Mano: Restriking Manifold Optimization for LLM Training

Authors: Yufei Gu and Zeke Xie
Speaker: Zeke Xie

Assistant Professor,
Information Hub, HKUST(GZ)

March 22, 2026

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

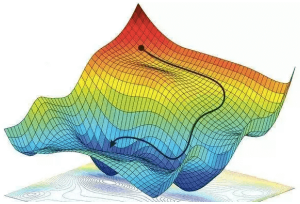
Overview

1. Preliminaries: From Adam to Muon

Background: Stochastic Optimization

- Gradient Descent (GD) for searching $\theta^\star = \operatorname{argmin} \mathcal{L}(\theta)$

$$\theta_{t+1} = \theta_t - \eta \frac{\partial \mathcal{L}(\theta)}{\partial \theta}$$



Searching the minimum.
Deep Neural Networks have the high dimensional and non-convex landscape.

- Difficulty: getting stuck in local minima and saddle points!
- Stochastic Gradient Descent (SGD): selecting some samples as a minibatch per iteration

$$\theta_{t+1} = \theta_t - \frac{\eta}{B} \sum_{(x^{(i)}, y^{(i)}) \in \mathcal{B}_t} \frac{\partial \mathcal{L}(\theta, (x^{(i)}, y^{(i)}))}{\partial \theta}$$

learning rate: η
batch size: B
the t-th-iteration minibatch: $\mathcal{B}_t$

From Second-order Optimization to Adam

In second-order optimization (e.g. Newton's method), the update rule is usually:

$$\theta_{t+1} = \theta_t - H^{-1}\nabla_\theta \mathcal{L} \tag{1}$$

- The optimizer preconditioner is the inverse Hessian H^{-1}.
- However, in deep learning, computing H^{-1} is expensive ($O(N^2)/O(N^3)$).
- In convex/quadratic settings, the expected squared gradient correlates with diagonal entries of the Hessian:

$$\mathbb{E}[g_t^2] \approx \text{diag}(H) \tag{2}$$

- Adam preconditioner estimates curvature (H^{-1}) separately per parameter with the diagonal matrix $\frac{1}{\sqrt{\hat{v}_t}+\epsilon}$, avoids off-diagonal estimates [1].

Adaptive Moment Estimation (Adam)

Adam is the most popular optimizer for accelerating training of deep networks [1]. (Kingma et al., 2014), >240k citations

Adam combines:

1. Momentum: moving average of past gradients
2. Adaptive Learning Rate: large learning rates along flat directions

Algorithm Adam

$$
\begin{aligned}
g_t &= \nabla \mathcal{L}(\theta_{t-1}) \\
m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\
v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
\hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\
\hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\
\theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon} \hat{m}_t
\end{aligned}
$$

AdamW: Adam with Decoupled Weight Decay Regularization

Algorithm AdamW

$$
\begin{aligned}
&g_t = \nabla \mathcal{L}(\theta_{t-1}) \\
&m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \\
&v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\
&\hat{m}_t = \frac{m_t}{1-\beta_1^t} \\
&\hat{v}_t = \frac{v_t}{1-\beta_2^t} \\
&\theta_t \leftarrow \theta_{t-1} - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_{t-1} \right)
\end{aligned}
$$

AdamW improves Adam with a Decoupled Weight Decay Scheme [2]:

- Control parameter norms through regularization.
- Better than L2 Regularization (add $\lambda \theta_{t-1}$ to g_t): decouples the optimal choice of weight decay factor from learning rate scheme.

Shampoo: Preconditioned Stochastic Tensor Optimization

The Hessian can be approximated by the gradient outer product:

$$\mathbb{E}[g_t g_t^T] = \mathbb{E}[\nabla \log p \nabla \log p^T] \approx \mathbb{E}[\nabla_\theta^2 \log p] = \mathbb{E}[H_t(\mathcal{L})]$$

Algorithm Shampoo

$g_t = \nabla \mathcal{L}(\theta_{t-1})$
$L_t \leftarrow L_{t-1} + g_t g_t^\top$
$R_t \leftarrow R_{t-1} + g_t g_t^\top$
$\theta_t \leftarrow \theta_{t-1} - \eta \cdot L_t^{-1/4} g_t R_t^{-1/4}$

Shampoo maintains memory-efficient left (L_t) and right (R_t) preconditioners,

- Maintains second moment information of the accumulated gradients.
- Precondition gradient steps using the Kronecker product of $L_t^{-1/4}$ and $R_t^{-1/4}$.

