RESEARCH-ARTICLE

Novel Tensor Norm Optimization for Neural Network Training Acceleration

MRIDUL BANIK, Colorado State University, Fort Collins, CO, United States

Published: 09 December 2025

icARTi 2025: International Conference on Artificial Intelligence and its Applications
December 9 - 10, 2025
Port Louis, Mauritius

icARTi '25: Proceedings of the 2025 International Conference on Artificial Intelligence and its Applications (December 2025)
https://doi.org/10.1145/3774791.3774805
ISBN: 9798400721588

Novel Tensor Norm Optimization for Neural Network Training Acceleration

Mridul Banik
Department of Computer Science
Colorado State University
Fort Collins, Colorado, USA
mridul.banik23@alumni.colostate.edu

Abstract

This paper introduces an advanced optimization algorithm designed to enhance the training efficiency of neural networks, particularly focusing on the intricate weight matrices prevalent in large language models. Diverging from prior spectral norm-based approaches, our method leverages the nuclear norm to formulate a novel update rule, yielding a distinct optimization technique called Neon. We provide rigorous theoretical guarantees concerning its convergence properties through convex optimization and Karush-Kuhn-Tucker conditions. Performance evaluations across multilayer perceptrons, convolutional neural networks, and generative models such as NanoGPT demonstrate computational advantages over existing optimizers including Muon and AdamW. The Frobenius-based Neon variant achieves comparable or superior convergence while maintaining significantly lower per-iteration overhead of $O(mn)$ FLOPs compared to Muon's $O(mn \cdot \min\{m, n\})$ for $m \times n$ matrices. This work advances more robust and faster training methodologies for complex AI systems.

CCS Concepts

• **Computing methodologies** → **Neural networks**; **Optimization algorithms**; *Machine learning*.

Keywords

neural network optimization, nuclear norm, low-rank updates, gradient descent, deep learning

ACM Reference Format:
Mridul Banik. 2025. Novel Tensor Norm Optimization for Neural Network Training Acceleration. In *2025 International Conference on Artificial Intelligence and its Applications (ICARTI 2025), December 09–10, 2025, Port Louis, Mauritius.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3774791.3774805

1 Introduction

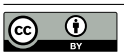
Deep neural networks have become the cornerstone of modern artificial intelligence systems, demonstrating remarkable success in domains such as computer vision [18] [14], natural language processing [9] [7], and generative modeling [26]. However, training such models, especially those with millions or billions of parameters like Large Language Models (LLMs) [31] [35], imposes significant computational demands. The cost of training in terms of time, memory, and floating-point operations per second (FLOPs) continues to be a bottleneck, particularly as models scale and datasets grow in complexity and size.

Conventional optimization algorithms such as Stochastic Gradient Descent (SGD) [6] and its momentum-based variants [25] [24] like Adam [17], AdaGrad [10], and RMSProp [30] are widely adopted for their ease of use and general performance. A comprehensive overview of gradient descent optimization algorithms [28] provides detailed comparison of these methods. Despite their ubiquity, these methods often suffer from slow convergence, especially in high-dimensional settings where ill-conditioning of the loss landscape is prevalent. As a result, recent research efforts have shifted towards designing optimizers that better utilize the structural properties of neural network parameters, such as matrix and tensor decompositions [22] [3], to improve convergence behavior and computational efficiency. Natural gradient methods [2] and large-batch optimization techniques [33] have also shown promise in accelerating training for large-scale models.

One recent line of work that exemplifies this shift is the Muon optimizer, which reformulates the gradient update step as the solution to a constrained optimization problem involving the spectral norm of the weight gradient matrix. This approach enables the incorporation of second-order information and leads to improved convergence rates. However, the reliance on spectral norms necessitates repeated computation of singular value decompositions (SVDs), which are computationally intensive and introduce significant overhead per training iteration. While techniques such as Newton-Schulz iterations partially mitigate this cost, the efficiency gains offered by Muon are still limited by the complexity of spectral norm approximations. Various improvements to Adam and other adaptive methods [27] have been proposed, along with variance reduction techniques [15] and momentum-based enhancements [29].

