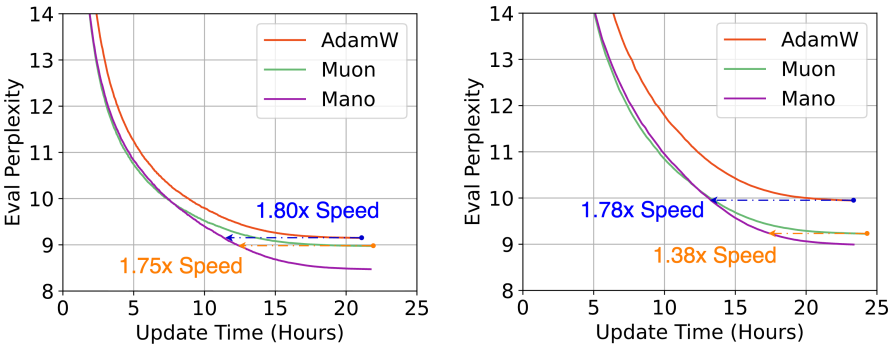
香港科技大学(广州)
THE HONG KONG
UNIVERSITY OF SCIENCE AND
TECHNOLOGY (GUANGZHOU)

xLeaf Lab
Machine Learning Foundations

Mano: Restriking Manifold Optimization for LLM Training

Yufei Gu, Zeke Xie

February 6, 2026



(a) LLaMA-350M / `Pile` (b) LLaMA-1.3B / `Pile`

Figure: One day pretraining for which Mano achieves 1.75× and 1.38× the convergence speed of Muon in wall-clock time on 4×NVIDIA H800-80G GPUs.

Table of Contents

Modern Optimizer Paradigm: Adaptive or Spectral?

Adam-based Optimizers:

- Utilize diagonal estimates of per-parameter curvature.
- Most popular in DL and widly adopted in LLM infra.
- Ignore spectral information and matrix structures, e.g., diagonal / off-diagonal elements are equally treated.

Spectral Optimizers, eg., Muon:

- Performs spectral normalization to explore all gradient directions with the same magnitude.
- Introduces Newton-Schulz iteration to approximate matrix orthogonalization (the `msign` function).
- Eliminates the curvature information encoded.
- Consumes computational overhead (though scalable).

Have we exhausted all optimization paradigms?

Traditional Manifold Optimization

A Riemannian manifold $\mathcal{M}$ is a smooth geometric space equipped with a metric that defines a smoothly varying inner product on the tangent space of $\mathcal{M}$. Manifold optimization concerns the problem of minimizing a real-valued function f over such a manifold $\mathcal{M}$, i.e.,

$$\min_{x \in \mathcal{M}} f(x) \tag{1}$$

where $f : \mathcal{M} \to \mathbb{R}$. Vanilla SGD that optimizes a loss function defined over the Euclidean vector space $\mathbb{R}^n$ can be interpreted as operating in a Riemannian manifold $(\mathbb{R}^n, g_{ij})$ with metric $g_{ij} = \delta_{ij}$.

Riemannian-SGD performs gradient updates on Riemannian manifolds through the following operations [1]:

$$\begin{cases} g_t = \nabla f(\theta_t) \\ v_t = \textbf{proj}_{\mathcal{T}_{\theta_t}\mathcal{M}}(g_t) \\ \theta_{t+1} = \textbf{proj}_{\mathcal{M}}(\theta_t - \eta_t v_t) \approx \exp_{\theta_t}(-\eta_t v_t) \end{cases} \tag{2}$$

However, traditional manifold optimization fails to generalize to modern neural networks on general tasks, particularly LLMs:

- CV/NLP objective functions are typically optimized for Euclidean space and may fail to generalize to non-Euclidean manifolds.
- Weight constraints may restrict the expressivity of LLMs.

Revisit Modern Optimizers: The Manfiold Hypothesis on Learning Trajectory

Revisiting modern optimizers like AdamW and Muon, we hypothesize that they find a optimal learning trajectory, where constituent update steps can be mapped onto some "smooth surfaces" with geometric structures that facilitate convergence and escape from local minima.

- We thus compute the average geodesic distance across 1000 constituent update steps of a Qwen3-0.6B model trained with AdamW on some popular manifolds.
- Tab. 1 suggest that the Oblique manifold best captures the geometric properties of the learning trajectory of AdamW.
- *This experiment only serves as an intuitive justification, where further studies are required to validate the hypothesis.*

Table: Average geodesic distance measured on the Oblique, Sphere, and Stiefel manifold of 1000 consecutive update steps of Qwen3-0.6B trained with the AdamW optimizer. The distance metrics are reported separately between the attention projections (Q, K, V, O) and MLP layers.