From Shampoo to Muon: Preconditioned tensor optimization

If we remove the preconditioner L_t and R_t in Shampoo; For $g_t = USV^T$, we have

$$\begin{aligned}\theta_t &\leftarrow \theta_{t-1} - \eta(g_t g_t^T)^{-1/4} g_t (g_t^T g_t)^{-1/4} \\ &= \theta_{t-1} - \eta(US^2U^T)^{-1/4}(USV^T)(VS^2V^T)^{-1/4} \\ &= \theta_{t-1} - \eta(US^{-1/2}U^T)(USV^T)(VS^{-1/2}V^T) \\ &= \theta_{t-1} - \eta UV^T\end{aligned}$$

Given matrix-sign function $\text{sign}_M(A) = \text{sign}_M(U_A \Sigma_A V_A^T) = U_A V_A^T$ $(\Sigma_A^* = I)$,

$$\theta_t \leftarrow \theta_{t-1} - \eta UV^T = \theta_{t-1} - \eta \underset{M}{\text{sign}}(g_t)$$

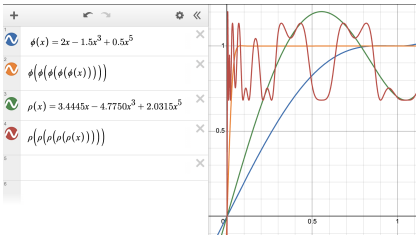
We have derived the Muon optimizer (w/o momentum)!

Muon: MomentUm Orthogonalized by Newton-Schulz

However, performing SVD decomposition on the momentum is costly. We need an effective derivation of orthogonal eigenvectors with eigenvalues close to 1 $\Sigma^* = I$.

Newton-Schulz iteration: approximates the nearest semi-orthogonal matrix:

$$X_{t+1} = aX_t + bX_t(X_t^T X_t) + cX_t(X_t^T X_t)^2$$
$$\Rightarrow a = 3.4445, b = 4.7750, c = 2.0315$$



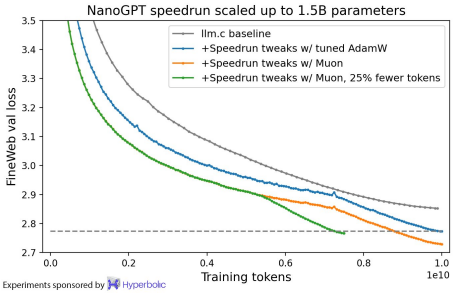
Orthogonalization mat effectively increases the scale of other "rare directions" which have small magnitude in the update but are nevertheless important for learning [3].

Muon: MomentUm Orthogonalized by Newton-Schulz

Algorithm Muon, Moonshot ver.

$$g_t = \nabla \mathcal{L}(\theta_{t-1})$$
$$M_t \leftarrow \mu M_{t-1} + g_t$$
$$O_t \leftarrow \text{Newton-Schulz}(\mu M_t + g_t)^a$$
$$\theta_t \leftarrow \theta_{t-1} - \eta(0.2\sqrt{(m,n)}^b O_t + \lambda \theta_{t-1})$$

[a] Nesterov-style momentum.
[b] Update RMS is aligned with AdamW.



State-of-the-art LLMs such as DeepSeek, Kimi, and GLM leverage Muon for their large-scale pretraining pipelines.

Modern Optimizer Paradigm: Adaptive or Spectral?

Adam-based Optimizers:

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

Spectral Optimizers, eg., Muon:

- Spectral normalization explores all gradient directions with the same magnitude.
- Introduces Newton-Schulz iteration to approximate matrix orthogonalization (the `msign` function); But also eliminates the curvature information encoded.

Have we exhausted all optimization paradigms?

2. Mano: Reformed Manifold Normalization

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.,

$$\min_{x \in \mathcal{M}} f(x) \tag{3}$$

where $f : \mathcal{M} \to \mathbb{R}$.

Riemannian-SGD performs gradient updates on Riemannian manifolds through the following operations [4]:

$$\begin{cases} g_t = \nabla f(\theta_t) \\ v_t = \mathbf{proj}_{\mathcal{T}_{\theta_t}\mathcal{M}}(g_t) \\ \theta_{t+1} = \mathbf{proj}_{\mathcal{M}}(\theta_t - \eta_t v_t) \end{cases} \tag{4}$$

Traditional Manifold Optimization

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 the metric defined as $g_{ij} = \delta_{ij}$.

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: 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.

- 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.

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

Table: *This experiment only serves as an intuitive justification.*

This suggests the Oblique manifold best captures the geometric properties of the learning trajectory of AdamW.

The Oblique Manifold and Rotation

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.

We hypothesize that for the Oblique manifold, only applying column-wise normalization may be insufficient, so 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.
- By consistently applying this rotation, we effectively perform manifold normalization to every gradient dimension.

