OPT2025: 17th Annual Workshop on Optimization for Machine Learning

What really matters in matrix-whitening optimizers?

Kevin Frans KVFRANS@BERKELEY.EDU
Pieter Abbeel
Sergey Levine
UC Berkeley

Abstract

A range of recent optimizers have emerged that approximate the same *matrix-whitening* transformation in various ways. In this work, we systematically deconstruct such optimizers, aiming to disentangle the key components that explain performance. Across tuned hyperparameters across the board, all flavors of matrix-whitening methods reliably outperform elementwise counterparts, such as Adam. Matrix-whitening is often related to spectral descent – however, experiments reveal that performance gains are *not explained solely by accurate spectral normalization* – particularly, SOAP displays the largest per-step gain, even though Muon more accurately descends along the steepest spectral descent direction. Instead, we argue that matrix-whitening serves *two* purposes, and the *variance adaptation* component of matrix-whitening is the overlooked ingredient explaining this performance gap. Experiments show that variance-adapted versions of optimizers consistently outperform their sign-descent counterparts, including an adaptive version of Muon.

1. Introduction

In recent years, increasing growth in the scale of neural networks has resulted in a strong need to understand how neural networks can be trained efficiently. The workhorse of modern deep learning, gradient descent, has proven extensively scalable yet remains an inherently iterative process. By gaining a deeper understanding of such processes through both theoretical and empirical reconciliation, the field may continue the steady march in improving neural network training.

A range of recent optimizers have emerged that share a similar *matrix-whitening* transformation [7, 12, 32]. While differing in their exact approximations, such optimizers can generally be derived from the same core principles [3]. However, the concrete algorithms proposed have often contained auxiliary implementation details or unevenly tuned hyperparameters, potentially obscuring the root cause of the performance gain [24, 31, 34].

In this work, we systematically deconstruct such optimizers, aiming to disentangle the key components that explain performance. We establish a controlled experimental setup, with an explicit emphasis on breaking down methods into their constituent parts. We conduct a thorough sweep over four key hyperparameters, noting that optimal learning rate and weight decay parameters vary greatly across optimizer flavors. When all methods are tuned, we confirm that matrix-whitening optimizers reliably outperform elementwise transformations like Adam by a nontrivial margin.

However, the story *comparing* matrix-whitening optimizers is less clear. Empirically in our setting, SOAP [30] displayed the largest per-step gain in performance, outperforming Muon [14]. In an effort to understand the cause of these gains, we consider a hypothesis that the strength of matrix-whitening comes from its interpretation as steepest spectral descent [3], and that Shampoo-style explicit matrix inversion provides a more accurate spectral normalization than approximate

© K. Frans, P. Abbeel & S. Levine.

Newton-Schulz iteration. However, metrics show that Muon-style methods result in a *tighter* spread of singular values than SOAP, leading to a conclusion that *performance gains are not explained solely by accurate spectral normalization*.

In contrast, we argue that matrix-whitening serves *two* purposes—both spectral normalization and *variance adaptation*, and this variance adaptation aspect of whitening is a crucial, and often overlooked, ingredient in achieving strong performance. We identify three optimizer pairs that utilize the same spectral transformation, but opt to use signed descent vs. variance-adapted descent – Signum vs. Adam, SPlus vs. SOAP, and Muon vs. AdaMuon. In all cases, the variance-adapted versions result in superior performance difference *almost equal to the gap between Adam and Muon*.

Our main contributions in this work are in establishing a controlled experimental framework for comparing optimizer flavors, and in the use of this framework to identify variance-adaptation as an critical ingredient. We further detail this claim through a thorough ablation of variance-adaptation across three matrix-whitening optimizer families. We *do not claim* that the spectral-descent view of matrix-whitening is incorrect, rather, we show that spectral normalization is consistently effective, but argue it is not the full picture, and pure orthogonalization methods – such as Muon, Dion [1] and Polargrad [17], among others – can be further improved. We hope our findings encourage the study of optimizer flavors in terms of interchangeable components rather than entirely separate methods.

