IMPROVING SOAP USING ITERATIVE WHITENING AND MUON

Nikhil Vyas* Harvard University
Rosie Zhao* Harvard University
Depen Morwani* Harvard University
Mujin Kwun Harvard University

Sham Kakade Kempner Institute at Harvard University

ABSTRACT

TL;DR: Combining SOAP and Muon: We first apply SOAP on the smaller side (as in GaLore) and then apply Muon on the resulting preconditioned gradient (along with some heuristics— see Section 2.1). Code: `https://github.com/nikhilvyas/SOAP_Muon`.

The analyses presented in this report utilize simple mathematical models to explore the effects of these optimizers, serving as a basis for inspiration in designing new ones. These findings should not be interpreted as definitive claims about actual gradient distributions. The experiments are preliminary and primarily intended to provide an intuitive sense of performance; we hope the accompanying code serves as a foundation for further optimization research.

1 WHITENING

In this section, we first describe whitening and then examine the conditions under which certain optimizers— Adam (Kingma & Ba, 2015), Shampoo/SOAP (Gupta et al., 2018a; Vyas et al., 2024b), and Muon (Jordan et al., 2024)— effectively achieve gradient whitening.

Let g be the random vector that denotes the gradient. This distribution may arise from inputs and sampled labels— yielding the Gauss-Newton component of the Hessian in the case of cross-entropy loss— or from past time steps, as seen in Adagrad. Let $H = E[gg^T]$. The final direction[1] for many optimization methods roughly takes the form $H^{-p}g$ for some constant p. Specifically, we will focus on the case when $p = 1/2$. Denoting $g' = H^{-1/2}g$, then we have $E[g'g'^T] = I$. This suggests that an implicit goal of certain optimization strategies may be to whiten gradients. **While whitening may not necessarily be the correct objective, we will assume it is for our analysis.**

Diagonal version of whitening: Suppose we are promised that H is a diagonal matrix. In this case, we can simply estimate $D_i = E[g_i^2]$ and use this to obtain g'. This is equivalent to Adam (after adding momentum and replacing expectations for running averages).

1.1 KRONECKER PRODUCTS AND SHAMPOO/SOAP

The idea behind Shampoo is to exploit that fact that neural network parameters are structured as matrices. For a layer of size $m \times n$ let g be the random vector that denotes the gradient and G be the same vector restructured as a $m \times n$ matrix. Shampoo perfectly whitens the distribution if we assume that there exist matrices O_L, O_R such that for $G' = O_L G O_R$ and $g' = \text{vec}(G')$ we have that $E[g'g'^T]$ is a diagonal[2] matrix, i.e. we become diagonal after applying some rotation matrices. This is equivalent[2] to assuming that the matrix H is a Kronecker product. In this sense, SOAP directly generalizes Adam since the class for which we perfectly whiten the gradients is larger.

*Equal contribution. Correspondence to `vyasnikhil96@gmail.com`, `rosiezhao@g.harvard.edu`, `dmorwani@g.harvard.edu`.

[1]Note that here we being informal and using g for both the loss gradient and the random variable indicated previously. This distinction will not be too important for our purposes.

[2]This is an over-claim. Specifically, let $E[g'g'^T] = D$, we also need that the mn diagonal entries of D when reshaped into matrix, form a rank 1 matrix. The rank 1 condition can be relaxed for SOAP. We will ignore these subtleties for simplicity.

1.2 MUON

Muon (Jordan et al., 2024; Bernstein & Newhouse, 2024) uses an interesting operation for whitening the gradients. Specifically, given the singular value decomposition $G = USV^T$, Muon uses $G' = UV^T$ (referred to as orthonormalization from now onwards) as the final gradient direction. This approach offers computational advantages, as it is significantly faster (by using Newton-Schultz iterations) than QR or eigenvector decomposition—at least based on the implementations we are aware of—though it must be applied at every optimization step. To better understand the effect of this operation, let's analyze it in simple cases based on the distribution of the gradient discussed above. Suppose G is distributed as MD, where M is a random matrix with i.i.d. Gaussian entries and D is a fixed diagonal matrix. In this setting, whitening can be achieved by simply right-multiplying by D^{-1} to rescale back the columns. Note that this distribution falls in the class of distributions perfectly whitened by Adam[2]. Let us see how Muon performs in this setting (results in `https://github.com/nikhilvyas/SOAP_Muon/blob/main/Muon_effect.ipynb`).