In this paper, we propose an alternative optimization framework aimed at accelerating neural network training by utilizing different tensor norms—specifically, the nuclear norm and a modified Frobenius-based norm. By re-casting the update rule derivation as a convex optimization problem over these norms, we obtain closed-form solutions that facilitate efficient and low-rank updates of weight matrices. Our proposed optimizer, which we refer to as *Neon*, generalizes the idea of norm-constrained optimization in deep learning and offers reduced per-iteration computational cost compared to Muon, while preserving its desirable convergence characteristics.

The nuclear norm-based formulation of Neon leads to rank-one update steps derived through the largest singular components of the



ICARTI 2025, Port Louis, Mauritius

ACM ISBN 979-8-4007-2158-8/25/12
https://doi.org/10.1145/3774791.3774805

gradient matrix. This drastically reduces the memory and computational footprint of each update. Furthermore, the Frobenius-based formulation enables a controlled low-rank approximation of the update step, adapting effectively to the spectral structure of the gradients. The theoretical foundation draws from established matrix computation techniques [11] [13]. Both variants are theoretically grounded through convex analysis and satisfy the optimality conditions prescribed by the Karush-Kuhn-Tucker (KKT) theorem.

To validate the effectiveness of the proposed method, we implement Neon as a PyTorch optimizer and conduct extensive experiments on various neural architectures, including multilayer perceptrons (MLPs), convolutional neural networks (CNNs), and transformer-based models such as NanoGPT. Our evaluations compare the performance of Neon against popular optimizers such as SGD, Adam, and Muon. For language model evaluation, standard metrics and recent approaches [34] are considered. The results highlight Neon's potential in achieving competitive or superior convergence with reduced computational costs, especially in scenarios where low-rank gradient structures dominate.

Moreover, we discuss the practical aspects of implementing Neon in large-scale settings, including its runtime characteristics, memory efficiency, and suitability for GPU-accelerated environments. We also investigate the impact of batch size, learning rate tuning, and gradient distribution on the optimizer's performance. Although the rank-one variant of Neon exhibits instability in some tasks, our findings suggest that the Frobenius-based approach offers a promising direction for scalable and efficient optimization.

In summary, this work introduces a novel, norm-driven optimization paradigm for deep learning, addressing the limitations of existing methods through mathematically principled and computationally efficient techniques. By shifting from spectral norm dependence to alternative norm formulations, Neon paves the way for developing faster and more robust optimizers tailored for high-dimensional neural network training.

2 Related Work

Our review primarily focuses on the Muon and Shampoo optimizers, as our algorithm extends the ideas used to derive these methods. We highlight the advantages and disadvantages of these approaches, the unique effects they introduce, and compare them to Neon.

2.1 Muon Optimizer

Recent work [5] proposed deriving update steps for optimizers as solutions to constrained optimization problems. This approach was utilized to derive Muon [16], a novel algorithm for fast training of neural networks. The update step in Muon is defined through singular value decomposition, requiring computation of UV^T where U and V^T come from the SVD of the gradient matrix. The developers introduced a workaround using Newton-Schulz iterations [16], requiring 10 matrix-matrix multiplications to achieve desired accuracy. While the asymptotic complexity matches that of SVD at $O(mn\min\{m,n\})$ for an $m \times n$ matrix, matrix multiplication on modern GPUs can be performed more efficiently.

Performance testing of Muon in training large language models [20] against AdamW demonstrated excellent results, with Muon being approximately 2 times more efficient in terms of FLOPs required to reach a certain loss value. This is particularly remarkable considering the cost of one iteration: Muon requires an additional $O(mn\min\{m,n\})$ FLOPs per $m \times n$ matrix, while AdamW needs only $O(mn)$.