Code for replication: `https://github.com/kvfrans/matrix-whitening`.

See the full paper for more details: `https://arxiv.org/abs/2510.25000`.

2. Background

Gradient descent on non-Euclidean metrics. Gradient descent can be seen as solving for a trade-off between linear improvement and a distance penalty over parameters. While standard gradient descent assumes a Euclidean distance over parameters, we can generally represent second-order distances using a symmetric positive-definite metric M, with an analytic solution of:

$$u = \text{argmin}_{\Delta\theta} \underbrace{-g^T\Delta\theta}_{\text{Improvement}} + \underbrace{(1/2)\Delta\theta^T M \Delta\theta}_{\text{Distance Penalty}} \quad = \quad M^{-1}g, \tag{1}$$

where $g = \nabla_\theta L(\theta)$ and M^{-1} is sometimes referred to as a *preconditioner*.

Whitening metric. While there are many possible distance metrics to descend on, many recent optimizers have converged on a specific metric in particular, which we refer to as the *whitening* metric following [32]. Mechanically, the whitening metric can be written as the square-root uncentered covariance of incoming gradients:

$$M_{\text{Whitening}} = \mathbb{E}_{x,y}\left[\nabla_\theta L(\theta,x,y)\nabla_\theta L(\theta,x,y)^T\right]^{1/2} = \mathbb{E}_{x,y}\left[gg^T\right]^{1/2}. \tag{2}$$

Prior works have examined the relation of the whitening metric to the Hessian and to the Fisher information matrix, for which we defer to previous discussion [16]. Adam [15] can be understood as utilizing an *elementwise* approximation to the whitening metric, resulting in an efficient update where $m = diag(M)$:

$$m = E_{x,y}\left[g^2\right]^{-1/2} \qquad u = g/m. \tag{3}$$

Matrix-based whitening. Two powerful connections appear when we accept that in neural networks, parameters are structured *matrices* rather than an arbitrary set. First, we can represent the per-layer whitening metric in terms of its Kronecker factors. Second, we can precondition via the inverted Kronecker factors directly, without ever actually forming the full product. This results in the following matrix-form whitening update for dense layers utilized by the Shampoo [12] family:

$$E_{x,y}[gg^T]^{-1/2}g \quad \leftarrow \text{approx.} \rightarrow \quad E_{x,y}[GG^T]^{-1/4}\, G\, E_{x,y}[G^TG]^{-1/4} \tag{4}$$

Second, if we ignore the expectation, the term above is equivalent to the *orthogonalization* of G [3, 6, 7, 17, 28]. This relation can be derived by rewriting G as its singular-value decomposition, $G = U\Sigma V^T$:

$$(GG^T)^{-1/4}\, G\, (G^TG)^{-1/4} = (U\Sigma^2U^T)^{-1/4}\, U\Sigma V^T\, (V\Sigma^2V^T)^{-1/4} = UV^T, \tag{5}$$

and is the solution to steepest descent under the *spectral norm* of the matrix.

A range of optimizer families – such as PSGD, Shampoo, and Muon – can be seen as approximating the above behaviors, and we refer to these as **matrix-whitening** methods. While similar in motivation, these families differ in their core algorithmic decisions, and we will take a step towards disentangling these choices in the following section.

3. A Brief Benchmark

We now conduct an empirical study of optimizers that approximate the matrix-whitening update. In our experimental setting, we train a standard GPT-2 architecture Transformer [5, 29] on a next-token prediction language modelling objective with the OpenWebText dataset [11]. The model follows the "Base" size architecture and has 162M total parameters. We train for 10,000 gradient steps on a batch of 1024 sequences of length 256, which is roughly a 1x Chinchilla ratio [13]. We use a fixed warmup of 200 steps and a cosine learning rate schedule afterwards.

The primary aim of this comparison is to remove confounding factors and examine only the core differences between each optimizer. Thus, we ensure that each trial uses the same data ordering, random seed, and initial parameters. For nonstandard parameters (i.e. layer norm scales and input/output heads), we update using a separate Adam optimizer with fixed tuned hyperparameters. Whenever possible, we disregard auxiliary design choices in each algorithm (e.g. learning rate grafting, Nesterov momentum, or iterate averaging) and focus on the core whitening behavior.

