uses NewtonSchulz iterations to approximate matrix square roots and applies successive square-root operations for $[1 - \log_2 p]$ rounds.

C.2 Training Loss Curve

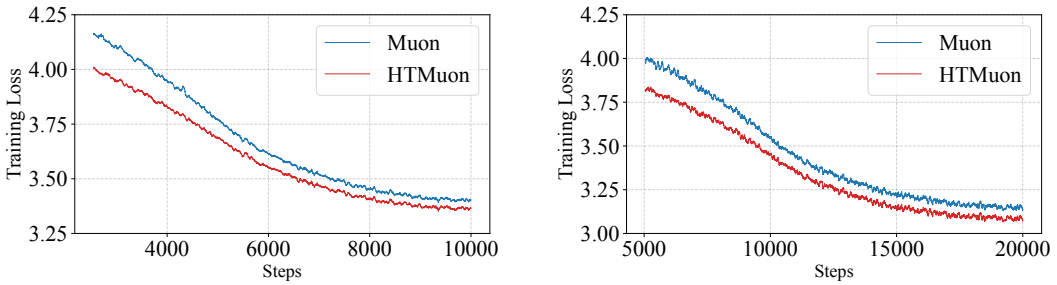
In Figure 8, we show the smoothed training loss curves for LLaMA-60M and LLaMA-135M on C4 dataset. We find in the later stages of training, HTMuon consistently achieves a noticeably lower training loss than Muon, demonstrating the effectiveness and training stability.

C.3 More Experiments Results

In Table 7, we conduct evaluations on LLaMA-60M and LLaMA-135M on the C4 datatset, comparing HTMuon with multiple Muon-variant optimizers as well as other state-of-the-art optimizers. HTMuon consistently achieves the best results. Furthermore, combining HTMuon with other Muon-variant optimizers will achieve better performance. Detailed hyperparameters are reported in Appendix D.

In Table 8 and 9, we report the perplexity and average per-step runtime for HTMuon, HTMuon_NS, and HTMuon+NorMuon under different update intervals. We measure per-step runtime on 4 NVIDIA A6000 GPUs. We find that increasing the interval substantially reduces the average overhead of HTMuon, while it still outperforms Muon. Under the same interval, HTMuon_NS further reduces the average overhead and also remains better than Muon. Similarly, for HTMuon+NorMuon, larger intervals make its average overhead approach that of NorMuon, while consistently outperforming NorMuon. For more visualizations, please refer to Figure 5 and 9.

Besides reporting per-step runtime overhead, we also report the wall-clock time for Muon, HTMuon and HTMuon_NS with intervals. To better report wall-clock time, we rerun training on NVIDIA RTX PRO 6000 GPUs: LLaMA-60M on C4 using 2 GPUs, and LLaMA-135M on C4 using 4 GPUs. We use the optimal hyperparameter settings reported in the Table 19. We summarize the PPL and wall-clock time of Muon, HTMuon, HTMuon_NS, HTMuon (Interval $= 5$), and HTMuon_NS (Interval

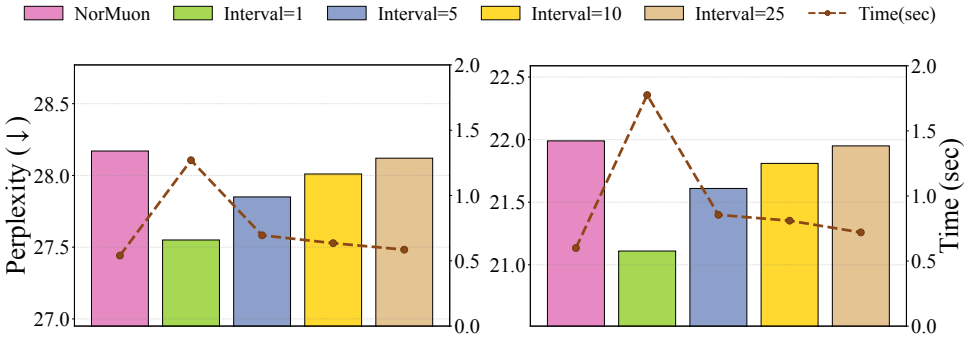


(a) LLaMa-60M on C4 (b) LLaMA-135M on C4

Figure 8: Training loss curves for LLaMA-60M and LLaMA-135M. Learning rate for both models and optimizers is 0.03. Both curves are smoothed via a simple moving average (uniform weights, window = 50).

Table 7: We evaluate HTMuon on LLaMA-60M and LLaMA-135M, comparing it against several Muon variant optimizers and other state-of-the-art optimizers. Red indicates the best value, and blue denotes the second-best value. We find that HTMuon consistently outperforms these baselines. More hyperparameter details are provided in Appendix D.

	Muon	HTMuon	NorMuon	HTMuon+NorMuon	AdaMuon	MARS	SOAP	Cautious	COSMOS	GaLore	Sophia
LLaMa-60M	28.80	27.88	28.17	**27.55**	28.67	29.13	28.96	29.71	28.62	34.26	34.02
LLaMa-135M	22.23	21.25	21.99	**21.11**	22.43	22.42	22.43	23.15	22.15	25.11	25.63



(a) LLaMA-60M (b) LLaMA-135M

Figure 9: we evaluate applying HTMuon+ NorMuon on LLaMA-60M and LLaMA-135M every 1, 5, 10, and 25 steps (while other steps applying NorMuon) . We report the average per-step runtime overhead for all methods. Detailed results and hyperparameter settings are provided in Table 8 and 9 in Appendix D.

$= 5$) in the Table 10. We observe that for LLaMA-60M, HTMuon_NS (Interval $= 5$) achieves only a $\sim$6% additional overhead yet outperforms Muon by 0.34 PPL. Moreover, the time for HTMuon_NS (Interval $= 5$) to reach 28.84 PPL is 0.64 hour, compared to 0.67 hour for Muon. For LLaMA-135M, HTMuon_NS (Interval = 5) incurs only $\sim$11% additional overhead while outperforming Muon by 0.25 PPL; it reaches 22.27 PPL in 1.32 hour, compared to 1.40 hour for Muon.

We also give the FLOPs analysis for Muon and HTMuon_NS. We consider initial weights $\boldsymbol{W}_0 \in \mathbb{R}^{m\times n}$.