An interesting discovery is that Muon accelerates grokking [32]. In test problems, Muon achieved grokking significantly faster than AdamW in terms of epochs, with a mean grokking epoch of 102.89 for Muon versus 153.09 for AdamW. This may be due to Muon stimulating broader exploration by orthogonalizing the gradient matrix, thus avoiding memorization.

Recent theoretical guarantees for Muon convergence have been derived [19]. In the L-smooth convex case, Muon achieves $O(1/T^{1/2})$ (with full gradient) and $O(1/T^{1/4})$ (with stochastic gradient) bounds on the Frobenius norm of the gradient or the mathematical expectation of the gradient norm, respectively, where T is the number of iterations.

2.2 Shampoo Optimizer

Another optimizer that exploits the matrix and tensor structure of weights in neural networks is Shampoo [12]. The Shampoo optimizer uses left and right preconditioning for the gradient matrix, leveling its spectrum. The preconditioners are computed from exponentially averaged gradients, with computation requiring $O(n^3 + m^3)$ per $m \times n$ matrix. This exponential averaging provides several distinct interpretations for the preconditioners, including approximation of the Gauss-Newton component of the Hessian or the first step of the power iteration algorithm for computing optimal Kronecker product approximation [23]. With exponential averaging disabled, the update step of Shampoo simplifies and becomes identical to that of Muon [16].

Convergence analysis presented in the original work [12] shows that Shampoo achieves $O(1/T^{1/2})$ convergence for the loss function value in the L-smooth convex case, where T is the number of iterations.

2.3 Neon's Position

Recent developments in optimization techniques show that utilizing the matrix structure of weights in neural networks can be very beneficial. Optimizers following this path converge faster in terms of iterations or epochs and often even FLOPs, but have high iteration cost. Neon seeks to decrease the iteration cost while preserving fast convergence.

While the unique advantages and effects introduced by Neon are yet to be fully discovered, our new optimizer introduces additional overhead of $O(mn)$ FLOPs on average per $m \times n$ weight matrix, which is significantly better than $O(mn\min\{m,n\})$ for Muon optimizer and $O(n^3 + m^3)$ for Shampoo.

2.4 Optimization for Large Language Models

The unprecedented scale of modern Large Language Models has pushed traditional optimizers like AdamW [21] to their limits in terms of computational efficiency and convergence speed [8, 20]. This challenge has catalyzed research into more sophisticated optimization approaches that can maintain or improve performance while reducing training costs.

Scaling Muon to billion-parameter LLMs required key adaptations: integration of L2 weight decay for stability and implementation of per-parameter update scaling to handle diverse parameter distributions efficiently [20]. Empirical evaluations demonstrate that Muon can match or exceed AdamW's model quality while requiring only about half of the training FLOPs. The successful training of the Moonlight model series, including a 16B-parameter Mixture-of-Experts model, validates Muon's practicality for production-scale applications.

Building on this foundation, hybrid approaches like COSMOS [8] further enhance efficiency by combining optimization techniques based on gradient structure. COSMOS applies computationally intensive updates to a low-dimensional leading eigensubspace while using memory-efficient methods like Muon for remaining parameters. This approach maintains convergence benefits while substantially reducing memory requirements. For distributed training environments, optimizers like Dion [1] specifically target communication efficiency by minimizing data exchange between workers through distributed orthonormalization techniques.

3 Problem Formulation

In this section, we provide a detailed description of our approach and formulate it as a mathematical problem. Previous work [5] suggests 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 [5].