Importantly, we sweep over four key hyperparameters – learning rate, weight decay, momentum coefficient β_1, and variance coefficient β_2 – and do so independently for each method. We sweep learning rate within a resolution of $10^{1/8} \approx 1.32$, weight decay within $10^{1/4} \approx 1.78$, β_1 within a half-life of $10^{1/4} \approx 1.78$, and β_2 within a half-life of $10^{1/2} \approx 3.15$. All methods are tuned to within a local optimum of these hyperparameters as displayed in Figure 3.

We benchmark the performance of the following optimizers, choosing method-specific settings that lead to the strongest performance when computationally reasonable:

- **Adam** [15], a baseline optimizer that is the current standard for training deep neural networks. Updates are normalized by an elementwise second moment buffer.
- **Signum** [4], which updates via elementwise sign rather than normalizing by second-moment.
- **Shampoo** [12, 26], a matrix optimizer which explicitly tracks Kronecker factors as in Equation (4). Every N gradient steps, the left and right preconditioners are calculated by raising each factor to the $-(1/4)$ matrix power, and this result is cached until the next recalculation. We consider $N \in \{10, 100\}$.

Method	LR	WD	β_1	β_2	Walltime	Adam Steps	Val Loss
Adam	0.001	1.0	0.95	0.99	1.0	1.0	2.982 $\pm .008$
Signum	0.000177	3.162	0.9	-	1.0	> 1.0	3.006 $\pm .008$
PSGD	0.000264	0.001	0.968	-	4.8	0.95 - 1.0	2.973 $\pm .006$
Shampoo-10	0.00132	1.0	0.9	0.99	3.2	0.80 - 0.83	2.963 $\pm .004$
SPlus-100	0.1	0.01	0.99	0.968	1.3	0.80 - 0.83	2.962 $\pm .007$
SPlus-10	0.1	0.01	0.99	0.99	3.2	0.77 - 0.80	2.954 $\pm .007$
SOAP-100	0.00175	0.316	0.9	0.99	1.2	0.71 - 0.74	**2.946** $\pm .003$
SOAP-10	0.00311	0.316	0.968	0.99	3.1	0.66 - 0.68	**2.939** $\pm .003$
Muon	0.00770	0.1	0.9	-	1.07	0.80 - 0.83	2.964 $\pm .005$
AdaMuon	0.000312	3.162	0.968	0.99	1.07	0.74 - 0.77	**2.950** $\pm .003$

Table 1: Under optimal hyperparameters, **matrix-whitening methods outperform Adam**. The highest per-step performance is achieved by SOAP, followed by AdaMuon which strikes a strong balance between wallclock time and final validation loss.

- **SOAP** [30], a variant of Shampoo where updates are rotated onto the *eigenbasis* of the left/right factors. In this rotated space, the updates are normalized via an elementwise uncentered variance (i.e. an inner Adam update), then rotated back.
- **SPlus** [10], which similarly to SOAP rotates updates onto the eigenbasis, but takes the elementwise sign rather than normalizing by an explicit second moment buffer.
- **Muon** [14], which orthogonalizes updates via Newton-Shulz iteration, and can be seen as descending under the spectral norm (Equation (5)).
- **AdaMuon** [27], a variant on Muon where a variance buffer is estimated over *post*-orthogonalized updates, and is used for elementwise normalization. We use a simplified form of the original algorithm that does not use the pre-NS sign transformation. Also concurrently proposed as NorMuon [20].
- **PSGD (Fisher-Kron)** [18, 19], which keeps track of a left/right preconditioner that is learned via iterative gradient descent. We update the precondioner at every step.

For all optimizers, preconditioning is performed on a momentum buffer, as is standard practice.