- FLOPs analysis for Muon(Algorithm 3):

$$F_{Muon} = 20mnr + O(mn).$$

- FLOPs analysis for HTMuon_NS(Algorithm 5):

$$F_{HTMuon\_NS} = 20mnr + 4mn^2 + 6LTn^3 + O(n^2),$$

where $r = \min(m, n), L = \left\lceil \log_2\left(\frac{2}{p}\right)\right\rceil$ and T=ns_steps in Algorithm 5.

Compared to Muon, the additional FLOPs in HTMuon_NS mainly come from the Newton-Schulz iterations used for the fractional power approximation.

Table 8: We report the exact PPL obtained when training LLaMA-60M and LLaMA-135M using HTMuon, HTMuon_NS, and HTMuon+NorMuon under different update intervals. For Muon and NorMuon, interval updates are not applicable; therefore, we only report their PPL after training.

	LLaMA-60M				LLaMA-135M			
Optimizer	1	5	10	25	1	5	10	25
Muon		28.80				22.23		
HTMuon	27.88	28.36	28.58	28.64	21.25	22.01	22.13	22.20
HTMuon_NS	28.03	28.42	28.67	28.74	21.35	22.05	22.14	22.20
NorMuon		28.17				21.99		
HTMuon+NorMuon	27.55	27.85	28.01	28.12	21.11	21.61	21.81	21.95

Table 9: We report the average per-step runtime overhead obtained when training LLaMA-60M and LLaMA-135M using HTMuon, HTMuon_NS, and HTMuon+NorMuon under different update intervals. We measure per-step runtime on 4 NVIDIA A6000 GPUs. For Muon and NorMuon, interval updates are not applicable; therefore, we only report average per-step runtime overhead.

	LLaMA-60M				LLaMA-135M			
Optimizer	1	5	10	25	1	5	10	25
Muon		0.51				0.59		
HTMuon	1.26	0.67	0.61	0.54	1.70	0.82	0.70	0.63
HTMuon_NS	0.66	0.59	0.56	0.54	0.92	0.68	0.64	0.63
NorMuon		0.54				0.60		
HTMuon+NorMuon	1.27	0.70	0.64	0.59	1.78	0.86	0.81	0.72

We find that although HTMuon could take longer than Muon, the substantial gains achieved through heavy-tailed spectral correction suggest that HTMuon captures the correct inductive bias for learning. Furthermore, we believe that efficient variants such as HTMuon_NS with interval-based updates provide a strong balance between performance and efficiency.

In Figure 10a, 10b, we visualize the layer-wise PL exponent α for ResNet18 on CIFAR-100 and CIFAR-10 dataset to further support that HTMuon makes the updated matrices more heavy-tailed than Muon.

C.4 More Ablation Study

C.4.1 Varying learning rates

In Figure 12, Table 11, 12 and 13, we conduct learning grid search on LLaMA and ResNet models, our experiments show that for most learning rates, HTMuon outperforms Muon and NorMuon, which exhibits our method's effectiveness and robustness.

C.4.2 Varying different p

In table 14, we give the exact results for ablation study on p in Figure 7, we test multiple values of p on LLaMA-60M and LLaMA-135M on the C4 dataset, and find that $p = 0.125$ is a robust choice. We therefore use $p = 0.125$ in most experiments.

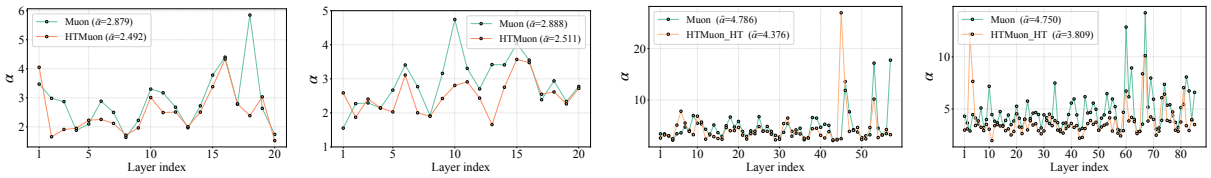
Here we discuss more about hyperparameter sensitivity for p. Although introducing p adds a tuning dimension and add hyperparameter search burden, also we only do grid search on LLaMA models to obtain the optimal $p = 0.125$, we will clarify that $p = 0.125$ can serve as a recommended default for other tasks.

- $p = 0.125$ **generalizes well across tasks and architectures.** In our paper, we also use $p = 0.125$ for HTMuon when training GPT-2 small on OpenWebText dataset, ResNet-18 and ResNet-50 on CIFAR datasets, Tables 2, 12and 13 in our paper show that HTMuon consistently outperforms Muon.

- $p = 0.125$ **is often optimal or near-optimal across tasks and architectures.** To explore whether $p = 0.125$ is also close to optimal on other tasks, we performed grid searches over p on the CIFAR and OpenWebText datasets in Table 15 and 16. We find that on CIFAR-10, $p = 0.125$ achieves the best performance for both ResNet-18 and ResNet-50, and on OpenWebText, p=0.125 also yields the best

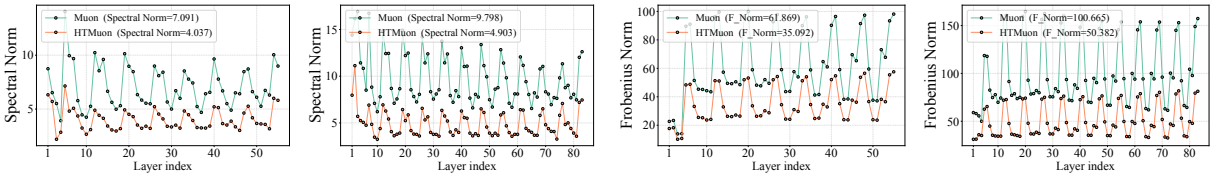
Table 10: We report the perplexity and wall-clock time of when training LLaMA-60M and LLaMA-135M using Muon, HTMuon, HTMuon_NS, HTMuon(Interval=5), HTMuon_NS(Interval=5). We measure wall-clock time on 4 NVIDIA RTX PRO 6000 GPUs.