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**:

$$\begin{cases} g_t = \nabla f(\theta_t) \\ \hat{\theta}_t = \mathcal{N}_{\mathcal{M}}(\theta_t) \\ v_t = \textbf{proj}_{\mathcal{T}_{\hat{\theta}_t}\mathcal{M}}(g_t) \\ \hat{v}_t = \mathcal{N}_{\mathcal{M}}(v_t) \\ \theta_{t+1} = \theta_t - \eta_t\, \hat{v}_t \end{cases} \tag{5}$$

Mano: Manifold Normalized Optimizer [Yufei Gu and Zeke Xie, 2026]

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

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.

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

Assume that $f(\theta)$ *is an L-smooth function,* f *is lower bounded as* $f(\theta) \geq f_{\inf}$, $\mathbb{E}[\xi] = 0$ *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*

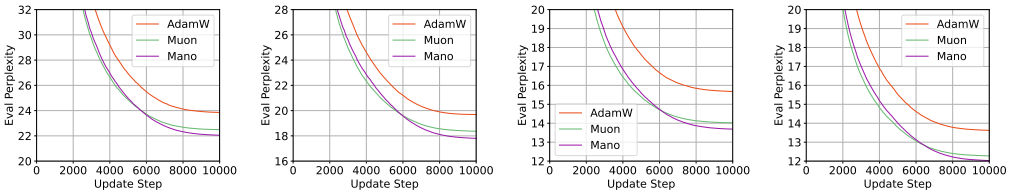
$$\min_{t=0,\ldots,T} \mathbb{E}[\|\nabla f(\theta_t)\|^2] \leq \frac{1}{\sqrt{T+1}}(C_1 + C_2), \tag{6}$$

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

3. Mano's Empirical Results & Dynamics

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) Qwen3-0.6B / `Pile` (d) 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.

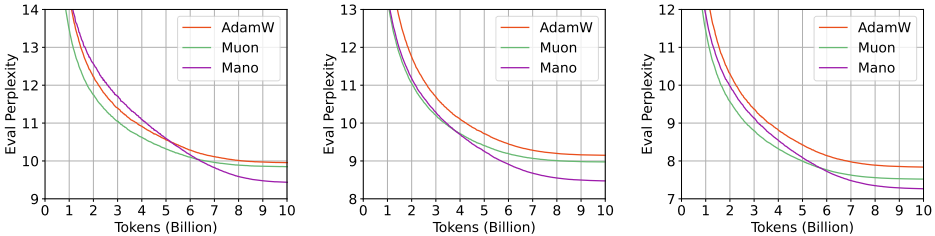
Mano's Empirical Results

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**

Mano's Empirical Results

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

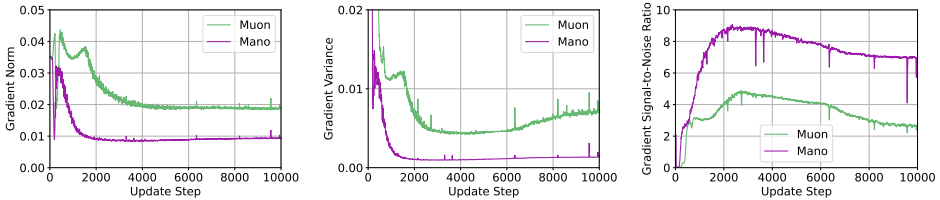


(a) LLaMA-130M / `Pile` (b) LLaMA-350M / `Pile` (c) LLaMA-1B / `Pile`

Figure: LLaMA-130M, -350M and -1B models trained on the `Pile` dataset for 10B tokens.

Mano's Learning Dynamics: Gradient Stability

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



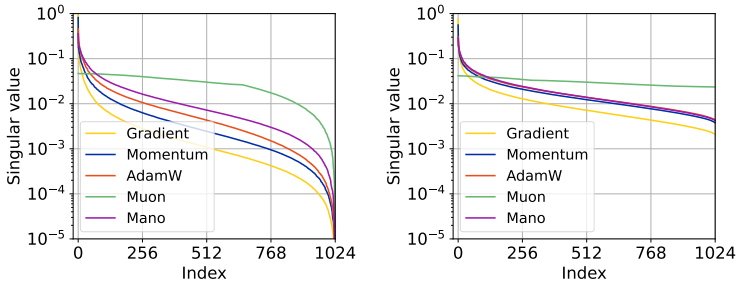
(a) Gradient Norm (b) Gradient Variance (c) Gradient SNR

Figure: The average gradient norm, variance, and 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

Though Mano does not directly compute spectral information, it may also be viewed as a structural regularization approach.



(a) Attention Layer (b) MLP Layer

While Muon performs whitening and flattens the spectrum, it discards the singular order information, which can be suboptimal from a theoretical perspective [5].

Mano's 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.

Mano's Computational Overhead

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: Reported values denote the average over 1000 PyTorch runs, with peak GPU memory usage measured via `torch.cuda` on NVIDIA RTX-4090 GPUs.