As shown in Table 1, the considered set of optimizers outperform Adam across the board, reaching an equivalent validation loss within **66% to 83%** of the gradient steps for the Shampoo and Muon families. We report a margin of error as the difference within our smallest hyperparameter search resolution, and note that the gap between optimizer flavors is an order-of-magnitude higher. Notably, the gains in performance from utilizing a more performant optimizer are consistent even when considering sub-optimal hyperparameters, e.g. Muon with a 2x greater-than-optimal learning rate remains stronger than Adam with the equivalent adjustment, as shown in Figure 3.

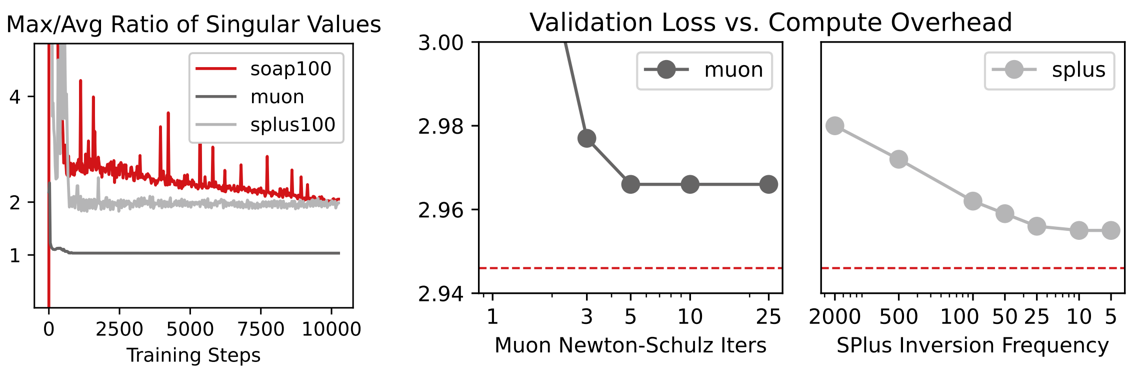

Figure 1: **Left: Muon descends under the spectral norm more accurately than SOAP or SPlus**, as all singular values in the update are near ± 1. In contrast, the Shampoo-style methods perform this only loosely, with a ratio between 2 to 3. Adam results in a ratio of ≈ 12 (not plotted). **Right: Even with increased computation, Muon or SPlus do not reach the empirical performance of SOAP.** For Muon, we increase the number of Newton-Schulz iterations at each step. For SPlus, we increase the frequency of updating the eigenbasis. The red dotted line represents the performance of SOAP-100.

4. Performance gains are not explained solely by accurate spectral normalization

In our experimental setting, SOAP displays the largest per-step gain in performance and outperforms other optimizer flavors. What reasons may explain this performance?

One notable comparison is between SOAP and Muon, as the two optimizers utilize different computational strategies to perform the matrix-whitening operation. SOAP keeps a historical average of the left and right second moments, then uses an explicit solver to locate the eigenbasis (we use `eigh` in our implementation). In contrast, Muon utilizes a Newton-Schulz iteration to implicitly orthogonalize the momentum buffer, aiming to set all singular values to ± 1.

A reasonable hypothesis is that the approximate nature of the Newton-Schulz iteration is not as effective as the explicit eigendecomposition used in Shampoo-style methods. To investigate this claim, we log both the maximum singular value (i.e. spectral norm) and the average singular value of updates, visualized in Figure 1 (left). As expected, the gap between the maximum and average singular values is largest in Adam, around ≈ 12. However, in comparison to SOAP which ranges from 2 to 3, *Muon achieves a tighter spread in its singular values, with a ratio very close to 1*. In other words, **even though Muon achieves a more accurate solution to the steepest descent direction under the spectral norm (Equation (5)), SOAP results in a stronger final performance**.

Additionally, we find in Figure 1 (right) that Muon and SPlus cannot reach the performance of SOAP even with additional computational budget for the optimizer. Specifically, we increase the amount of Newton-Schulz iterations in Muon, and the frequency of matrix-inversions in SPlus, and find that gains from a more accurate preconditioner plateau.

