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 **?** 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 **?**, 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 **?** 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 **?**.

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 **??** with an update step defined by:

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

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., **?**), 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

TO-DO

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. Recorded the distribution of singular values of gradients during NanoGPT training (add fig).
4. 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). Besides, Neon doe not converge (add fig).
5. Neon, Muon, AdamW and SGD are compared on MLP and CNN (see Figures 1, 2, 3, and 4). All methods work correctly, but again there is the problem with which one is the fastest (for now, it's SGD).

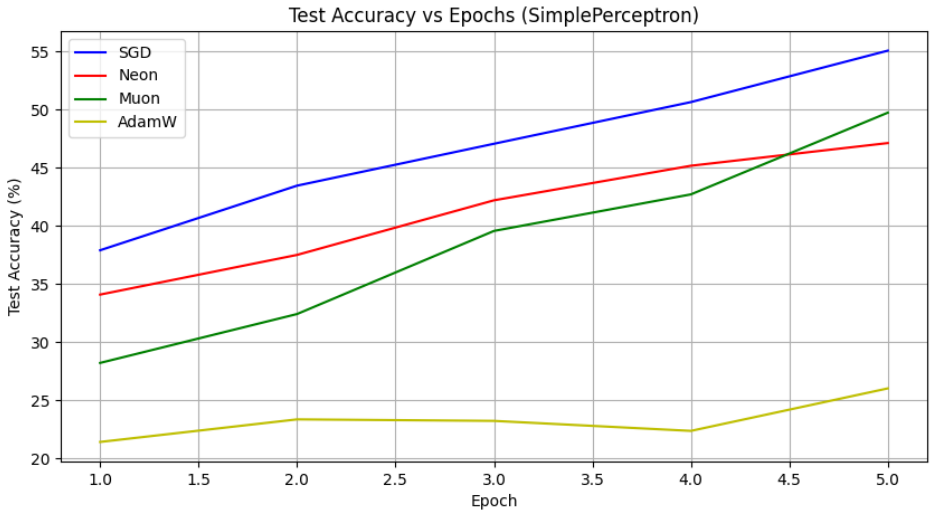

Figure 1: MLP: self.linear1 = nn.Linear(32*32*3, 512), self.linear2 = nn.Linear(512, 10), self.activ = nn.GELU()

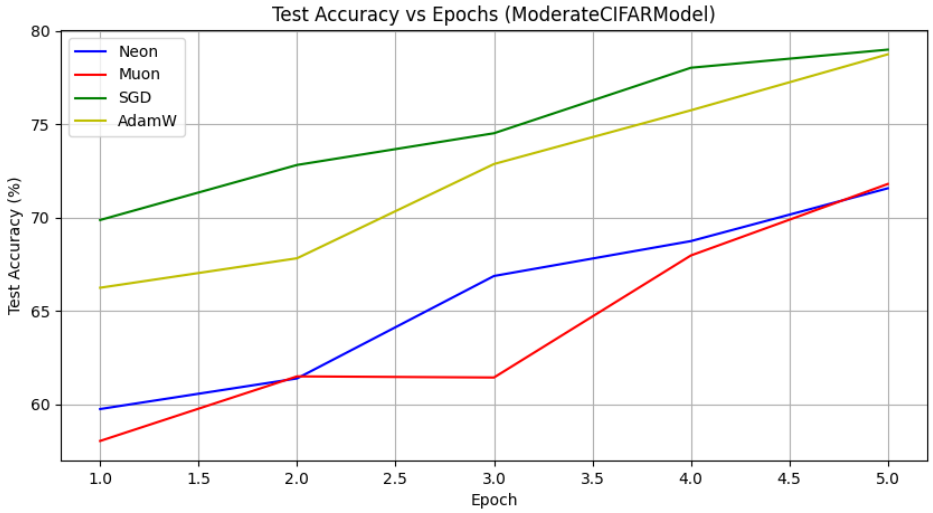

Figure 2: CNN: 2 convolutional blocks, 2 fully connected layers, activation + dropout

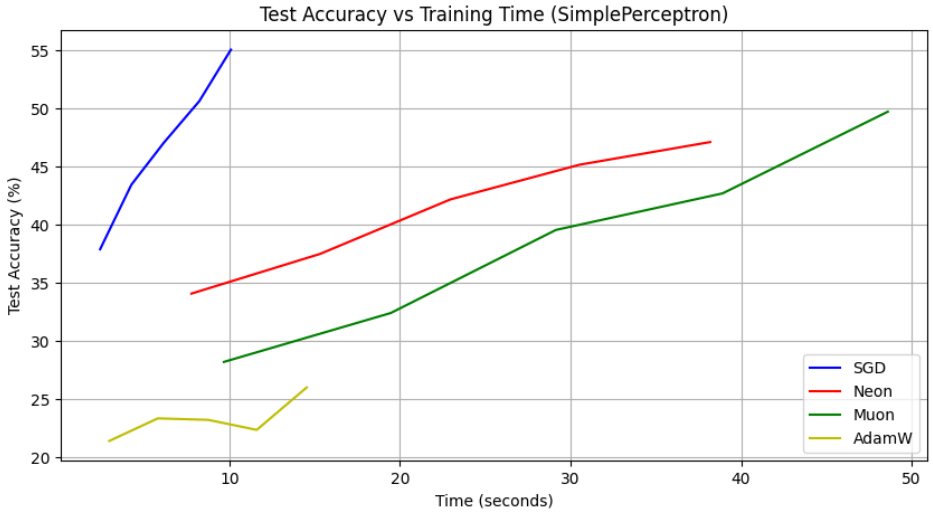

Figure 3: MLP: wallclock time measurements

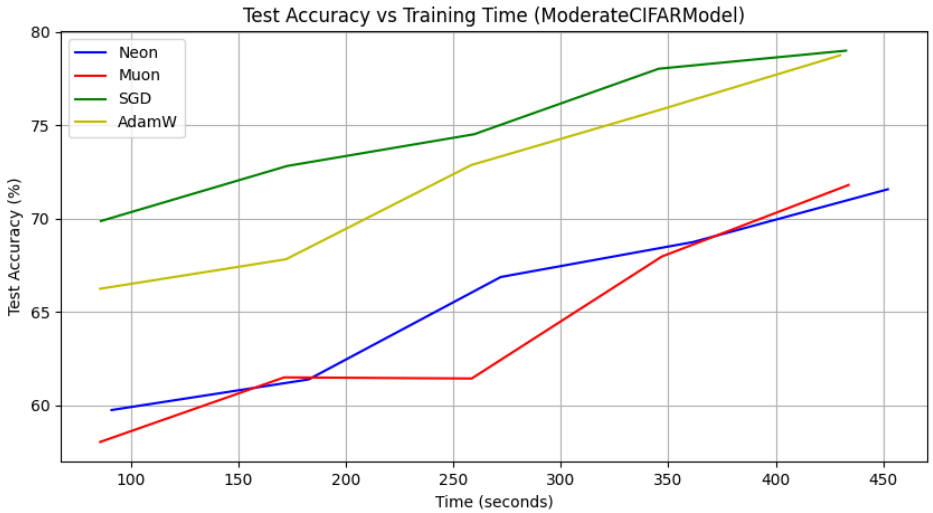

Figure 4: CNN: wallclock time measurements