4. An Improved Version Mano-V2

Mano-V2: Two Lines of Code Outperforms Newton-Schulz Iteration

Row/Column normalization of the Parameters are unnecessary, and removing it improves performance in final convergence.
(Oblique Manifold→ Product Manifold of Spheres)

```
class Mano_V2:
tan_mt = g - (torch.sum(g*p.data, dim=dim, keepdim=True)
    * p.data)
update = tan_mt / (torch.norm(tan_mt, p=2, dim=dim,
    keepdim=True) + eps)
```

The Mano's design can be simplified to two elements: **Tangent Projection** and **Normalization** applied axis-wise (row/column).

- Tangent projection could be greatly overlooked in high-dimensional optimization.
- Row/Column-wise Normalization has been proven useful in many different optimizers (AdamW/Muon), but TP+Norm alone, it surprisingly outperforms expensive Newton-Schulz iterations in Muon.

Mano-V2: Two Lines of Code Outperforms Newton-Schulz Iteration]

Algorithm The Mano_V2 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**

$g_t \leftarrow \nabla f(\theta_t)$

$M_t \leftarrow \mu M_{t-1} + g_t$

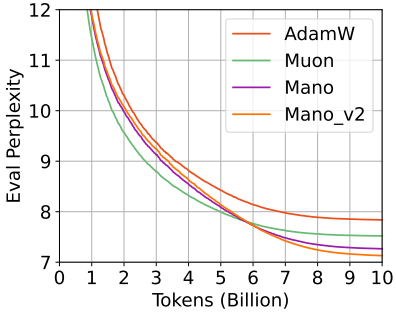
$v_t \leftarrow M_t - \theta_t \odot \langle M_t, \theta_t \rangle_{(t \bmod 2)}$ {Tangent Momentum}

$\hat{v}_t \leftarrow v_t \oslash \|v_t\|_{2,(t \bmod 2)}$ {Manifold Normalization}

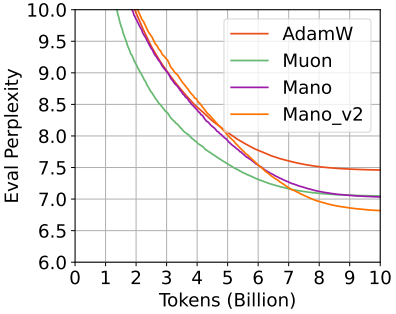
$\theta_{t+1} \leftarrow \theta_t - \eta_t(0.2\sqrt{n_k}\,\hat{v}_t + \lambda\theta_t)$

end for

Mano-V2: Empirical Results



(c) LLaMA-1.3B / `Pile`



(d) LLaMA-3B / `Pile`

Figure: Better Scaling. Mano_v2 significantly improves performance on larger model and more data.

5. Conclusion & Future Works

Relationship to Prior Optimizers

1. **Adafactor/Shampoo** [6, 7] utilize second-moment-based normalization across parameter dimensions. Mano achieves regularization through geometric constraints without relying on second-moment statistics.

2. **Spectral Optimizers**, e.g., Muon [3], Conda [8] rely on matrix-wide spectral information, while Mano performs only vector-based operations to apply geometric constraints.

3. **LMO-based Optimizers** [9, 10] are conditional gradient methods without explicit projection, where Mano deviates by employing manifold-style tangent space projections and unconstrained parameter updates.

4. **Hyperball/SSO** [11, 12] regulate step sizes and weight norms via manifold retraction and μP-aligned spectral constraints [13] to control update magnitude. Mano normalizes the raw gradient with standard weight decay and descent.

Conclusion & Future Works

Mano outperforms AdamW and Muon on existing experiments without introducing additional hyperparameters under even less computational cost/memory overhead.

Many future works still remain...

1. Scaled-up experiments at the production level of base model pretraining (100B to 1T Tokens) and broader optimization regimes.
2. Exploring the generalization of tangent projection and the theoretical position of Mano_v2.
3. Further theoretical studies on convergence and training dynamics.
4. Beyond pretraining.

Thank you for your attention!

References I

[1] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, 2015.

[2] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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

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

References II

[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] Junjie Wang, Pan Zhou, Yiming Dong, Huan Li, Jia Li, Xun Zhou, Qicheng Lao, Cong Fang, and Zhouchen Lin. Conda: Column-normalized adam for training large language models faster. *arXiv preprint arXiv:2509.24218*, 2025.

[9] 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.

[10] Thomas Pethick, Wanyun Xie, Mete Erdogan, Kimon Antonakopoulos, Tony Silveti-Falls, and Volkan Cevher. Generalized gradient norm clipping & non-euclidean (l_0, l_1)-smoothness. *arXiv preprint arXiv:2506.01913*, 2025.

[11] 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.

References III

[12] 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.

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