Together, these observations lead us to believe that faithfully descending along the spectral norm may not be the optimal behavior for a matrix-whitening optimizer. Instead, are there other aspects of matrix-whitening that may be equally important?

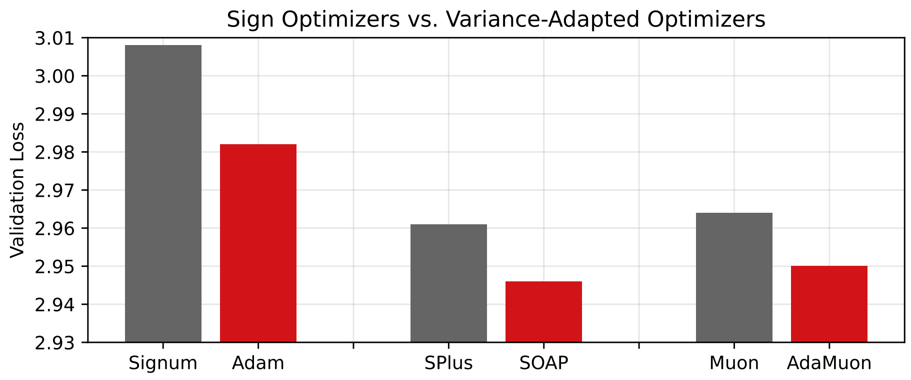

Figure 2: **Variance-adapted variants of optimizers outperform their strictly signed-descent counterparts.** Variance adaptation can be interpreted as imposing a signal-to-noise dependent adaptive trust region, composable with the rotational or spectral-normalizing aspects of matrix-whitening (Section .1).

5. Variance adaptation is a crucial matrix-whitening ingredient

When examining the design of optimizer flavors, a recurring choice occurs in approximating Equation (2) – regardless of the prior or post transformations, a raw update can be normalized by either 1) its instantaneous sign, or 2) by its square-root historical (uncentered) variance, which we refer to as **variance adaptation** following [2]. This distinction can be made explicitly clear by considering three pairs of optimizers:

$$\textbf{Signum: } \mathrm{sign}(\bar{g}) \quad \rightarrow \quad \textbf{Adam: } \bar{g} \odot \mathbb{E}[g^2]^{-1/2} \tag{6}$$

$$\textbf{SPlus: } \mathrm{unrot}(\mathrm{sign}(\mathrm{rot}(\bar{g}))) \quad \rightarrow \quad \textbf{SOAP: } \mathrm{unrot}(\mathrm{rot}(\bar{g}) \odot \mathbb{E}[\mathrm{rot}(g)^2]^{-1/2}) \tag{7}$$

$$\textbf{Muon: } \mathrm{NS}(\bar{g}) \quad \rightarrow \quad \textbf{AdaMuon: } \mathrm{NS}(\bar{g}) \odot \mathbb{E}[\mathrm{NS}(g)^2]^{-1/2} \tag{8}$$

For each pair, the same rotational behavior is used (e.g. an identity basis, a rotated eigenbasis, or implicit Newton-Shulz basis), but the elementwise normalizations are handled differently. Note that the Newton-Shulz operator of Muon is implicitly a signed descent method, as it approximates the orthogonalization of $\bar{g}$ such that all singular values are ± 1.

We find that utilizing variance adaptation consistently achieves stronger results than otherwise. This trend remains consistent across all three optimizer pairs, as shwon in Figure 2, and the performance difference is nontrivial – for example, the difference between Muon and Adamuon is almost as large as the difference between Adam and Muon itself, indicating that variance adaptation is roughly as important as the spectral-normalizing aspect of matrix whitening.

Notably, variance adaptation is a natural consequence of the original whitening metric (Equation (2)), but theoretical equivalences between matrix-whitening methods and spectral descent [3] often rely on "disabling the accumulation" and treating all methods as signed descent (in a basis of choice), which may not be capturing the full picture. In fact, comparing Adam and Muon may be *understating* the gains from Newton-Schulz orthogonalization; a more fine-grained comparison would be Signum vs. Muon, or Adam vs. AdaMuon. We believe that proposed optimizers that focus solely on orthogonalizing updates [1, 17] will benefit from re-implementing variance-adaptation in some form.