	LLaMA-60M		LLaMA-135M	
Optimizer	PPL	Time(hour)	PPL	Time(hour)
Muon	28.84	0.67	22.27	1.40
HTMuon_NS(Interval=5)	28.50	0.71	22.02	1.55
HTMuon(Interval=5)	28.36	1.02	22.04	2.40
HTMuon_NS	28.06	0.87	21.36	2.17
HTMuon	27.87	2.46	21.24	6.57



(a) ResNet18 on CIFAR-100 (b) ResNet18 on CIFAR-10 (c) HTMUON_HT_60M (d) HTMUON_HT_135M

Figure 10: (a)(b): Layer-wise PL α for ResNet-18 model weight ESDs on CIFAR-100 and CIFAR-10 trained with Muon and HTMuon. (c)(d): Layer-wise PL α for LLaMA model weight ESDs trained with Muon and HTMuon_HT. All models used for visualization are trained using each optimizer's best-performing hyperparameter configuration. For hyperparameter configurations, please refer to Appendix D.



(a) LLaMA-60M on C4 (b) LLaMA-135M on C4 (c) LLaMA-60M on C4 (d) LLaMA-135M on C4

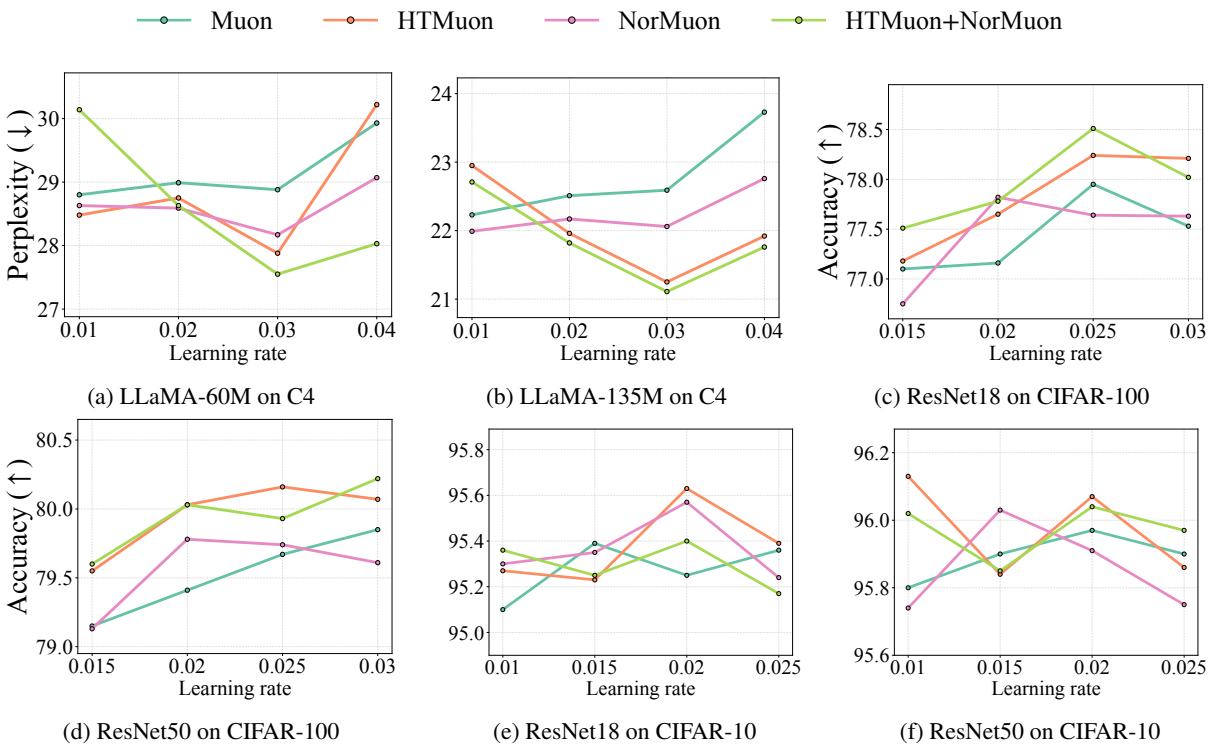
Figure 11: Layer-wise spectral norm and frobenius norm for LLaMA model weights trained with Muon and HTMuon. All models used for visualization are trained using each optimizer's best-performing hyperparameter configuration. For hyperparameter configurations, please refer to Appendix D.

result for GPT-2 small. While on CIFAR-100 a larger value (e.g., $p = 0.2$) can perform slightly better, this does not change our overall conclusion that p=0.125 is a strong and practical recommended default.

C.4.3 Varying different ways to make Muon update more heavy-tailed

In Algorithm 7, we explore another way HTMuon_HT to make Muon updates more heavy-tailed , which directly replaces the momentum matrix's singular values with a pre-specified heavy-tailed distribution. We use $\alpha = 0.25$ in Algorithm 7 for experiments. In Table 17, we find that, although HTMuon_HT is not as strong as HTMuon, it consistently outperforms Muon. In Figure 10c and 10d, we visualize the layer-wise PL exponent α for LLaMA-60M and LLaMA-135M, and observe that HTMuon_HT also yields consistently smaller α than Muon across layers. This suggests that the proposed heavy-tailed spectral correction can indeed improve Muon.

Since some readers maybe confused that the performance improvement attribution of our method HTMuon is from inducing heavy tails or preserving singular value information. Here we discuss more about the performance improvement attribution for our algorithm HTMuon based on HTMuon_HT. Our motivation for heavy-tail spectral correction is that Muon enforces unit singular values for update matrices. This induces a light-tailed spectrum and can over-emphasize noise-dominated directions, which may limit performance. Guided by HT-SR theory, we therefore consider heavy-tailed spectral correction. We study two ways of introducing heavy-tailed spectra: HTMuon_HT(which merely induces



(a) LLaMA-60M on C4 (b) LLaMA-135M on C4 (c) ResNet18 on CIFAR-100

(d) ResNet50 on CIFAR-100 (e) ResNet18 on CIFAR-10 (f) ResNet50 on CIFAR-10

Figure 12: We conduct learning-rate grid searches for Muon, NorMuon, HTMuon, and HTMuon+NorMuon across various datasets and architectures, including LLaMA models on C4 and ResNet models on CIFAR-100 and CIFAR-10. We find that HTMuon outperforms Muon and NorMuon across most learning rates, demonstrating the effectiveness and robustness of our method.

heavy tails) and HTMuon(which introduces heavy tail spectral correction on top of preserved spectral information). Below, we discuss the benefits of merely inducing heavy tails versus preserving singular value information.