Geodesic distance	Oblique	Sphere	Stiefel
Attention	36.50	41.12	58.52
MLP Layer	21.13	37.82	53.48

The Oblique manifold $\mathcal{OB}(n, m)$ is defined as the set of $\mathbb{F}-$valued matrices with unit norm column.

- Column-wise normalization offers a highly efficient projection for the Oblique manifold, outperforming some other manifold constraints.
- Following our hypothesis, manifold-normalized update steps constrain the learning trajectory to some smoother surfaces.

Mano: Reformed Manifold Normalization

We first define the following operation:

- Tangent space projector: $\textbf{proj}_{\mathcal{T}_P\mathcal{M}}(Q)$, which project matrix Q on the first-order approximation of the manifold surface $\mathcal{M}$ around P.
- Manifold normalization operation: $\mathcal{N}_{\mathcal{M}}(A)$, which constrain matrix A on the target manifold surface $\mathcal{M}$.

For weight $\theta_t \in \mathbb{R}^{m\times n}$, gradient g_t, and learning rate η_t at timestep t, we arrived at the **Manifold Normalized Optimization**:

$$\begin{cases} g_t = \nabla f(\theta_t) \\ \hat{\theta}_t = \mathcal{N}_{\mathcal{M}}(\theta_t) \\ v_t = \textbf{proj}_{\mathcal{T}_{\hat{\theta}_t}\mathcal{M}}(g_t) \\ \hat{v}_t = \mathcal{N}_{\mathcal{M}}(v_t) \\ \theta_{t+1} = \theta_t - \eta_t\, \hat{v}_t \end{cases} \tag{3}$$

This reformed update rule is different from the original definition of manifold optimization:

- Eq. 1 defined the function f on the Riemannian manifold.
- Eq. 3 projects the gradient onto the parameter manifold and ensures that each update step remains on the manifold surface.
- By imposing a soft manifold constraint, we keep the objective function and solution unchanged in the Euclidean space.

However this update rule is still not complete...

Mano: Rotational Manifold Scheme

We hypothesize that for the Oblique manifold, only applying column-wise normalization may be insufficient:

- Spectral optimizers like Muon demonstrate that all spectral directions hold comparable importance to model convergence.
- For LLMs, the importance of column and row directions of parameter matrices should be equal.

To address this potential insufficiency of the standard Oblique manifold, we **rotate** it on a timely basis.

- We employ an alternating normalization scheme, applying column-wise scaling during odd iterations and row-wise scaling during even iterations.
- These periodic division operations induce a custom Oblique manifold characterized by an oscillating orientation.
- By consistently applying this rotation to both the tangent space of the parameters and the update step, we effectively perform manifold normalization to every parameter dimension.

We define the following operators to formulate the Mano optimizer:

- Element-wise product ($\odot$): $P \odot Q \triangleq \left(P_{ij} Q_{ij}\right)_{i,j}$.
- Element-wise division ($\oslash$): $P \oslash Q \triangleq \left(P_{ij} / Q_{ij}\right)_{i,j}$.
- Dimension-wise inner product ($\langle \cdot, \cdot \rangle_k$): For $j \in \{0, \ldots, n_k - 1\}$ and the k-th dimension, the j-th component $\langle Q, P \rangle_k^{(j)} = \langle Q^{(j)}, P^{(j)} \rangle$.
- Dimension-wise norm ($\| \cdot \|_{2,k}$): For the k-th dimension, $\|P\|_{2,k}^{(i)} = \|P^{(i)}\|_2$.

Thus, the column- and row-wise norm is denoted by $\|P\|_{2,0}^{(i)} = \|P_{i,:}\|_2$ and $\|P\|_{2,1}^{(j)} = \|P_{:,j}\|_2$ respectively.