References

[1] Kwangjun Ahn, Byron Xu, Natalie Abreu, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

[2] Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *International Conference on Machine Learning*, pages 404–413. PMLR, 2018.

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

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

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] David Carlson, Volkan Cevher, and Lawrence Carin. Stochastic spectral descent for restricted boltzmann machines. In *Artificial intelligence and statistics*, pages 111–119. PMLR, 2015.

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

[8] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

[9] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

[10] Kevin Frans, Sergey Levine, and Pieter Abbeel. A stable whitening optimizer for efficient neural network training. *arXiv preprint arXiv:2506.07254*, 2025.

[11] Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

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

[13] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[14] K Jordan, Y Jin, V Boza, Y Jiacheng, F Cecista, L Newhouse, and J Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024b. *URL https://kellerjordan. github. io/posts/muon*, 2024.

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

[16] Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical fisher approximation for natural gradient descent. *Advances in neural information processing systems*, 32, 2019.

[17] Tim Tsz-Kit Lau, Qi Long, and Weijie Su. Polargrad: A class of matrix-gradient optimizers from a unifying preconditioning perspective. *arXiv preprint arXiv:2505.21799*, 2025.

[18] Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5):1454–1466, 2017.

[19] Xi-Lin Li. Preconditioner on matrix lie group for sgd. *arXiv preprint arXiv:1809.10232*, 2018.

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

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

[22] Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham Kakade, and Lucas Janson. A new perspective on shampoo's preconditioner. *arXiv preprint arXiv:2406.17748*, 2024.

[23] Antonio Orvieto and Robert Gower. In search of adam's secret sauce. *arXiv preprint arXiv:2505.21829*, 2025.

[24] Robin M Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley-benchmarking deep learning optimizers. In *International Conference on Machine Learning*, pages 9367–9376. PMLR, 2021.

[25] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[26] Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel pytorch implementation of the distributed shampoo optimizer for training neural networks at-scale. *arXiv preprint arXiv:2309.06497*, 2023.

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

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

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

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

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

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

[34] Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models. *arXiv preprint arXiv:2407.07972*, 2024.