- **Merely inducing a heavy-tailed spectrum already yields non-trivial gains.** Our HTMuon_HT ablation enforces a rigid, prespecified heavy-tailed spectral distribution. In Table 12 of our paper, although this design discards the data-driven singular-value structure, HTMuon_HT still outperforms Muon in perplexity by 0.28 on LLaMA-60M and by 0.58 on LLaMA-135M. We would like to kindly emphasize that this is not "poor performance." First, Muon is a strong baseline: in Table 1, Muon outperforms the widely used Adam/AdamW by up to 3.21 PPL on LLaMA-60M and 135M. Under such a strong baseline, simply replacing the spectrum with a predefined heavy-tailed distribution that ignores data information yet still yields gains provides direct evidence that heavy-tail correction itself is effective. Second, the improvement of HTMuon_HT is not insignificant: as suggested by Table 2 in (Liu et al., 2025b) and (He et al., 2025) and Figure 2 in (Wen et al., 2025), after carefully tune the hyperparameters of the baselines on LLaMA/C4, an improvement of ≥ 0.2 PPL over Muon is generally regarded as non-negligible. For example, COSMOS outperforms Muon by 0.15 PPL for LLaMA-135M in (Liu et al., 2025b) and AlphaDecay outperforms Adam by 0.11 PPL for LLaMA-1B in (He et al., 2025). Therefore, these results strongly indicate that merely inducing heavy-tail correction is already useful. Moreover, in Figure 1 of our motivating experiment, we observe that Muon_NS outperforms Muon_SVD, this further suggests that strictly enforcing a light-tailed, unit-singular-value spectrum may be overly restrictive, and that relaxing it via heavy-tail correction can be beneficial.

- **Heavy tail spectral correction on top of preserved spectral information is more effective.** We acknowledge that the data-driven spectral information encoded in singular values is important, as it reflects meaningful structure learned from data. Our method HTMuon therefore performs heavy-tail spec-

Table 11: Detailed results for learning rate grid search on C4. Red indicates the best value, and blue denotes the second-best value.

	LLaMA-60M				LLaMA-135M			
Optimizer	0.01	0.02	0.03	0.04	0.01	0.02	0.02	0.04
Muon	28.80	28.99	28.88	29.93	22.23	22.51	22.59	23.73
NorMuon	28.63	28.59	28.17	29.07	21.99	22.17	22.06	22.76
HTMuon	28.48	28.75	**27.88**	30.22	22.95	21.96	**21.25**	21.92
HTMuon+NorMuon	30.14	28.63	**27.55**	28.03	22.71	21.82	**21.11**	21.76

Table 12: Detailed results for learning rate grid search on CIFAR-100. Red indicates the best value, and blue denotes the second-best value.

	ResNet18				ResNet50			
Optimizer	0.015	0.02	0.025	0.03	0.015	0.02	0.025	0.03
Muon	77.10	77.16	77.95	77.53	79.15	79.41	79.67	79.85
NorMuon	76.75	77.82	77.64	77.63	79.13	79.78	79.74	79.61
HTMuon	77.18	77.65	**78.24**	78.21	79.55	80.03	**80.16**	80.07
HTMuon+NorMuon	77.51	77.78	**78.51**	78.02	79.60	80.03	79.93	**80.22**

tral correction based on the current spectral information: specifically, we raise singular values to the power p (as motivated by Lemma 3.1) to give the spectrum a heavier tail than Muon's unit-spectrum, without discarding the learned geometry. This combination preserves directional information while attenuating noise-dominated directions more than signal-aligned ones compared to Muon, leading to stronger gains than using a rigid, pre-specified heavy-tailed spectrum alone.

D Detailed Hyperparameters

Since in practice our Muon is a hybrid MuonwithAdamW variant(Jordan et al., 2024), we use the same hyperparameter settings as AdamW for the small subset of parameters updated with AdamW.

D.1 Hyperparameter settings for LLM pretraining

For the C4 task, we adopt the LLaMA architecture specifications listed in Table 18 to ensure reproducibility and consistency with prior work. All model variants are trained with a uniform maximum sequence length of 256, a batch size of 512, and an aggregate of 13K tokens per batch. For LLaMA-60M, we run all experiments on two NVIDIA L40 GPUs or two NVIDIA A6000 GPUs without gradient accumulation; for LLaMA-135M, we run all experiments on four NVIDIA L40 GPUs or four NVIDIA A6000 GPUs without gradient accumulation; for LLaMA-350M, we run all experiments on four NVIDIA RTX PRO 6000 Blackwell without gradient accumulation; for LLaMA-1B, we run all experiments on four NVIDIA RTX PRO 6000 Blackwell with gradient accumulation.

For the OpenWebText task, we consider a uniform maximum sequence length of 1024 , a batch size of 480 and 10000 iterations for GPT-2 small. we run all experiments on four NVIDIA RTX PRO 6000 Blackwell with gradient accumulation=8. We do learning rate grid search on {0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1} for Muon and HTMuon, we set learning rate$= 6e-4$ for AdamW. We consider weight decay$= 0.1$ for all optimizers.

We apply $p = 0.125$ for all LLM pretraining experiments. **In our experiments, we apply Muon updates to the embedding and output layers during training.** For more detailed hyperparameter settings, please refer to Table 19 and 20.

D.2 Hyperparameter settings for Image classification

For training on CIFAR datasets, we set batch size $= 512$ and we set $p = 0.125$. We run all the experiments on one NVIDIA L40 GPU. For more detailed hyperparameter settings, please refer to Table 21 and 22.

For training on ImageNet-1K datasets, we set batch size $= 1024$ and we set $p = 0.03125$. We

Table 13: Detailed results for learning rate grid search on CIFAR-10. Red indicates the best value, and blue denotes the second-best value.

	ResNet18				ResNet50			
Optimizer	0.01	0.015	0.02	0.025	0.01	0.015	0.02	0.025
Muon	95.10	95.39	95.25	95.36	95.8	95.90	95.97	95.90
NorMuon	95.30	95.35	95.57	95.24	95.74	96.03	95.91	95.75
HTMuon	95.27	95.23	**95.63**	95.39	**96.13**	95.84	96.07	95.86
HTMuon+NorMuon	95.36	95.25	**95.40**	95.17	96.02	95.85	**96.04**	95.97

Table 14: Detailed results for ablation study on p on C4 dataset. Red indicates the best value, and blue denotes the second-best value.