Mano: Manifold Normalized Optimizer

The resulting **Mano** optimizer is defined as follows:

Algorithm The Mano Optimizer

Require: Layer Weight $\theta_t \in \mathbb{R}^{m\times n}$, momentum $M_t \in \mathbb{R}^{m\times n}$, learning rate η_t at step t, momentum coefficient μ, and weight decay λ.
Initialize $M_0 \leftarrow \mathbf{0} \in \mathbb{R}^{m\times n}$, $t \leftarrow 0$.
for each step **do**
$g_t \leftarrow \nabla f(\theta_t)$
$M_t \leftarrow \mu M_{t-1} + g_t$
$k \leftarrow t \bmod 2$ {Rotating Manifold}
$\hat{\theta}_t \leftarrow \theta_t \oslash \|\theta_t\|_{2,k}$ {Manifold Normalization}
$v_t \leftarrow M_t - \hat{\theta}_t \odot \langle M_t, \hat{\theta}_t \rangle_k$ {Tangent Momentum}
$\hat{v}_t \leftarrow v_t \oslash \|v_t\|_{2,k}$ {Manifold Normalization}
$\theta_{t+1} \leftarrow \theta_t - \eta_t(0.2\sqrt{n_k}\,\hat{v}_t + \lambda\theta_t)$
end for

The Mano optimizer can be summarized as **Manifold optimization with Euclidean descent**, with the reformed strategies:

- The parameters θ_t are not retracted on the manifold; the update process follows weight decay and Euclidean descent.
- Rotating Oblique manifold instead of a static geometric structure, alternating through each parameter dimension at every time step.
- We first compute the tangent momentum via parameter-space manifold projection, then apply a momentum-space manifold constraint to ensure the update step remains on the Oblique surface.

Mano (Simplified): Proof of Convergence

We provide a proof of convergence for the following simplified setting of Mano, which excludes the momentum, and fixes the Oblique manifold at the 0-th dimension (with dimension size m).

$$\begin{cases} g_t \leftarrow \nabla f(\theta_t) \\ \hat{\theta}_t \leftarrow \theta_t \oslash \|\theta_t\|_{2,0} \\ v_t \leftarrow g_t - \hat{\theta}_t \odot \langle g_t, \hat{\theta}_t \rangle_0 \\ \hat{v}_t \leftarrow v_t \oslash \|v_t\|_{2,0} \\ \theta_{t+1} \leftarrow \theta_t - \eta\sqrt{m}\hat{v}_t \end{cases} \tag{4}$$

Theorem (Convergence of Mano w/o Momentum/Rotation)

Assume that $f(\theta)$ is an L-smooth function, f is lower bounded as $f(\theta) \geq f_{\text{inf}}$, $\mathbb{E}[\xi] = 0$ for gradient noise ξ of sub-sampling, $\sin(\phi_t^{(j)}) \geq \gamma > 0$ for angle $\phi_t^{(j)}$ between $g_t^{(j)}$ and the parameter $\theta_t^{(j)}$ and the tangential component γ. Let Mano run for $T+1$ iterations. If $\eta \leq \frac{C}{\sqrt{T+1}}$ and m equals column dimension size, we have

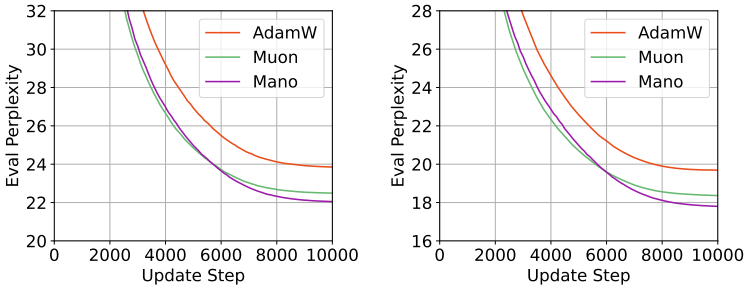
$$\min_{t=0,\ldots,T} \mathbb{E}[\|\nabla f(\theta_t)\|^2] \leq \frac{1}{\sqrt{T+1}}(C_1 + C_2), \tag{5}$$

where $C_1 = \frac{f(\theta_0) - f_{\text{inf}}}{m^{\frac{1}{2}}\gamma C}$, $C_2 = \frac{L m^{\frac{3}{2}} C}{2\gamma}$.

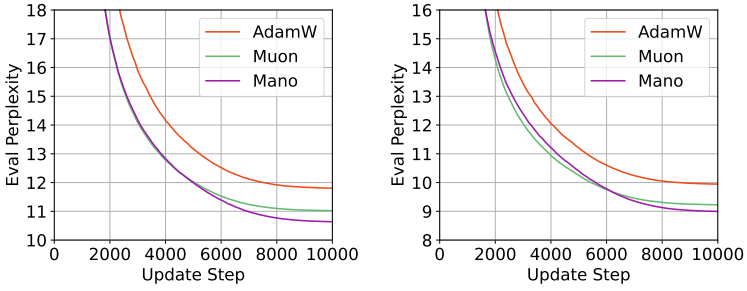
The complete proof is presented in out paper. We are expecting to extend this proof to Mano with momentum.