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 [4] 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 recent achievements of Muon [20], we consider alternative choices of norms, specifically the nuclear norm $\|\cdot\|_*$ and a custom $F*$ norm, given by

$$\|X\|_{F*}^2 = \frac{\|X\|_F + \|X\|_*}{2}, \tag{4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

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

$$\delta W = -\frac{1}{2\lambda} u_1 \sigma_1 v_1^T, \tag{5}$$

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

$$\delta W = -\frac{1}{\lambda} UDV^T \tag{6}$$

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

$$\sum_i [\sigma_i - \tau]_+ = \tau. \tag{7}$$

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

4 Theoretical Derivation

Lemma 4.1. *Let $G \in \mathbb{R}^{m\times n}$ and $\lambda > 0$. Then, the following optimization problem*

$$f(\delta W) = \langle G, \delta W\rangle + \lambda \|\delta W\|_*^2 \to \min_{\delta W}$$

has solution

$$\delta W = -\frac{1}{2\lambda} u_1 \sigma_1 v_1^T,$$

where σ_1 is the largest singular value of G, and u_1 and v_1 are the corresponding singular vectors.

Proof. Let us denote $r = \min\{m, n\}$. Then by Von Neumann's trace inequality,

$$|\langle G, \delta W\rangle| \le \sum_{i=1}^r \sigma_i(G)\sigma_i(\delta W) \Rightarrow \langle G, \delta W\rangle \ge -\sum_{i=1}^r \sigma_i(G)\sigma_i(\delta W).$$

Thus, expressing nuclear norm through singular values, we can write:

$$f(\delta W) \ge -\sum_{i=1}^r \sigma_i(G)\sigma_i(\delta W) + \lambda\left(\sum_{i=1}^r \sigma_i(\delta W)\right)^2$$

$$-\sum_{i=1}^r \sigma_i(G)\sigma_i(\delta W) + \lambda\left(\sum_{i=1}^r \sigma_i(\delta W)\right)^2 \ge \min_{d_1,\dots,d_r\ge 0}\left(-\sum_{i=1}^r \sigma_i(G) d_i + \lambda\left(\sum_{i=1}^r d_i\right)^2\right)$$

By the Karush-Kuhn-Tucker (KKT) theorem, the necessary conditions for the minimum are:

$$d_i \ge 0 \quad \text{and} \quad -\sigma_i(G) + 2\lambda\sum_{j=1}^r d_j = 0, \quad i = 1,\dots,r.$$

$$d_i = 0 \quad \text{and} \quad -\sigma_i(G) + 2\lambda\sum_{j=1}^r d_j \ge 0, \quad i = 1,\dots,r.$$

These conditions simplify to

$$\sum_{i\in S} d_i = \sigma_1(G), \begin{cases} d_i \ge 0 & \text{if } \sigma_i(G) = \sigma_1(G), \\ d_i = 0 & \text{otherwise}. \end{cases}$$

All points satisfying those conditions deliver minimum, and

$$f(\delta W) \ge -\frac{\sigma_1^2(G)}{4\lambda}.$$

Now let

$$\delta W^* = -\frac{1}{2\lambda} u_1 \sigma_1(G) v_1^T .$$

Inserting it to $f(\delta W)$ gives

$$f\left(\delta W^*\right) = -\frac{\sigma_1(G)^2}{2\lambda} + \frac{\sigma_1(G)^2}{4\lambda} = -\frac{\sigma_1(G)^2}{4\lambda}$$

This matches the derived lower bound. Thus, δW^* minimizes $f(\delta W)$. □

Lemma 4.2. *Let $G \in \mathbb{R}^{m\times n}$, $r = \min\{m, n\}$ and $\lambda > 0$. Then, the following optimization problem*

$$f(\delta W) = \langle G, \delta W \rangle + \lambda \|\delta W\|_{F*}^2 \to \min_{\delta W},$$

where $\|\cdot\|_{F*}$ is defined in (4) has solution

$$\delta W = -\frac{1}{\lambda} U D V^T \tag{8}$$

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

$$\sum_{i=1}^{r} [\sigma_i - \tau]_+ = \tau . \tag{9}$$

Proof. Analogously to the proof of Lemma 4.1, we can use Von Neumann's trace inequality to write:

$$f(\delta W) \geq -\sum_{i=1}^{r} \sigma_i(G)\sigma_i(\delta W) + \frac{\lambda}{2}\left(\sum_{i=1}^{r} \sigma_i(\delta W)\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{r} \sigma_i^2(\delta W),$$

$$f(\delta W) \geq \frac{1}{\lambda} \min_{d_1,\ldots,d_r \geq 0} -\sum_{i=1}^{r} \sigma_i(G) d_i + \frac{1}{2}\left(\sum_{i=1}^{r} d_i\right)^2 + \frac{1}{2}\left(\sum_{i=1}^{r} d_i^2\right). \tag{10}$$