	LLaMA-60M					LLaMA-135M				
Optimizer	0	0.1	0.125	0.2	0.25	0	0.1	0.125	0.2	0.25
HTMuon	28.80	27.91	**27.88**	28.56	29.97	22.23	21.33	**21.25**	21.66	22.43
HTMuon+NorMuon	28.17	**27.48**	27.55	28.15	29.54	21.99	21.21	**21.11**	21.33	22.1

run all the experiments on one NVIDIA RTX PRO 6000 Blackwell. We set learning rate for {0.003, 0.004, 0.005} for Adam, Muon, HTMuon.

E Baseline Optimizers

In this section, we provide the algorithms for all optimizers evaluated in our study. We adopt the following notation: w_t denotes the model parameters at step t, g_t the corresponding gradient, η the learning rate, λ the weight decay, β_1 and β_2 the moment decay rates, ϵ a numerical stability constant, g_{norm} the gradient norm, and m and v the first and second moments, respectively. All operations are element-wise unless stated otherwise.

NorMuon (Li et al., 2025): A Muon-based optimizer that applies normalized and orthogonalized gradient updates with shape-aware scaling. We put the implementations in Algorithm 8.

AdaMuon (Si et al., 2025): An adaptive variant of Muon that incorporates second-moment information into orthogonalized updates. We put the implementations in Algorithm 9.

MARS (Yuan et al., 2024): A momentum-based adaptive optimizer included as a representative adaptive baseline. We put the implementations in Algorithm 10.

SOAP (Vyas et al., 2024): An optimizer that applies stochastic orthogonalization or projection to gradient updates. We put the implementations in Algorithm 11.

Cautious (Liang et al., 2024): An optimizer that modifies update application in a conservative manner based on gradient information. We put the implementations in Algorithm 12.

COSMOS (Liu et al., 2025b): An optimizer that incorporates geometry-aware scaling into its parameter updates. We put the implementations in Algorithm 13.

GaLore (Zhao et al., 2024): A memory-efficient optimizer that performs low-rank gradient projection to reduce optimizer-state and update costs, enabling large-model training under limited GPU memory. We put the implementations in Algorithm 14.

Sophia (Liu et al., 2024a): : A second-order optimizer that uses a Hessian-based (diagonal) curvature estimate to precondition gradients and applies clipped updates for stability and efficiency in large-scale training. We put the implementations in Algorithm 15.

Table 15: Detailed results for ablation study on p for HTMuon on CIFAR datasets. Red indicates the best value.

	ResNet18					ResNet50				
Dataset	0	0.1	0.125	0.2	0.25	0	0.1	0.125	0.2	0.25
CIFAR-10	95.39	95.14	**95.63**	95.46	95.49	95.97	95.88	**95.95**	95.95	95.95
CIFAR-100	77.95	77.65	78.24	**78.58**	78.32	79.67	79.40	80.16	**80.30**	80.27

Table 16: Detailed results for ablation study on p for HTMuon on OpenWebText dataset. Red indicates the best value.

Optimizer	0	0.1	0.125	0.2	0.25
HTMuon	22.46	22.40	**22.20**	22.49	22.83

Table 17: Detailed results for HTMuon_HT. Red indicates the best value, and blue denotes the second-best value. We use $\alpha = 0.25$ and learning rate $= 0.03$ in Algorithm 7 for experiments.

	Muon	HTMuon	NorMuon	HTMuon+NorMuon	HTMuon_HT	HTMuon_HT+NorMuon
LLaMa-60M	28.8	27.88	28.17	**27.55**	28.52	28.02
LLaMa-135M	22.23	21.25	21.99	**21.11**	21.65	21.26

Table 18: Hyperparameters of LLaMA models.

Params	Hidden	Intermediate	Heads	Blocks	Steps	Data amount	Batch Size
60M	512	1376	8	8	10K	1B	512
135M	768	2048	12	12	20K	2B	512
350M	1024	2736	16	24	60K	6B	512
1B	2048	5461	32	24	90K	9B	512

Table 19: Hyperparameters for LLaMA-60M and LLaMA-135M On C4 Dataset. The boldfaced values denote the optimal hyperparameters.

	LLaMA-60M		LLaMA-135M	
Optimizer	LR	WD	LR	WD
Adam	**1e-3**	1e-5	**1e-3**	1e-5
AdamW	**1e-3**	0.1	**1e-3**	0.1
Muon	{**0.01**,0.02,0.03,0.04}	0.1	{**0.01**,0.02,0.03,0.04}	0.1
NorMuon	{0.01,0.02, **0.03**,0.04}	0.1	{**0.01**,0.02,0.03,0.04}	0.1
AdaMuon	{6e-4, 1e-3, **5e-3**, 1e-2}	0.1	{ 1e-3, 5e-3, **1e-2**, 2e-2}	0.1
MARS	{1e-3, 2e-3, **3e-3**, 4e-3}	0.1	{1e-3, **2e-3**, 3e-3, 4e-3}	0.1
SOAP	{2e-3, **3e-3**, 4e-3}	0.01	{ 2e-3, **3e-3**, 4e-3}	0.01
Cautious	{1e-3, 2e-3, 3e-3, **4e-3**}	0	{1e-3, **2e-3**, 3e-3, 4e-3}	0
COSMOS	{1e-3, 2e-3, **3e-3**, 4e-3}	0.1	{1e-3, **2e-3**, 3e-3, 4e-3}	0.1
GaLore	{5e-3, 1e-2, **2e-2**, 3e-2, 4e-2}	0	{5e-3, 1e-2, **2e-2**, 3e-2, 4e-2}	0
Sophia	{1e-4, 2e-4,3e-4, **4e-4**, 5e-4, 6e-4}	0.1	{1e-4, **2e-4**,3e-4, 4e-4, 5e-4, 6e-4}	0.1
HTMuon	{0.01,0.02,**0.03**,0.04}	0.1	{0.01,0.02,**0.03**,0.04}	0.1
HTMuon+NorMuon	{0.01,0.02,**0.03**,0.04}	0.1	{0.01,0.02,**0.03**,0.04}	0.1

Table 20: Hyperparameters for LLaMA-350M and LLaMA-1B On C4 Dataset. The boldfaced values denote the optimal hyperparameters.

