Neon: Nuclear Norm to Beat Muon

Alexey Kravatskiy
kravtskii.aiu@phystech.edu

Ivan Kozyrev
kozyrev.in@phystech.edu

Nikolay Kozlov
kozlov.na@phystech.edu

Alexander Vinogradov
vinogradov.am@phystech.edu

April 26, 2025

In this paper, we develop a new algorithm for optimization of functions of weight matrices, which are typical for training large language models. Changing spectral norm, which was used to derive Muon, to nuclear norm, we pose a new optimization problem for an update matrix, solution of which defines a novel algorithm we name Neon. To make it feasible, we use Lanzos algorithm to find a required step. After providing theoretical guarantees of Neon convergence, we compare performances of Neon, Muon, and Adam on training multilayer perceptron, convolutional neural network and NanoGPT.

1 Idea

The goal of the project is to make variations on Muon to speed it up. Recently, authors of [1] have proposed to look at different optimizers as the solution of the optimization problem for an update. This approach can be utilized to derive Muon [2], a novel algorithm for fast training of neural networks. Instead of using spectral norm in the problem, we use nuclear norm to produce a new problem.

1.1 Problem (Project description)

In this subsection, we provide a more detailed description of our idea and formulate it as a mathematical problem. The authors of [1] suggest obtaining the update step as a solution to the optimization problem:

$$\langle g, \delta w\rangle + \lambda\|\delta w\|^2 \to \min_{\delta w}, \tag{1}$$

where w is the weight vector, g is a gradient-like vector (e.g., obtained via momentum SGD), and $\|\cdot\|$ represents a certain norm. Many popular optimizers, such as Adam (with exponential moving average disabled) and vanilla SGD, can be cast within this framework [1].

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

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

where $\|\cdot\|$ denotes a certain matrix norm. By setting this norm to the RMS-to-RMS norm (a scaled version of the spectral norm), we recover the Muon optimizer [3, 1] with an update step defined by:

$$\delta W = -\frac{1}{\lambda}\sqrt{\frac{n}{m}}UV^T, \tag{3}$$

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

Motivated by the recent achievements of the Muon optimizer (e.g., [4]), we consider alternative choices of norms, specifically the kernel norm $\|\cdot\|_*$ and a custom $F*$ norm, given by $\|X\|_{F*}^2 = (\|X\|_F + \|X\|_*)/2$, where $\|\cdot\|_F$ denotes the Frobenius norm.

Using the kernel norm in (2) leads to a rank-one update of the weight matrices:

$$\delta W = \frac{1}{\lambda} u_1 \sigma_1 v_1^T \,, \tag{4}$$

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

Another choice is the $F*$ norm. With this choice, (2) yields

$$\delta W = \frac{1}{\lambda} U D V^T \tag{5}$$

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

$$\sum_{i=1}^{n} [\sigma_i - \tau]_+ = \tau \,. \tag{6}$$

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

In this article we show how one can quickly compute weight updates defined by (4) and (5). Then we finilize the methods by adding momentum and test their performance against those of Muon ant training multilayer perceptron and transformer. The results will be fast algorithms, which we will convert into a new optimizer classes for PyTorch, as was done with Muon.

2 Outcomes

The main results of the project are expected to be:

1. Deriving of Neon and argumentation of the numerical methods which will be used in the algorithm
2. Code implementation of Neon as PyTorch optimizer class
3. Numerical experiments comparing performance of training with Neon, Muon, SVD and Adam on several architectures: MLP, CNN, and NanoGPT.
4. Participation in CIFAR-10 challenge with Neon (if the alrothims turns out to be competitive)
5. Article describing all the results. If they are satisfactory, we intend to be in time to apply for NeurIPS 2025. It is a challenge, but I think we must give it a try.

3 Literature review

Our review focuses mainly on Muon and Shampoo optimizers, since our algorith extends ideas used to derive those. We highlight advantages and disadvantages of those methods, unique effects they introduce, and compare them to Neon.

In the previous section we described the theory behind weights update step in Muon optimizer, but we have not discussed how to obtain required matrices on practice. The update step is defined by (3), which requires UV^T with U and V^T from singular value decomposition of gradient-like matrix G. Naive solution would be to compute SVD of G, and construct required expression. However, developers of Muon optimizer came up with a workaround which uses Newton-Shulz iterations [2]. Newton-Shulz iterations from original article [2] require 10 matrix-matrix multiplications to achieve desired accuracy. Asimptotic complexity of such operation is identical to those of SVD and equals $O(mn \min\{m,n\})$ for $m \times n$ matrix, but matrix multiplication on modern GPUs can be performed much more efficiently.

The performance of Muon on training large language models was tested [4] against those of AdamW. The testing demonstrated excellent performance of Muon — it was ~ 2 times more efficient in terms of FLOPs required to reach certain loss value. It is even more remarkable, if cost of one iteration is taken into account: Muon requires additional $O(mn \min\{m,n\})$ FLOPs per $m \times n$ matrix, while AdamW needs only $O(mn)$.

Another interesting discovery about Muon optimizer is that it accelerates grokking[5]. At the test problem Muon achieved grokking significantly faster then AdamW in terms of passed epochs, mean grokking epoch for Muon was 102.89, while for AdamW it wes 153.09. Authors suggest that this may be due to the fact that Muon stimulates broder exploration by orthogonalizing gradient matrix, and thus avoids memorization.

Recently certain thoretical guarantees for Muon convergence have been derived [6]. In particular, in L-smooth convex case it achieves $1/T^{\frac{1}{4}}$

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

4 Quality metrics

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

5 Preliminary plan

Week April 28 - May 4

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

Week May 5 - May 11

- For Alexey: finalize the proofs. Verify them via small experiments on MLP and CNN. Write with Alexander Experiments for the article.
- For Ivan: join Nikolay to finalize algebra part of the article. Estimate FLOPS, memory and other overheads (produce O bounds)
- For Nikolay: write a draft of the poster (before May 6), and work with Ivan
- For Alexander: agressively test algorithms, prove that Neon outperforms competitors and prepare the results for the article.
- For everybody: write and edit the article
- May 11: submit an abstract to NeurIPS.
- May 12-14: the article is being polished.
- May 15: the article must be sent.

6 Prototyping phase report

1. Update rule is derived, see idea
2. Update rule methods are tested: power iteration vs Lanzos (see playground)
3. NanoGPT is tested on Muon and Adam (add fig). The results are strange, because the methods do not converge at the expected rate (Muon is slower)
4. Neon, Muon, AdamW and SGD are compared on MLP and CNN (add fig). All methods work correctly, but again there is the problem with which one is the fastest (for now, it's SGD).

References

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

[2] 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.

[3] Jeremy Bernstein. Deriving muon, 2025.

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

[5] Amund Tveit, Bjørn Remseth, and Arve Skogvold. Muon optimizer accelerates grokking, 2025.

[6] Jiaxiang Li and Mingyi Hong. A note on the convergence of muon and further, 2025.