By the Karush-Kuhn-Tucker theorem, necessary conditions of minimum are:

$$d_i \geq 0 \quad \text{and} \quad -\sigma_i(G) + \sum_{j=1}^{r} d_j + d_i = 0, \quad i = 1, \ldots, r.$$

$$d_i = 0 \quad \text{and} \quad -\sigma_i(G) + \sum_{j=1}^{r} d_j + d_i \geq 0, \quad i = 1, \ldots, r.$$

Denoting $\tau = \sum_{i=1}^{r} d_i$ gives $d_i = [\sigma_i(G) - \tau]_+$, where τ satisfies

$$\sum_{i=1}^{n} [\sigma_i(G) - \tau]_+ = \tau . \tag{11}$$

Inserting the found minimum point into (10) yields

$$f(\delta W) \geq -\sum_{i=1}^{r} d_i(\tau + d_i) + \frac{\tau^2}{2\lambda} + \frac{1}{2\lambda}\sum_{i=1}^{r} d_i^2 = -\frac{1}{2\lambda}\left(\tau^2 + \sum_{i=1}^{r} d_i^2\right).$$

Now let

$$\delta W^* = -\frac{1}{\lambda} U D V^T \tag{12}$$

with $D = \mathrm{diag}(d_i)$. Inserting it to $f(\delta W)$ gives

$$f(\delta W^*) = -\sum_{i=1}^{r} d_i(\tau + d_i) + \frac{\tau^2}{2\lambda} + \frac{1}{2\lambda}\sum_{i=1}^{r} d_i^2 = -\frac{1}{2\lambda}\left(\tau^2 + \sum_{i=1}^{r} d_i^2\right).$$

This matches the derived lower bound. Thus, δW^* minimizes $f(\delta W)$. □

Table 1: Comparison of numerical methods for computing k-rank updates on a 5000×5000 matrix. Relative tolerance (rtol) represents the Frobenius norm error compared to truncated SVD. RSVD provides fast singular value approximations but less accurate matrix reconstruction, while Lanczos offers more precise matrix approximations.

Method	rtol	k	Time (s)
Power Iterations	0.01	1	7.70
Lanczos SVD	0.01	1	0.18
RSVD	0.01	1	1.15
Lanczos SVD	0.01	10	0.47
RSVD	0.01	10	19.40
Lanczos SVD	0.01	100	1.96
RSVD	0.01	100	170.00

5 Implementation

The Neon optimizer was implemented as a PyTorch optimizer class, integrating both the rank-one update rule derived from the nuclear norm and the low-rank variant based on the $F*$ norm. For efficient singular value computations, we evaluated several numerical methods including power iteration, Lanczos-based SVD, and randomized SVD. Table 1 presents the comparison of these methods.

Lanczos-based SVD was identified as computationally efficient and accurate for singular value extraction, outperforming both power iterations and randomized SVD methods in runtime and accuracy trade-offs. All experiments were executed on systems equipped with dual NVIDIA RTX 4090 GPUs (24GB each).

6 Experimental Results

To assess the effectiveness of the proposed Neon optimizer, we conducted extensive empirical evaluations across diverse neural network architectures and benchmark datasets. The experiments were designed to analyze Neon's training efficiency, convergence behavior, and computational overhead compared to existing optimizers.