	LLaMA-350M		LLaMA-1B	
Optimizer	LR	WD	LR	WD
Adam	**1e-3**	1e-5	**6e-4**	1e-6
AdamW	**1e-3**	0.1	**6e-4**	0.1
Muon	{0.0025, **0.005**,0.01,0.015}	0.1	{**0.005**,0.01}	0.1
HTMuon	{0.0025, **0.005**,0.01,0.015}	0.1	{**0.005**,0.01}	0.1

Table 21: Hyperparameters for ResNet18 and ResNet50 On CIFAR-100 Dataset. The boldfaced values denote the optimal hyperparameters.

	ResNet18		ResNet50	
Optimizer	LR	WD	LR	WD
SGDM	{0.1,0.2,**0.3**,0.4}	5e-4	{0.1,0.2,**0.3**,0.4}	5e-4
Muon	{0.015,0.02,**0.025**,0.03}	0.1	{0.015,0.02,0.025,**0.03**}	0.1
NorMuon	{0.015,**0.02**,0.025,0.03}	0.1	{0.015,**0.02**,0.025,0.03}	0.1
HTMuon	{0.015,0.02,**0.025**,0.03}	0.1	{0.015,0.02,**0.025**,0.03}	0.1
HTMuon+NorMuon	{0.015,0.02,**0.025**,0.03}	0.1	{0.015,0.02, 0.025, **0.03**}	0.1

Table 22: Hyperparameters for ResNet18 and ResNet50 On CIFAR-10 Dataset. The boldfaced values denote the optimal hyperparameters.

	ResNet18		ResNet50	
Optimizer	LR	WD	LR	WD
SGDM	{0.1,**0.2**,0.3,0.4}	5e-4	{0.1,0.2,**0.3**,0.4}	5e-4
Muon	{0.01,**0.015**,0.02,0.025}	0.1	{0.01,0.015,**0.02**,0.025}	0.1
NorMuon	{0.01,0.015,**0.02**,0.025}	0.1	{0.01,**0.015**,0.02,0.025}	0.1
HTMuon	{0.01,0.015,**0.02**,0.025}	0.1	{**0.01**,0.015,0.02,0.025}	0.1
HTMuon+NorMuon	{0.01,0.015,**0.02**,0.025}	0.1	{0.01,0.015,**0.02**,0.025}	0.1

Algorithm 3 Muon

Input: Initial weights $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, loss function L, learning rate η, momentum parameter β, weight decay λ.
Initialize $\mathbf{M}_0 \in \mathbb{R}^{m\times n} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  $\mathbf{G}_t \leftarrow \nabla_{\mathbf{W}} L(\mathbf{W}_t)$
  $\mathbf{M}_t \leftarrow \beta \mathbf{M}_{t-1} + (1-\beta)\mathbf{G}_t$
  $\boldsymbol{O}_t \leftarrow$ NewtonSchulz5$(\mathbf{M}_t)$
  $s \leftarrow \sqrt{\max\left(1, \frac{m}{n}\right)}$
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\lambda\mathbf{W}_t - \eta s \boldsymbol{O}_t$
end for

Algorithm 4 Muon_SVD

Input: Initial weights $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, loss function L, learning rate η, momentum parameter β, weight decay λ.
Initialize $\mathbf{M}_0 \in \mathbb{R}^{m\times n} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  $\mathbf{G}_t \leftarrow \nabla_{\mathbf{W}} L(\mathbf{W}_t)$
  $\mathbf{M}_t \leftarrow \beta \mathbf{M}_{t-1} + (1-\beta)\mathbf{G}_t$
  $\boldsymbol{U}_t, \boldsymbol{\Sigma}_t, \boldsymbol{V}_t^\top \leftarrow \mathtt{SVD}(\mathbf{M}_t)$
  $\boldsymbol{O}_t \leftarrow \boldsymbol{U}_t \boldsymbol{V}_t^\top$
  $s \leftarrow \sqrt{\max\left(1, \frac{m}{n}\right)}$
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\lambda \mathbf{W}_t - \eta s \boldsymbol{O}_t$
end for

Algorithm 5 HTMuon_NS

Input: Initial weights $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, loss function L, learning rate η, momentum parameter β, weight decay λ, power p.
Initialize $\mathbf{M}_0 \in \mathbb{R}^{m\times n} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  $\mathbf{G}_t \leftarrow \nabla_{\mathbf{W}} L(\mathbf{W}_t)$
  $\mathbf{M}_t \leftarrow \beta \mathbf{M}_{t-1} + (1-\beta)\mathbf{G}_t$
  $\boldsymbol{O}'_t \leftarrow \mathtt{NS5}(\mathbf{M}_t)$
  $\boldsymbol{O}''_t \leftarrow \mathtt{NS\_2p\_root}(\mathbf{M}_t^\top \mathbf{M}_t)$
  $\boldsymbol{O}_t \leftarrow \boldsymbol{O}'_t \boldsymbol{O}''_t$
  $s \leftarrow \sqrt{\max\left(1, \frac{m}{n}\right)}$
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\lambda \mathbf{W}_t - \eta s \boldsymbol{O}_t$
end for

Algorithm 6 NS_root

Input: Positive definite matrix $\boldsymbol{X}^{n\times n}$, small constant ε, power p, ns_steps T.
for $t = 1, 2, \ldots, [\log_2 \frac{2}{p}]$ **do**
  $\alpha \leftarrow \|X\|_{\mathrm{F}}$
  $\boldsymbol{X} \leftarrow \frac{\boldsymbol{X}}{\|X\|_{\mathrm{F}}+\varepsilon}$
  $\boldsymbol{Y} \leftarrow \boldsymbol{X}, \boldsymbol{Z} \leftarrow \boldsymbol{I}$
  for $i = 1, 2, \ldots, T$ **do**
    $\boldsymbol{Q} \leftarrow 3.0\boldsymbol{I} - \boldsymbol{Z}\boldsymbol{Y}$
    $\boldsymbol{Y} \leftarrow 0.5\boldsymbol{Y}\boldsymbol{Q}$
    $\boldsymbol{Z} \leftarrow 0.5\boldsymbol{Q}\boldsymbol{Z}$
  end for
  $\boldsymbol{X} \leftarrow \sqrt{\alpha}\boldsymbol{Y}$, $\boldsymbol{X} \leftarrow \frac{\boldsymbol{X}+\boldsymbol{X}^\top}{2} + \epsilon\boldsymbol{I}$
end for