- At $m = n$, Muon perfectly whitens the distribution.
- At $4m = n$ (similar shape to MLP layers of Transformers), Muon reduces the max/min column norm ratio[3] from 100 to around 3.4.
- At $50m = n$ (similar shape to first/last layers of language models), Muon reduces the max/min column norm ratio from 100 to around 14.

Let us now consider a more complicated gradient structure where G is distributed as $D_L M D_R$ where D_L and D_R are fixed diagonal matrices, i.e. both columns and rows are rescaled. Again, it can be seen that this belongs in the class of distributions perfectly whitened by Adam (and hence also SOAP).

- At $m = n$, Muon reduces the expected max/min entry ratio from 100 to around 4.5.
- At $4m = n$, Muon reduces the expected max/min entry ratio from 100 to around 7.

In this simple mathematical model of whitening (which may not encompass all potential benefits of Muon), the overall takeaways are as follows:

- Muon effectively whitens the gradient when the matrix is nearly square and conditioning is applied to one side.
- Muon is less effective at whitening the gradient when the matrix is highly rectangular or when conditioning is applied on both sides.

2 COMBINING SOAP AND MUON

Given above conclusions, we propose combining SOAP and Muon in the following way: we first apply SOAP on the smaller side (as in GaLore) and then apply Muon on the resulting preconditioned gradient. SOAP removes the conditioning from one side and then Muon conditions the other side (assuming[4] the matrix is near square). Intuitively, this has the benefit of being able to precondition both sides (at least approximately) without significantly increasing computational costs. Empirically, we observe the following benefits:

1. Since SOAP is applied on the smaller side, the space and time overhead is significantly reduced compared to full SOAP.
2. At large batch sizes, we improve over both one sided-SOAP and Muon. Curiously, we also do better than SOAP (see Section 3).

[2] and hence also SOAP, by just taking $O_L = I_m, O_R = I_n$. If O_L, O_R were some other orthonormal matrices SOAP would continue to work. We take $O_L = I_m, O_R = I_n$ w.l.o.g. for understanding the effect of Muon since it is rotationally invariant to O_L, O_R.

[3] Equivalent to sqrt of condition number.

[4] If the matrix is not square this would not work perfectly. Fortunately, most rectangular matrices have diagonal structure on the large side, for example: last layer of language models (Zhao et al., 2024c) which one-sided-SOAP can whiten since it uses the diagonal basis for the large side for such layers.

3. Since Muon can be run at every step, this empirically allows us to run SOAP with much higher preconditioning frequencies (such as 40) with nearly no loss[5] in performance.

2.1 Fixing instabilities

In some setups (such as Modded-NanoGPT) but not others (such as the OLMo codebase (Groeneveld et al. (2024), `https://github.com/rosieyzh/optimizers-llm`)) we saw that SOAP-Muon would have unstable behavior and perform poorly. We found that applying another round of whitening by the SignSGD operation (or Adam) in the basis used by one-sided SOAP fixed this (this can be interpreted as another (iterative) whitening operation—see Section 3). Later we found that the intermediate operation of taking elementwise square root (along with fixing back the overall gradient norm) performed even better. Even in cases where the square root was not needed (such as OLMo codebase) we found no harm in performance by using it, and as such we have set it to occur by default. This square root operation can be seen as an instantaneous analogue of pAdam (Chen et al., 2020) with $p = 1/2$ i.e. a more conservative version of SignSGD/Adam.

When not using the square root operation we sometimes found that fewer steps[6] of Newton-Schultz (inside Muon) did better than running till convergence. We are unsure of what causes this; our best guess would be that when orthonormalization adds too much noise when it is applied to gradient distributions which are already close to whitened.