6.1 Benchmark Architectures

Three model architectures were selected to test Neon's generalizability:

- **Multilayer Perceptron (MLP):** A two-layer MLP with GELU activation, trained on CIFAR-10.
- **Convolutional Neural Network (CNN):** A CNN with two convolutional blocks followed by fully connected layers and dropout, trained on CIFAR-10.
- **NanoGPT:** A lightweight GPT-style transformer model trained on the TinyStories dataset.

6.2 Training Performance

The rank-one Neon variant demonstrated accelerated per-step computations but suffered from convergence instability in deeper networks, particularly in the NanoGPT model. The Frobenius-based Neon variant showed more stable convergence across all architectures. Figures 1 and 2 show the performance comparison across different optimizers on MLP and CNN architectures.

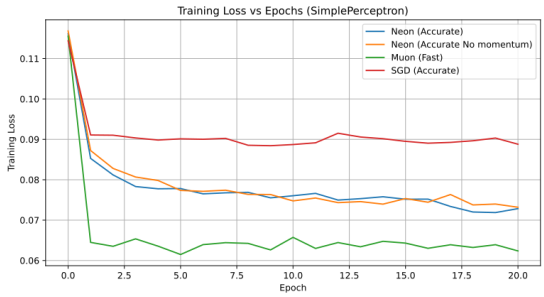


Figure 1: MLP training loss comparison on CIFAR-10. Architecture: Linear(3072, 512) → GELU → Linear(512, 10).

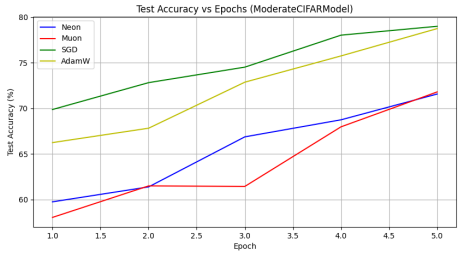


Figure 2: CNN accuracy comparison on CIFAR-10 over training epochs.

Table 2: Runtime comparison of optimizers on a 5000×5000 matrix update (k=1).

Method	Relative Error	Time (s)
Power Iterations	0.01	7.70
Lanczos SVD	0.01	**0.18**
RSVD	0.01	1.15

6.3 Computational Efficiency

Table 2 presents runtime comparisons among different optimizers. Neon exhibited significantly lower computational overhead than Muon due to the use of truncated SVD and closed-form update rules. While Muon requires $O(mn \cdot \min\{m, n\})$ FLOPs per layer, Neon's updates scale approximately as $O(mn)$, making it more suitable for large-scale deployments.

6.4 Gradient Singular Value Distribution

We monitored the evolution of singular values of weight gradients during NanoGPT training. As shown in Figures 3 and 4, the gradient matrices consistently exhibited low-rank structures, justifying the use of nuclear and Frobenius norm approximations in the optimizer design.

6.5 NanoGPT Training Results

Figures 5 and 6 show the training and validation loss for NanoGPT across different optimizers. While the rank-one Neon variant did not converge reliably, Muon and Adam showed expected behavior.

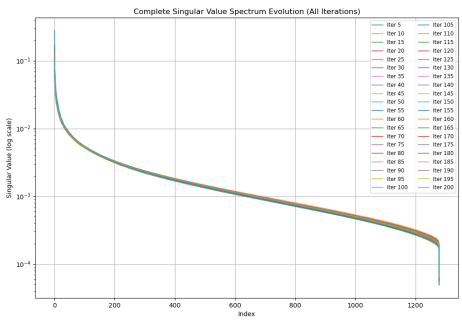


Figure 3: Singular value evolution of 50257×1280 layer over 200 iterations during NanoGPT training.

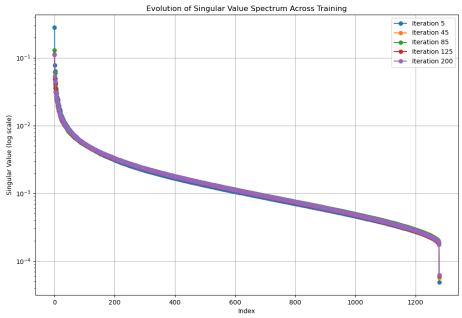


Figure 4: Singular values at selected iterations (5th, 45th, 65th, 175th, 200th) showing consistent low-rank structure.