Algorithm 7 HTMuon_HT

Input: Initial weights $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, loss function L, learning rate η, momentum parameter β, weight decay λ, power α.
Initialize $\mathbf{M}_0 \in \mathbb{R}^{m\times n} \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  $\mathbf{G}_t \leftarrow \nabla_{\mathbf{W}} L(\mathbf{W}_t)$
  $\mathbf{M}_t \leftarrow \beta \mathbf{M}_{t-1} + (1-\beta)\mathbf{G}_t$
  $\boldsymbol{U}_t, \boldsymbol{\Sigma}_t, \boldsymbol{V}_t^\top \leftarrow \mathsf{SVD}(\mathbf{M}_t)$
  $\boldsymbol{\Sigma}_{tii} \leftarrow i^{-\alpha}$
  $\boldsymbol{O}_t \leftarrow \boldsymbol{U}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^\top$
  $s \leftarrow \sqrt{\max\left(1, \frac{m}{n}\right)}$
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\lambda \mathbf{W}_t - \eta s \boldsymbol{O}_t$
end for

Algorithm 8 NorMuon

Input: Initial weights $\mathbf{W}_0 \in \mathbb{R}^{m\times n}$, loss function L, learning rate η, momentum parameters (β_1, β_2), perturbation parameter ε, weight decay λ.
Initialize momentum $\mathbf{M}_0 \in \mathbb{R}^{m\times n} \leftarrow \mathbf{0}$, second moment $\mathbf{v}_0 \in \mathbb{R}^m \leftarrow \mathbf{0}$
for $t = 1, 2, \ldots$ **do**
  $\mathbf{G}_t \leftarrow \nabla_{\mathbf{W}} L(\mathbf{W}_t)$
  $\mathbf{M}_t \leftarrow \beta_1 \mathbf{M}_{t-1} + (1-\beta_1)\mathbf{G}_t$
  $\mathbf{O}_t \leftarrow \mathrm{NS5}(\mathbf{M}_t)$
  $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\,\mathrm{mean}_{\text{cols}}(\mathbf{O}_t \odot \mathbf{O}_t)$
  $\mathbf{V}_t \leftarrow \mathrm{ExpandRows}(\mathbf{v}_t)$ $(\mathbf{V}_t \in \mathbb{R}^{m\times n})$
  $\widehat{\mathbf{O}}_t \leftarrow \mathbf{O}_t \oslash (\sqrt{\mathbf{V}_t} + \varepsilon)$
  $\hat{\eta} \leftarrow 0.2\,\eta\,\sqrt{mn}/\|\widehat{\mathbf{O}}_t\|_F$
  $\mathbf{W}_{t+1} \leftarrow \mathbf{W}_t - \eta\lambda \mathbf{W}_t - \hat{\eta}\,\widehat{\mathbf{O}}_t$
end for

Algorithm 9 AdaMuon

Input: Initial 2D-weights $\mathbf{W}_0 \in \mathbb{R}^{n\times m}$, loss function $\mathcal{L}$, learning rate η, weight decay λ, momentum β, Newton–Schulz steps T, small constant ε.
Output: Updated weights $\mathbf{W}$
Initialize first-momentum $\mathbf{M}_0 \leftarrow \mathbf{0}$, second-momentum $\mathbf{V}_0 \leftarrow \mathbf{0}$
for each iteration $t = 1, 2, \ldots$ **do**
  Compute gradient: $\mathbf{G}_t = \nabla_{\mathbf{W}_t} \mathcal{L}(\mathbf{W}_t)$
  Update first momentum: $\mathbf{M}_t = \beta \cdot \mathbf{M}_{t-1} + \mathbf{G}_t$
  Compute sign-stabilized orthogonal direction: $\mathbf{O}_t = \text{Newton–Schulz}(\mathrm{Sign}(\mathbf{M}_t),\, T)$
  Update second momentum: $\mathbf{V}_t = \beta \cdot \mathbf{V}_{t-1} + (1-\beta) \cdot (\mathbf{O}_t \odot \mathbf{O}_t)$
  Apply second momentum update: $\hat{\mathbf{O}}_t = \mathbf{O}_t \oslash (\sqrt{\mathbf{V}_t} + \varepsilon \cdot \mathbf{1})$
  RMS-aligned: $\gamma_t = \frac{0.2 \cdot \sqrt{mn}}{\|\hat{\mathbf{O}}_t\|_F}$
  Update weights: $\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \left(\gamma_t \hat{\mathbf{O}}_t + \lambda \mathbf{W}_t\right)$
end for

Algorithm 10 MARS

Hyperparameters: $\beta_1,\ \beta_2,\ \gamma,\ \epsilon,\ \eta,\ \lambda,\ g_{norm}$
State: $m,\ v,\ g_{t-1}$
for $t = 1, 2, \ldots$ **do**
  $c_t = g_t + \gamma \frac{\beta_1}{1-\beta_1}(g_t - g_{t-1})$
  $\hat{c}_t = c_t \max\{1, \frac{g_{norm}}{\|c_t\|_2}\}$,
  $m_t = \beta_1 m_{t-1} + (1-\beta_1)\hat{c}_t$
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\hat{c}_t^2$
  $\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$
  $w_{t+1} = w_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} - \eta \lambda w_t.$
end for

Algorithm 11 SOAP

Hyperparameters: $\beta_1,\ \beta_2,\ \mu,\ k,\ \epsilon$, block_size, g_{norm}
Partition all parameters into block_size × block_size
Update Rules for Each Block:
for $t = 1, 2, \ldots$ **do**
  $\hat{g}_t = g_t \max\{1, \frac{g_{norm}}{\|g_t\|_2}\}$
  $\hat{g}_t = Q_A \hat{g}_t Q_B$
  $m_t = \beta_1 m_{t-1} + (1-\beta_1)\hat{g}_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2)\hat{g}_t^2$
  $\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \hat{v}_t = \frac{v_t}{1-\beta_2^t}$
  $w_{t+1} = w_t - \eta_t Q_A^\top \Big(\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}\Big) Q_B^\top$
  $G_A = \mu G_A + (1-\mu)\hat{g}_t \hat{g}_t^\top + \epsilon I, \quad G_B = \mu G_B + (1-\mu)\hat{g}_t^\top \hat{g}_t + \epsilon I$
  if $t \bmod k = 0$: $\quad Q_A = \mathrm{QR}(G_A Q_A), \quad Q_B = \mathrm{QR}(G_B Q_B)$
end for