2.2 Experimental Results

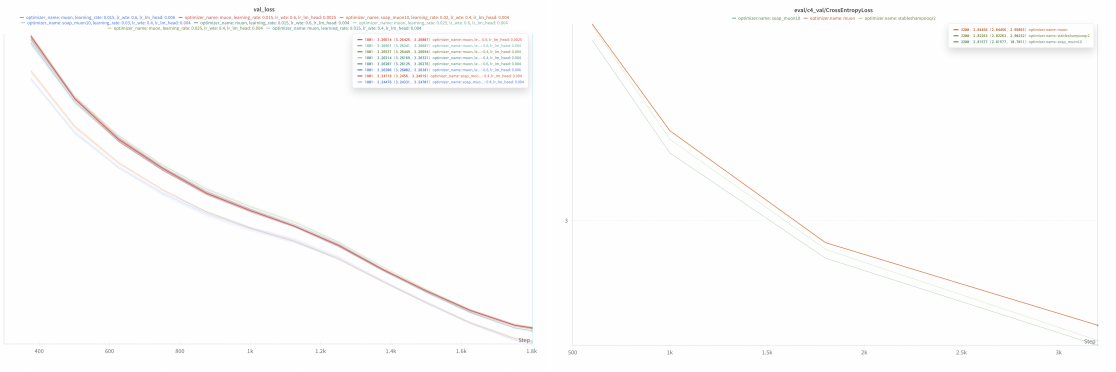

Figure 1: Experiments in Modded-NanoGPT and OLMo codebase respectively. `soap-muon10` refers to SOAP-Muon with sqrt correction, `stableshampooqr2` is equivalent to SOAP.

Modded-NanoGPT: We start with a older version of Modded-NanoGPT[7] (before the move to U-net based architecture) and increase[8] the batch size by 4x since we are interested in the large batch setting. We train for 1800 steps with 600 steps of cooldown. For SOAP-Muon, we use a frequency of 40. Additionally, since we use one-sided SOAP the overhead of the SOAP component remains relatively low. We provide a Wandb report at the following link: `https://api.wandb.ai/links/harvardml/3elqdb7c`. The plot is an average over 6 seeds.

OLMo: We provide another Wandb report using the OLMo codebase: `https://api.wandb.ai/links/harvardml/0saq48vk`, SOAP (stableshampooqr2 in the link) visualization seems to be buggy in wandb report. The plots show the minimum over all runs with different hyperparameters (79 runs for Muon and only 3 runs for SOAP-Muon, as prior experiments provided a good understanding of optimal hyperparameters). For Muon we separately tuned Adam's learning rate for the first/last layer, while for SOAP-Muon we tuned a single learning rate. We used

[5]This is in a setup such as Modded-NanoGPT where even frequency 10 did nontrivially worse than frequency 1.

[6]To define this consistently we begin by dividing the matrix by the top singular value (easily obtainable by power iteration) instead of the norm.

[7]In general we found this to be a good setup to stress test our optimizer since it is pretty well tuned w.r.t. Muon. For example, we found that changing things like activation function hurts Muon relatively more than SOAP/SOAP-Muon.

[8]The performance gap between Muon and SOAP-Muon at the default batch size of 512k was much smaller, around $\sim .005$.

frequency 10 for SOAP-Muon. Other hyperparameters/architecture details are the same as in the 360m parameter model experiments in Vyas et al. (2024b). In this setup we observe a slightly larger gap of around 0.03, likely due to the architecture and other details not being well optimized for Muon.

3 ITERATIVE WHITENING

In this section we motivate SOAP-Muon from a slightly different perspective, framing it as an iterative whitening process. First, let us look again into diagonal whitening.

Diagonal version of whitening : We can define a diagonal version of whitening in two ways:

- Let D be the optimal diagonal approximation of H and $g' = D^{-1/2}g$. Here the intuition is that we first approximate H itself by a diagonal matrix and use that in place of H.
- Let D be a diagonal matrix such that $g' = D^{-1/2}g$ satisfies $diag(E[g'g'^T]) = I$. The intuition here is that we have replaced the full constraint $E[g'g'^T] = I$ by a weaker constraint $diag(E[g'g'^T]) = I$ while also constraining the conditioning to be diagonal.

Surprisingly, for this diagonal case both approaches result in exactly the same answer. Let us investigate analogous conditions for the case of Kronecker approximations, relating this to Shampoo/SOAP.

3.1 KRONECKER PRODUCTS AND SHAMPOO

As mentioned earlier, the idea behind Shampoo is to exploit that fact that neural network parameters are structured as matrices. For a layer let g be the random vector that denotes the gradient and G be the same structured as a $m \times n$ matrix.