Mano's Empirical Results

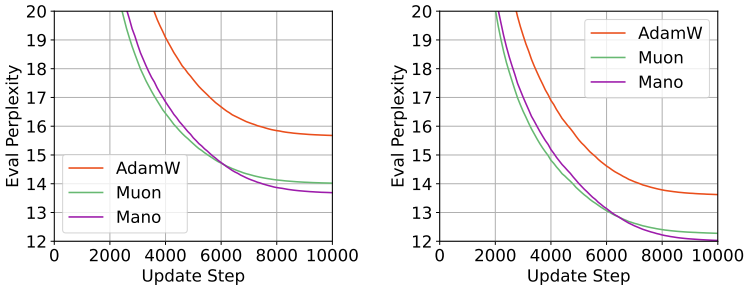
Mano exhibits faster convergence speed than baseline optimizers AdamW and Muon, with the simplest implementation and computational cost.



(a) LLaMA-350M / `C4` (b) LLaMA-1.3B / `C4`



(c) LLaMA-350M / `Pile` (d) LLaMA-1.3B / `Pile`



(e) Qwen3-0.6B / `Pile` (f) Qwen3-1.7B / `Pile`

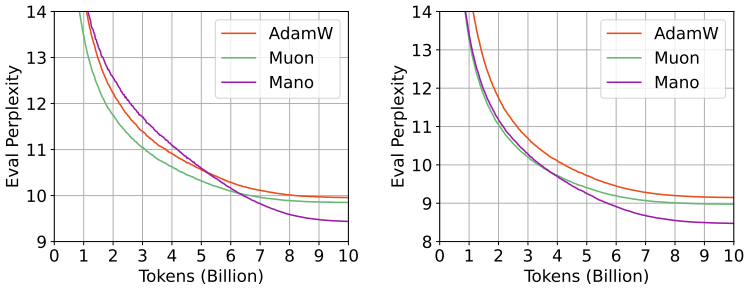
Figure: LLaMA-350M/1.3B and Qwen3-0.6B/1.7B models trained on the `C4/en` and `Pile` dataset for 10000 steps with different optimizers.

Mano's Empirical Results

Table: Numerical result of the final test perplexity of LLMs trained by different optimizers on the two pretraining corpora `C4` and `Pile` for 10000 update steps and consistent hyperparameters.

Datasets	`C4/en`		`Pile`			
Models	Llama-350M	Llama-1.3B	Llama-350M	Llama-1.3B	Qwen3-0.6B	Qwen3-1.7B
AdamW	23.852	19.690	11.803	9.945	15.679	13.624
Muon	22.491	18.365	11.022	9.227	14.020	12.276
Mano	**21.182**	**17.800**	**10.549**	**8.994**	**13.689**	**12.028**

With scaling data to over-trained settings, Mano consistently performed better than Muon and AdamW in the convergence speed.



(a) LLaMA-130M / `Pile` (b) LLaMA-350M / `Pile`

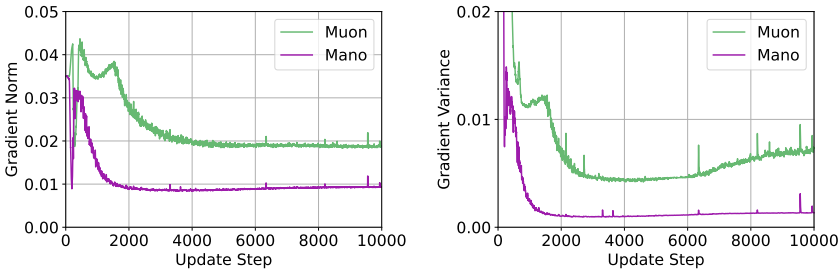
Figure: LLaMA-130M/350M trained on the `Pile` dataset for 10B tokens.

*We are expecting to expand these over-trained experiments to bigger LLMs and further understand this intriguing loss descent pattern of Mano in the later convergence stage.

Mano's Learning Dynamics: Gradient Stability

We investigate Mano's learning dynamics from the gradient perspective:

- Mano consistently exhibits a lower gradient variance compared to Muon, both under the same momentum coefficient $\mu = 0.95$.