Algorithm 12 Cautious

Hyperparameters: $\beta_1,\ \beta_2,\ \epsilon,\ \eta,\ \lambda,\ g_{norm}$
State: $m,\ v$
for $t = 1, 2, \ldots$ **do**
  $\hat{g}_t = g_t \max\{1, \frac{g_{norm}}{\|g_t\|_2}\}$
  $m_t = \beta_1 m_{t-1} + (1-\beta_1)\hat{g}_t$
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\hat{g}_t^2$
  $\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$
  $u_t = \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}, \quad s_t = \mathbb{I}\big(u_t \cdot \hat{g}_t > 0\big)$
  $\hat{u}_t = \frac{u_t \cdot s_t}{\mathrm{mean}(s_t)}, \quad w_{t+1} = w_t - \eta \hat{u}_t - \eta \lambda w_t$
end for

Algorithm 13 COSMOS

For an $m \times n$ layer W, the algorithm maintain four matrices: $U \in \mathbb{R}^{n\times r}$, $S \in \mathbb{R}^{r\times r}$, $V \in \mathbb{R}^{m\times r}$, and $M \in \mathbb{R}^{m\times n}$ per layer.

Input: Learning rate η, combination weight γ, projection rank $r \ll n$, momentum parameters (β_1, β_2), perturbation parameter ϵ. For simplicity, we omit the initialization.

for $t = 0, \ldots$ **do**

  Sample batch $\mathcal{M}_t$

  $G_t \leftarrow \nabla_W \phi_{\mathcal{M}_t}(W_t)$

  $M_t \leftarrow \beta_1 M_{t-1} + (1-\beta_1) G_t$

  $U_t \leftarrow \mathsf{QR}\big(\beta_2 U_{t-1} S_{t-1} + (1-\beta_2) G_t^\top G_t U_{t-1}\big)$

  $S_t \leftarrow U_t^\top \left(\beta_2 U_{t-1} S_{t-1} U_{t-1}^\top + (1-\beta_2) G_t^\top G_t\right) U_t$

  $V_t \leftarrow \beta_2 V_{t-1} + (1-\beta_2)(G_t U_t) \odot (G_t U_t)$

  $$A_t = \left(\frac{M_t U_t / (1-\beta_1^t)}{\sqrt{(V_t+\epsilon)/(1-\beta_2^t)}} \right) U_t^\top$$

  $$B_t \leftarrow \mathsf{NORM}\left(\mathsf{NS5}\left(\frac{M_t - M_t U_t U_t^\top}{\|M_t - M_t U_t U_t^\top\|_{\mathrm{F}}} \right) \right)$$

  $\tilde{G}_t \leftarrow A_t + \gamma \cdot B_t \cdot \sqrt{m}$

  $W_{t+1} \leftarrow W_t - \eta \cdot \mathsf{NORM}(\tilde{G}_t) \cdot \sqrt{m}$

end for

Algorithm 14 GaLore

For an $m \times n$ layer W, the algorithm maintains a rank-r subspace projector (left or right, depending on the shape) and Adam moments in the projected space.
Input: Learning rate η, scale factor α, projection rank r, subspace change frequency T, Adam parameters (β_1, β_2), numerical constant ϵ. For simplicity, we omit the initialization.
for $t = 0, 1, \ldots$ **do**
    Sample batch $\mathcal{B}_t$
    $G_t \leftarrow \nabla_W \phi_{\mathcal{B}_t}(W_t)$
    if $t \bmod T = 0$ **then**
        $U, \Sigma, V \leftarrow \texttt{SVD}(G_t)$
        if $m \leq n$ **then**
            $P_t \leftarrow U_{[:,1:r]}$ {left projector}
        else
            $Q_t \leftarrow V_{[:,1:r]}$ {right projector}
        end if
    else
        if $m \leq n$ **then**
            $P_t \leftarrow P_{t-1}$ {reuse projector}
        else
            $Q_t \leftarrow Q_{t-1}$ {reuse projector}
        end if
    end if
    if $m \leq n$ **then**
        $R_t \leftarrow P_t^\top G_t$ {project to compact space, $R_t \in \mathbb{R}^{r \times n}$}
    else
        $R_t \leftarrow G_t Q_t$ {project to compact space, $R_t \in \mathbb{R}^{m \times r}$}
    end if
    UPDATE(R_t) by Adam in the projected space:
    $M_t \leftarrow \beta_1 M_{t-1} + (1 - \beta_1) R_t$
    $V_t \leftarrow \beta_2 V_{t-1} + (1 - \beta_2)(R_t \odot R_t)$
    $\hat{M}_t \leftarrow M_t / (1 - \beta_1^{t+1})$
    $\hat{V}_t \leftarrow V_t / (1 - \beta_2^{t+1})$
    $N_t \leftarrow \hat{M}_t / (\sqrt{\hat{V}_t} + \epsilon)$
    if $m \leq n$ **then**
        $\tilde{G}_t \leftarrow \alpha \cdot P_t N_t$ {project back to original space}
    else
        $\tilde{G}_t \leftarrow \alpha \cdot N_t Q_t^\top$ {project back to original space}
    end if
    $W_{t+1} \leftarrow W_t - \eta \cdot \tilde{G}_t$
end for

Algorithm 15 Sophia

Input: Parameters θ_1, learning rates $\{\eta_t\}_{t=1}^{T}$, hyperparameters $\lambda, \gamma, \beta_1, \beta_2, \epsilon$, curvature estimator choice $\texttt{Estimator} \in \{\texttt{Hutchinson}, \texttt{Gauss-Newton-Bartlett}\}$.
Set $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, and $h_{1-k} \leftarrow 0$.
for $t = 1, \ldots, T$ **do**
    Compute minibatch loss $L_t(\theta_t)$.
    $g_t \leftarrow \nabla L_t(\theta_t)$.
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)\, g_t$.
    if $t \bmod k = 1$ **then**
        $\hat{h}_t \leftarrow \texttt{Estimator}(\theta_t)$.
        $h_t \leftarrow \beta_2 h_{t-k} + (1 - \beta_2)\, \hat{h}_t$.
    else
        $h_t \leftarrow h_{t-1}$.
    end if
    $\theta_t \leftarrow \theta_t - \eta_t \lambda \theta_t$ {weight decay}
    $\theta_{t+1} \leftarrow \theta_t - \eta_t \cdot \texttt{clip}\Big(\frac{m_t}{\max\{\gamma \cdot h_t,\ \epsilon\}},\ 1\Big)$.
end for