We define $G' = L^{-1/2}GR^{-1/2}$. In terms of Kronecker products, this is equivalent to $g' = (L^{-1/2} \otimes R^{-1/2})g = (L \times R)^{-1/2}g$. Now we can define the optimal L and R in at least two ways:

- Choose L, R such that $L \otimes R$ is the optimal Kronecker approximation of H. This question and its connection to Shampoo is studied in Morwani et al. (2024) and used in the design of SOAP. They show that the optimal L and R satisfy $L = c_1 E[GRG^T]$, $R = c_2 E[GLG^T]$ for some constants c_1, c_2.
- Choose L and R such that G' satisfies $E[G'G'^T] = I$ and $E[G'^T G'] = I$. This can be thought of as an "equilibrium" condition where applying Shampoo further would not yield any change in G'. In other words, it can be shown that the L and R which satisfy $L = c_1 E[GR^{-1}G^T]$, $R = c_2 E[GL^{-1}G^T]$ for some constants c_1, c_2 lead to such a G'. This solution was also obtained by Lin et al. (2024) (although we are unsure if the optimality conditions are the same).

Unlike the diagonal case, here we see that these two approaches lead to different solutions. We will later see some experiments suggesting that the second approach might lead to faster optimization.

Iterative Whitening: Before moving on let us consider an alternative approach which expands on the "equilibrium" notion above and approaches the second solution. Suppose we are interested in whitening G by *left* multiplying by $L_1^{-1/2}$. It can be shown (Gupta et al., 2018b; Duvvuri et al., 2024) that $L_1 = E[GG^T]$ is the right value to use. This gives us $G' = L^{-1/2}G$. Now given G' we can ask how one can whiten it by *right* multiplying by $R_1^{-1/2}$— again the right value of R is $E[G'^T G']$. We can continue doing this iteratively and it can be shown that we will converge to the solution obtained in the second approach. SOAP-Muon can be seen as the optimizer where the first preconditioning (on G) is done by one-sided SOAP and the second preconditioning (on G') is done by the Muon optimizer. We also experimented with optimizers which correspond to SOAP-SOAP and they usually outperform SOAP. This suggests that iterative whitening (as in SOAP-Muon and SOAP-SOAP) performs better than one shot approach used in SOAP.

3.2 SMALL DETAILS

Normalizing gradients' norm per layer: While the output from the Muon optimizer returns a matrix of fixed norm, in some cases we observed that it was beneficial to graft the magnitude of gradient obtained after the one-sided SOAP

operation (i.e. before applying Muon). In other cases, prior works have observed benefits by forcing updates per layer to be of fixed norm. Larger scale experiments are needed to determine which of these two works better.

Last and first layers in LLMs: Last and first layers of language models are usually very rectangular. Given the results in Section 1.2 it makes sense why applying Muon on last and first layer performs worse than Adam. Since we are applying a combination of both in SOAP-Muon we still continue to do well, although we observed very slight performance drop by applying Muon (on top of SOAP). Our best guess would be that in this case Adam/SOAP already preconditions the gradient well and adding Muon on top only adds some noise. To avoid this, for the experimental results above we used Adam for the first and last layer in Modded-NanoGPT (matching the Muon runs) and one-sided SOAP in OLMo (as done in SOAP for very rectangular layers).

4 Future Directions

We list some promising future directions:

Frequency warmup: We have observed[9] that scheduling the eigenvector update to be more frequent in the beginning of training stabilizes training runs. This was also done in the prior implementation of PSGD.

Adam variants like AdEMAMix: Various variants/modifications of Adam such as LaProp/ADOPT (Liu et al., 2020; Taniguchi et al., 2024), Cautious (Liang et al., 2025), GRAMS (Cao et al., 2024), AdEMAMix (Pagliardini et al., 2024) have been proposed. We are hopeful that these could be used in combination with SOAP/SOAP-Muon to further improve performance, at least for some settings. In particular, we have found in some experiments that SOAP+AdEMAMix outperforms SOAP on language modeling tasks. We also note that AdEMAMix can be interpreted (Morwani et al., 2025) as combining Adam and accelerated SGD.