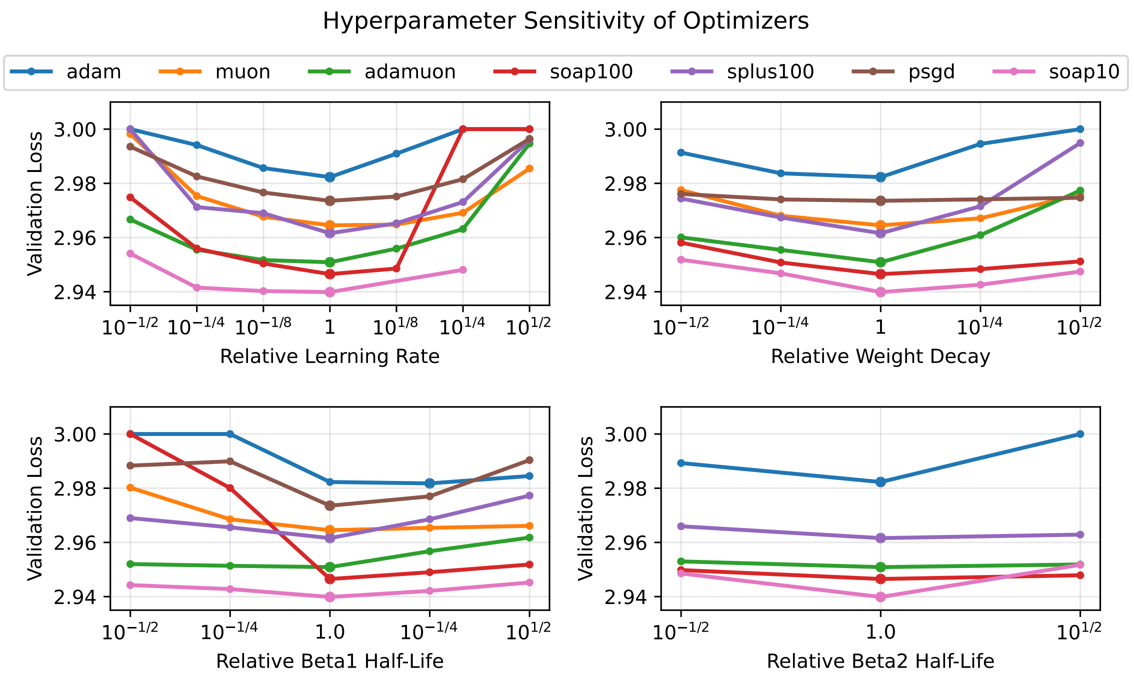

Figure 3: All methods are tuned to within a local optimum of four key hyperparameters. **Matrix-whitening optimizers generally maintain their relative performance gains across local adjustments to hyperparameters.** Plots are centered around each method's optimal hyperparameters.

.1. Why does variance adaptation still work when done after orthogonalization?

Interestingly, variance adaptation appears to provide a benefit regardless of the specific basis in which the adaptation is performed in. In SOAP, variance adaptation is performed in the *rotated eigenbasis*, as a pure alternative to signed descent. The same exchange is done in Adam versus Signum. However, in the AdaMuon setup, variance adaptation is performed in the *original elementwise basis*, after the update has already been spectrally-normalized via the Newton-Schulz iterations. In both cases, variance adaptation provides a reliable performance boost. One understanding that may explain the this phenomenon is the interpretation of variance adaptation as a heuristic for dynamically adjusting a trust region in proportion to a signal-to-noise ratio. As described in [23], when $\beta_1 = \beta_2$, Adam can be re-written as:

$$\textbf{Adam:} \quad \text{sign}(\bar{g}) \cdot \frac{1}{\sqrt{1 + \bar{\sigma}^2 / \bar{g}^2}} \tag{9}$$

where $\bar{\sigma}^2 = \beta \cdot \text{EMA}_\beta \left[(\bar{g} - g)^2 \right]$, i.e. an exponential moving average of the *centered* variance of gradients. Under this interpretation, the variance adaptation term serves as a dynamic learning-rate adjustment and does not necessarily have to share the same basis as the ‘sign‘ term.

For this reason, we argue that matrix-whitening as described in Equation (2) serves *two* interpretable purposes. The first is to spectrally normalize updates, in effect reducing the learning rates of correlated parameters to prevent over-updating. The second is to further modulate these learning

Method	Val Loss	Walltime	Memory Usage
Elementwise Basis			
Sign [Signum]	3.008 $\pm.008$	1.0	$2n^2$
Sign + Lookahead [Lion]	3.008 $\pm.008$	1.0	$2n^2$
Variance-Full ($\beta_1 = \beta_2$)	2.994 $\pm.008$	1.0	$3n^2$
Variance-Factorized [Adafactor]	**2.989** $\pm.008$	1.0	$2n^2 + 2n$
Variance-Full [Adam]	**2.982** $\pm.008$	1.0	$3n^2$
Shampoo Basis (Every 100)			
Sign [SPlus]	2.961 $\pm.003$	1.2	$4n^2$
Sign + Lookahead	**2.949** $\pm.003$	1.2	$4n^2$
Variance-Full ($\beta_1 = \beta_2$)	2.952 $\pm.003$	1.2	$5n^2$
Variance-Factorized	**2.946** $\pm.003$	1.2	$4n^2 + 4n$
Variance-Full [SOAP]	**2.946** $\pm.003$	1.2	$5n^2$
Newton-Schulz "Basis"			
Sign	2.964 $\pm.003$	1.07	$2n^2$
Sign + Lookahead [Muon]	2.961 $\pm.003$	1.07	$2n^2$
Variance-Full ($\beta_1 = \beta_2$)	2.953 $\pm.003$	1.07	$3n^2$
Variance-Factorized	**2.943** $\pm.003$	1.07	$2n^2 + 2n$
Variance-Full [AdaMuon]	2.950 $\pm.003$	1.07	$3n^2$

Table 2: **Ablations on variance-adaptation across three optimizer families.** When a specific combination resembles a previously proposed method, we include that method in brackets.

rates by a signal-to-noise term. While typically performed together, these two transformations can also be *decoupled* and done separately, as is the case in AdaMuon.