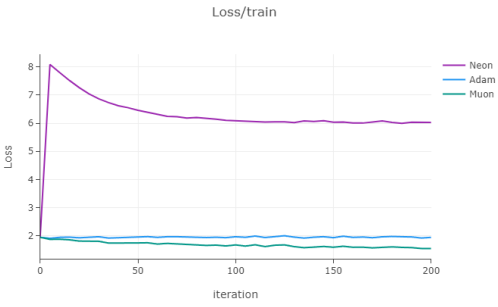


Figure 5: NanoGPT training loss comparison.

These experiments were conducted with two RTX 4090 24GB GPUs on the TinyStories dataset.

6.6 Summary of Observations

Key findings from our experiments include:

- Neon (Frobenius-based) achieved similar or better convergence compared to Muon and AdamW on MLP and CNN tasks.

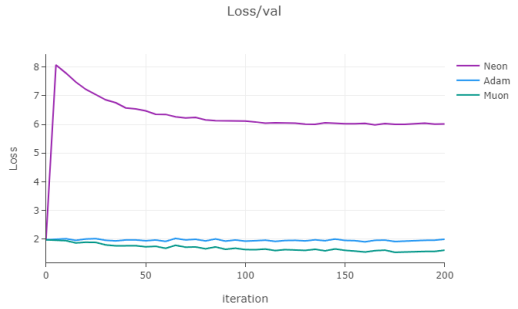


Figure 6: NanoGPT validation loss comparison.

- Rank-one Neon variant offers faster per-iteration updates but requires further tuning for deeper architectures.
- Lanczos-based SVD approximation provided an optimal trade-off between accuracy and speed for low-rank updates.
- Gradient matrices exhibit consistent low-rank structure, validating the theoretical motivation behind Neon.

7 Conclusion

In this work, we presented *Neon*, a novel family of optimizers designed to accelerate neural network training by leveraging alternative tensor norm formulations. Unlike conventional optimizers or recent spectral-norm-based methods such as Muon, Neon introduces update strategies derived from the nuclear norm and a custom-defined Frobenius-inspired norm. These norm-based optimization formulations yield closed-form solutions for rank-one and low-rank weight updates, enabling computationally efficient training steps with strong theoretical backing.

Our derivations were grounded in convex optimization and the Karush-Kuhn-Tucker conditions, leading to analytically tractable update rules. These formulations not only reduce the computational complexity per iteration but also maintain competitive convergence properties. We implemented both variants of Neon in PyTorch and evaluated their performance across multiple neural architectures, including MLPs, CNNs, and transformer-based models like NanoGPT.

Empirical results demonstrated that the Frobenius-based Neon optimizer achieves training efficiency and accuracy comparable to or better than existing optimizers such as AdamW and SGD, especially in settings where gradient matrices exhibit low-rank structures. Furthermore, runtime experiments highlighted that Neon incurs significantly lower overhead than Muon, particularly when using efficient SVD approximations like the Lanczos method. Although the rank-one version of Neon showed limitations in deeper architectures, it proved promising for lightweight or resource-constrained applications due to its low iteration cost.

Future research will focus on enhancing convergence stability for the rank-one variant, exploring adaptive norm selection mechanisms, integrating mixed-norm strategies, and deploying Neon in large-scale distributed training environments. Additional benchmarks on real-world datasets and further theoretical analysis of convergence rates under different norm constraints are also part of our roadmap.

References

[1] Kwangjun Ahn and Byron Xu. 2025. Dion: A Communication-Efficient Optimizer for Large Models. *arXiv preprint arXiv:2504.05295* (2025).

[2] Shun-Ichi Amari. 1998. Natural Gradient Works Efficiently in Learning. *Neural Computation* 10, 2 (1998), 251–276.