Merging basis across layers or experts. Transformers—and more broadly networks with identity residual connections—have the property that we expect the input distribution coming from the residual stream and the gradient distribution of the output feeding to the residual stream to not change much between consecutive layers. Assuming this, we can share the eigenbasis used for nearby layers. Specifically, we would add $E[GG^T]$ for two layers and use the eigenbasis of the sum for both of the layers. This plays particularly well with SOAP-Muon since most sides of matrices on which we apply SOAP correspond to either input or output to the residual stream. Also note when doing this with SOAP we have the benefit (unlike Shampoo and other preconditioners) that we would only be sharing the rotation, not the scales used to precondition. In a similar vein, the eigenbasis (for the input and output of experts) could be shared for multiple experts in MoE models.

More efficient blocking: When used for large scale networks, Shampoo is usually used along with blocking: the matrix is subdivided and Shampoo is applied separately on each submatrix. It has been observed (by Simo Ryu at `https://x.com/cloneofsimo/status/1843213855872483773`) that as each submatrix converges to a single entry, the optimizer converges to Adam. In keeping with Kronecker product structure we would ideally like to converge to Adafactor. This can be achieved by sharing the same left preconditioner for all submatrices which have the same rows and analogously for right preconditioner. For SOAP, we can share the eigenbasis. This can be seen as another setting where the eigenbases can be merged.

Other preconditioners: Given the perspective described in Section 3, we can combine various different whitening operations. Many other optimizers which support whitening such as PSGD (Li, 2018) could be used in combination with the above approaches and may have their own benefits. In particular PSGD and Fishleg (Garcia et al., 2023)-like approaches may allow us to parameterize larger classes of preconditioners since they do not have the restriction of being easily invertible.

Faster QR operations: Pytorch QR currently only support float32. It could be sped up considerably by adding support for BF16. Alternatively we could consider approximate version of QR operations which run faster. We explored this in `https://github.com/nikhilvyas/SOAP_Muon/blob/main/iterative_qr.py` but found that the benefits would be comparable to having an exact BF16 implementation. These iterative approaches

[9] Also by Lucas Nestler, `https://x.com/_clashluke/status/1855234255116218431`.

could also be used to improve quantization of eigenvectors since current approaches to quantizing eigenvectors (Wang et al., 2024) do not take into account the inherent ordering between eigenvectors.

Adafactor: In some cases running SOAP-Muon with Adafactor instead of Adam seems to work almost as well but in other cases it does not. We are hopeful that rank-k (Zhao et al., 2024b) or in general better approximations (perhaps rank-k + exact values for top eigenspace) of Adam's second moment which still take nearly the same space as Adafactor should suffice for recovering the performance of Adam.

Only using top eigenspace for SOAP/SOAP-Muon: When either (a) the space/time requirements for one-sided SOAP are too high or (b) the matrix is too rectangular for Muon to effectively precondition the larger dimension, we could consider using only the top eigenspace of SOAP for preconditioning. This approach is intuitive, as eigenvalues typically follow a scaling law; indeed, suppose the i^{th} eigenvalue scales as $1/i$ and the dimension is 1000. Then we can reduce the condition number from 1000 to 20 by using only 50 eigenvalues (just i.e. 5% of the space requirement of the full eigenbasis). Similar ideas are also used in GaLore (Zhao et al., 2024a) and AdaMeM (Vyas et al., 2024a) though they also store momentum only in the top eigenspace to further reduce space.

pSOAP: In some experiments on Imagenet+ResNets (but not with language models) we observed that pSOAP (defined analogously to pAdam (Chen et al., 2020) along with layerwise gradient normalization) with powers less than $p \sim 2/3$ outperformed SOAP (even on train loss after few epochs). We are unsure of what differentiates these settings, but understanding this might be crucial to improving the performance of SOAP on non-language modeling tasks.

Cross-layer terms: While we are hopeful about the aforementioned improvements we believe that we are fairly close to optimal (w.r.t iteration complexity) when restricted to layerwise operations. More significant improvements may require taking into account cross-layer terms. An extension which takes into account cross terms corresponding to adjacent layers was studied in Martens & Grosse (2015) but it is unclear if this is the right approximation for networks with residual connections.

REFERENCES

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *CoRR*, abs/2409.20325, 2024. doi: 10.48550/ARXIV.2409.20325. URL https://doi.org/10.48550/arXiv.2409.20325.

Yang Cao, Xiaoyu Li, and Zhao Song. Grams: Gradient descent with adaptive momentum scaling. *arXiv preprint arXiv:2412.17107*, 2024.