(a) Gradient Norm (b) Gradient Variance

Figure: The average gradient norm and variance of a LLaMA-350M model.

- The SNR of Mano, or the gradient norm-to-variance ratio, is notably higher than that of Muon, suggesting superior training stability.

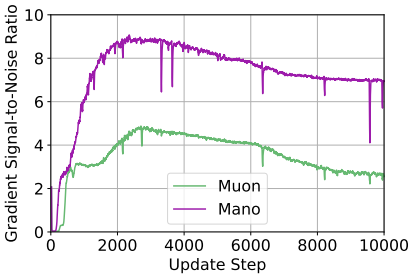


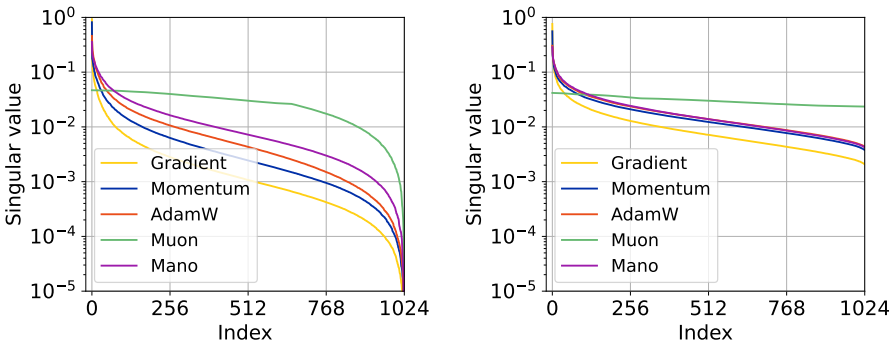
Figure: The gradient signal-to-noise ratio (SNR) of a LLaMA-350M model.

We hypothesize that Mano preserves the essential curvature information encoded within the original gradient step and promotes a more stable optimization landscape.

Mano's Learning Dynamics: Spectral Distribution

Spectral preconditioning has attracted widespread interest following the empirical success of Muon.

- Though Mano does not directly compute spectral information, the manifold normalization operation may also be viewed as a structural regularization approach.
- Fig. 6 shows that Mano lifts the update spectra while preserving the original singular distribution.
- While Muon performs whitening and flattens the spectrum, it discards the singular order information, which can be suboptimal from a theoretical perspective [2].



(a) Attention Layer (b) MLP Layer

Figure: The spectral distributions of (a) all attention layers and (b) all MLP layers from an LLaMA-350M model at the 1000 training step, including the model gradient, momentum, and the update matrix of the three optimizers.

Mano's Learning Dynamics: Computational Overhead

Computational Overhead: For each matrix parameter $\theta \in \mathbb{R}^{m \times n}$,

1. Row/column-wise normalization on θ_t and v_t, each $3mn$ FLOPs.
2. Tangent space projection requires at most $5mn$ FLOPs.

The theoretical FLOPs of Mano's update rule are at most $11mn$.

- Comparing to the baseline $6mnB$ FLOPs w.r.t. the number of inputs B passed through the layer, Mano consumes $11/6B$ FLOPs overhead, which is consistent for LLMs of different dimensions.
- Comparing to Muon's $5m/B$ FLOPs [3], Mano has significantly lower computational overhead for training large-scale LLMs.

We further report the practical computational cost of Newton-Schulz iteration (Muon) and manifold normalization (Mano) in Table. 3.

Method	**Attention**		**MLP**	
	Time	Memory	Time	Memory
LLaMA-3B	`BFloat16(2560, 2560)`		`BFloat16(2560, 6848)`	
Muon	4.09 (ms)	87.5 (MB)	8.19 (ms)	186.0 (MB)
Mano	0.08 (ms)	50.0 (MB)	0.36 (ms)	133.8 (MB)
LLaMA-7B	`BFloat16(4096, 4096)`		`BFloat16(4096, 11008)`	
Muon	14.83 (ms)	224.0 (MB)	30.22 (ms)	472.0 (MB)
Mano	0.34 (ms)	192.0 (MB)	1.45 (ms)	344.0 (MB)
LLaMA-70B	`BFloat16(8192, 8192)`		`BFloat16(8192, 16384)`	
Muon	110.79 (ms)	896.0 (MB)	184.33 (ms)	1536.0 (MB)
Mano	2.19 (ms)	512.0 (MB)	4.35 (ms)	1024.0 (MB)

Table: Computational cost comparison of Newton-Schulz iteration ($T = 5$) and the manifold normalization enforced by Mano on Attention and MLP matrices from LLaMA-3B, -7B, and -70B models in `BFloat16`. Reported values denote the average over 1000 PyTorch runs, with peak GPU memory usage measured via `torch.cuda` on NVIDIA RTX-4090 GPUs.