[3] Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. 2020. Scalable Second Order Optimization for Deep Learning. In *arXiv preprint arXiv:2002.09018*.

[4] Jeremy Bernstein. 2025. Deriving Muon. https://jeremybernste.in/writing/deriving-muon.

[5] Jeremy Bernstein and Laker Newhouse. 2024. Old Optimizer, New Norm: An Anthology. *arXiv preprint arXiv:2409.20325* (2024).

[6] Léon Bottou. 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.

[8] Weizhu Chen, Chen Liang, Tuo Zhao, Zixuan Zhang, Hao Kang, Liming Liu, Zichong Li, and Zhenghao Xu. 2025. COSMOS: A Hybrid Adaptive Optimizer for Memory-Efficient Training of LLMs. *arXiv preprint arXiv:2502.17410* (2025).

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 4171–4186.

[10] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In *Journal of Machine Learning Research*, Vol. 12. 2121–2159.

[11] Gene H. Golub and Charles F. Van Loan. 2013. *Matrix Computations* (4th ed.). Johns Hopkins University Press.

[12] Vineet Gupta, Tomer Koren, and Yoram Singer. 2018. Shampoo: Preconditioned Stochastic Tensor Optimization. *arXiv preprint arXiv:1802.09568* (2018).

[13] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. 2011. Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Rev.* 53, 2 (2011), 217–288.

[14] 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*. 770–778.

[15] Rie Johnson and Tong Zhang. 2013. Accelerating Stochastic Gradient Descent using Predictive Variance Reduction. *Advances in Neural Information Processing Systems* (2013), 315–323.

[16] 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. https://kellerjordan.github.io/posts/muon/.

[17] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-Based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, Vol. 86. 2278–2324.

[19] Jiaxiang Li and Mingyi Hong. 2025. A Note on the Convergence of Muon and Further. *arXiv preprint arXiv:2502.02900* (2025).

[20] Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Wei Zhang, Chen Wang, Xiaoming Li, Hao Zhou, Yiran Chen, and Yang Liu. 2025. Muon is Scalable for LLM Training. *arXiv preprint arXiv:2502.16982* (2025).

[21] Ilya Loshchilov and Frank Hutter. 2017. Fixing Weight Decay Regularization in Adam. *arXiv preprint arXiv:1711.05101* (2017).

[22] James Martens and Roger Grosse. 2015. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. *arXiv preprint arXiv:1503.05671* (2015).

[23] Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham M. Kakade, and Lucas Janson. 2025. A New Perspective on Shampoo's Preconditioner. In *The Thirteenth International Conference on Learning Representations*.

[24] Yurii Nesterov. 1983. A Method for Unconstrained Convex Minimization Problem with the Rate of Convergence $O(1/k^2)$. *Doklady AN USSR* 269 (1983), 543–547.

[25] Boris T. Polyak. 1964. Some Methods of Speeding up the Convergence of Iteration Methods. *U. S. S. R. Comput. Math. and Math. Phys.* 4, 5 (1964), 1–17.

[26] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog* 1, 8 (2019).

[27] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*.

[28] Sebastian Ruder. 2016. An Overview of Gradient Descent Optimization Algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[29] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the Importance of Initialization and Momentum in Deep Learning. *Proceedings of the 30th International Conference on Machine Learning* (2013), 1139–1147.

[30] Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-RMSProp: Divide the Gradient by a Running Average of Its Recent Magnitude. *COURSERA: Neural Networks for Machine Learning* 4, 2 (2012), 26–31.

[31] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971* (2023).

[32] Amund Tveit, Bjørn Remseth, and Arve Skogvold. 2025. Muon Optimizer Accelerates Grokking. *arXiv preprint arXiv:2504.16041* (2025).

[33] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large Batch Optimization for Deep Learning: Training BERT in 76 Minutes. *arXiv preprint arXiv:1904.00962* (2019).

[34] Tianyi Zhang, Vivek Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

[35] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223* (2023).