Jinghui Chen, Dongruo Zhou, Yiqi Tang, Ziyan Yang, Yuan Cao, and Quanquan Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. In Christian Bessiere (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pp. 3267–3275. ijcai.org, 2020. doi: 10.24963/IJCAI.2020/452. URL https://doi.org/10.24963/ijcai.2020/452.

Sai Surya Duvvuri, Fnu Devvrit, Rohan Anil, Cho-Jui Hsieh, and Inderjit S Dhillon. Combining axes preconditioners through kronecker approximation for deep learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8j9hz8DVi8.

Jezabel R Garcia, Federica Freddi, Stathi Fotiadis, Maolin Li, Sattar Vakili, Alberto Bernacchia, and Guillaume Hennequin. Fisher-legendre (fishleg) optimization of deep neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, et al. Olmo: Accelerating the science of language models. *arXiv preprint arXiv:2402.00838*, 2024.

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

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Ma-*

chine Learning Research, pp. 1837–1845. PMLR, 2018b. URL http://proceedings.mlr.press/v80/gupta18a.html.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://web.archive.org/web/20250122060345/https://kellerjordan.github.io/posts/muon/.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1454–1466, 2018. doi: 10.1109/TNNLS.2017.2672978.

Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. Cautious optimizers: Improving training with one line of code, 2025. URL https://arxiv.org/abs/2411.16085.

Wu Lin, Felix Dangel, Runa Eschenhagen, Juhan Bae, Richard E. Turner, and Alireza Makhzani. Can we remove the square-root in adaptive gradient methods? A second-order perspective. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 29949–29973. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/lin24e.html.

Ziyin Liu, Zhikang Wang, and Masahito Ueda. Laprop: a better way to combine momentum with adaptive gradient. *CoRR*, abs/2002.04839, 2020. URL https://arxiv.org/abs/2002.04839.

James Martens and Roger B. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In Francis R. Bach and David M. Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 2408–2417. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/martens15.html.

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.

Depen Morwani, Nikhil Vyas, Hanlin Zhang, and Sham Kakade. Connections between schedule-free optimizers, ademamix, and accelerated sgd variants, 2025. URL https://arxiv.org/abs/2502.02431.

Matteo Pagliardini, Pierre Ablin, and David Grangier. The ademamix optimizer: Better, faster, older. *CoRR*, abs/2409.03137, 2024. doi: 10.48550/ARXIV.2409.03137. URL https://doi.org/10.48550/arXiv.2409.03137.

Shohei Taniguchi, Keno Harada, Gouki Minegishi, Yuta Oshima, Seong Cheol Jeong, Go Nagahara, Tomoshi Iiyama, Masahiro Suzuki, Yusuke Iwasawa, and Yutaka Matsuo. Adopt: Modified adam can converge with any β_2 with the optimal rate, 2024. URL https://arxiv.org/abs/2411.02853.

Nikhil Vyas, Depen Morwani, and Sham M. Kakade. Adamem: Memory efficient momentum for adafactor. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ICML 2024)*, 2024a. URL https://openreview.net/forum?id=fZqMVTz7K5.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: improving and stabilizing shampoo using adam. *CoRR*, abs/2409.11321, 2024b. doi: 10.48550/ARXIV.2409.11321. URL https://doi.org/10.48550/arXiv.2409.11321.

Sike Wang, Jia Li, Pan Zhou, and Hua Huang. 4-bit shampoo for memory-efficient network training. *CoRR*, abs/2405.18144, 2024. doi: 10.48550/ARXIV.2405.18144. URL https://doi.org/10.48550/arXiv.2405.18144.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. *CoRR*, abs/2403.03507, 2024a. doi: 10.48550/ARXIV.2403.03507. URL https://doi.org/10.48550/arXiv.2403.03507.

Pengxiang Zhao, Ping Li, Yingjie Gu, Yi Zheng, Stephan Ludger Kölker, Zhefeng Wang, and Xiaoming Yuan. Adapprox: Adaptive approximation in adam optimization via randomized low-rank matrices. *CoRR*, abs/2403.14958, 2024b. doi: 10.48550/ARXIV.2403.14958. URL https://doi.org/10.48550/arXiv.2403.14958.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham M. Kakade. Deconstructing what makes a good optimizer for language models. *CoRR*, abs/2407.07972, 2024c. doi: 10.48550/ARXIV.2407.07972. URL https://doi.org/10.48550/arXiv.2407.07972.