Mano's Key Design Choices

Table: Ablation experiment results on the final test perplexity of LLaMA-350M and -1.3B models' trained on the `Pile` dataset.

Test Perplexity	LLaMA-350M	LLaMA-1B
AdamW	11.803	9.945
Muon	11.022	9.227
Mano (Static Manifold $\mathcal{M}$)	10.684	9.254
Mano (Retracting $M_t = v_t$)	10.540	8.998
Mano	10.549	8.994

We further discuss several key design choices:

1. **Dynamic or Static Oblique Manifold?** Ablation results in Tab. 4 suggest that a static Oblique manifold (fixed to the 0-th dimension) provides comparable performance on LLaMA-350M but significantly worse on LLaMA-1.3B.
2. **Momentum with or w/o Retraction?** By updating $M_t = v_t$, we can update the momentum buffer as the tangent momentum on the parameter space. However, essentially identical results suggest further studies to understand the momentum mechanism.
3. **Input/Output Parameters?** Mano can be extended for general tensors with arbitrary dimensionality, including 1-D bias and > 2-D parameters. Nevertheless, we follow the Muon's implementation to optimize the input and output parameters using AdamW [3]. We hypothesize that adaptive learning strategies may fit better for vocabulary activations with high sparsity [4].
4. **Nesterov-style Momentum?** We provided NAG as an option in our implementation, similar to that of Muon [3, 5], while reporting all experiment results w/o NAG.

Relationship to Prior Optimizers

We now discuss Mano's relationship to the prior optimizers:

1. **Adafactor/Shampoo:** While Adafactor [6] and Shampoo [7] utilize second-moment-based normalization across parameter dimensions (similar to that of Mano), Mano achieves regularization through geometric constraints without relying on second-moment statistics.

2. **Spectral Optimizers:** Mano fundamentally differs from existing spectral optimizers (e.g., Muon, Conda) in that it does not rely on matrix-wide spectral information, but instead performs only vector-based operations to apply geometric constraints at each step. Mano's operation provides more efficiency and flexibility.

3. **LMO-based Optimizers:** Conditional Gradient methods rely on Linear Minimization Oracle (LMO) to optimize without explicit projection [8], where Mano deviates by employing manifold-style tangent space projections and unconstrained parameter updates.

4. **Hyperball and SSO:** Hyperball optimization [9] and the Spectral Sphere Optimizer (SSO) [10] regulate step sizes and weight norms via manifold retraction and μP-aligned spectral constraints [11]. Both approaches integrate with AdamW or Muon to control update magnitude, while Mano applies manifold normalization for directional control while retaining standard weight decay and descent schemes.

Conclusion & Future Works

In conclusion, Mano outperforms AdamW and Muon on existing experiments without introducing additional hyperparameters, computational cost, or memory overhead.

Many future works still remain...

1. Additional experiments, including over-training of larger LLMs and broader optimization regimes.
2. Further theoretical studies on convergence with momentum and training dynamics.
3. Exploring the other integration routines of manifold optimization into modern optimizers.

Thank you for your attention!

References I

[1] Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

[2] Weijie Su. Isotropic curvature model for understanding deep learning optimization: Is gradient orthogonalization optimal? *arXiv preprint arXiv:2511.00674*, 2025.

[3] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[4] Tim Large, Yang Liu, Minyoung Huh, Hyojin Bahng, Phillip Isola, and Jeremy Bernstein. Scalable optimization in the modular norm. *Advances in Neural Information Processing Systems*, 37:73501–73548, 2024.

[5] 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.

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

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

[8] Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025.

[9] Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them ii: From weight decay to hyperball optimization, 11 2025.

[10] Tian Xie, Haoming Luo, Haoyu Tang, Yiwen Hu, Jason Klein Liu, Qingnan Ren, Yang Wang, Wayne Xin Zhao, Rui Yan, Bing Su, et al. Controlled llm training on spectral sphere. *arXiv preprint arXiv:2601.08393*, 2026.

[11] Greg Yang, James B Simon, and Jeremy Bernstein. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.