As a didactic example, we consider the "SPA" algorithm (SPlus-then-Adam), that first spectrally-normalizes updates with SPlus, and performs variance-modulation *afterwards*:

$$\textbf{SPA:}\ \text{unrot}(\text{sign}(\text{rot}(\bar{g}))) \ \odot\ \mathbb{E}[\text{unrot}(\text{sign}(\text{rot}(\bar{g}))^2]^{-1/2} \tag{10}$$

When tuned, this addition achieves a final validation loss of 2.955, improving upon SPlus (albeit to a lesser degree than SOAP). We leave further examination on how to reconcile the proper bases for spectral-normalization and variance-adaptation to future investigation.

.2. Can lookahead strategies replace variance adaptation?

The downside of variance adaptation is that one must keep track of an additional set of parameters in memory. Signed methods employing "lookahead" techniques (e.g. Lion [8] and under loose interpretations MARS [33] or Cautious optimizers [21]) are a way to approximate this behavior without the additional memory cost. The general idea is to calculate the sign of $(1-\beta_3)\bar{g} + \beta_3 g$, where β_3 is a new hyperparameter. For high-variance gradients, the intuition is that the sign will flip more often between subsequent updates, resulting in a smaller overall change. This can also be interpreted as a generalization of Nesterov momentum [9] which fixes $\beta_3 = 1 - \beta_1$. In Table 2, we sweep over β_3 for lookahead variants of the signed optimizers, and find that while gains can be achieved, these variants do not reliably reach the performance of variance-adapted variants (and require a sensitive additional hyperparameter).

.3. Can low-rank factorization reduce the memory footprint of variance adaptation?

An alternate way to reduce memory requirements is to utilize a rank-1 approximation of the variance buffer, reducing memory usage from mn to $m+n$. We find that factorized variance estimators retain almost exactly the same performance as the full matrices, and at times even improve performance. We utilize the following scaled Adafactor [25] update:

$$v_L \leftarrow (1-\beta_2)\cdot v_L + (1-\beta_2)\cdot \text{mean(G, axis=0)} \tag{11}$$
$$v_R \leftarrow (1-\beta_2)\cdot v_R + (1-\beta_2)\cdot \text{mean(G, axis=1)} \tag{12}$$
$$U = \bar{G} \ \oslash\ (v_L v_R^T)\cdot(\text{len}(v_R)/\text{sum}(v_L)) \tag{13}$$

As shown in Table 2 under the "Variance-Factored" label, using a rank-1 factorization results in negligible performance changes. For the specific case of Muon, the factorized variance estimator even *improves* performance over the full matrix estimator. We hypothesize that this may be due to a bias-variance tradeoff in the variance estimator. Taking the view from Section .1, variance adaptation can be seen as assigning a dynamic learning-rate to specific parameters, and this adaptivity may be more effective in practice if averaged over multiple parameters sharing a natural relationship, such as the input/output bases of a weight matrix [22].

Table 3: **Shared hyperparameters across model training.**

Hyperparameter	**Value**
Adam LR (Embed)	0.01
Adam LR (Output Head)	0.01
Adam LR (Layernorm)	0.01
Weight Decay (Embed)	0
Weight Decay (Output Head)	0.001
Weight Decay (Layernorm)	0
β_1 (Embed/Output/Layernorm)	0.9
β_2 (Embed/Output/Layernorm)	0.99
LR Warmup	200 steps
LR Decay	Cosine
Sequence Length	256
Batch Size	1024
Weights Precision	fp32
Optimizer Precision	fp32
Activation Precision	bf16
Hidden Size	768
MLP Ratio	4
Attention Heads	12
Num Blocks	12