Mano: Restriking Manifold Optimization for LLM Training

Yufei Gu [1] **Zeke Xie**† [1]

Abstract

While large language models (LLMs) have emerged as a significant advancement in artificial intelligence, the hardware and computational costs for training LLMs are also significantly burdensome. Among the state-of-the-art optimizers, AdamW relies on diagonal curvature estimates and ignores structural properties, while Muon applies global spectral normalization at the expense of losing curvature information. In this study, we restriked manifold optimization methods for training LLMs, which may address both optimizers' limitations, while conventional manifold optimization methods have been largely overlooked due to the poor performance in large-scale model optimization. By innovatively projecting the momentum onto the tangent space of model parameters and constraining it on a rotational Oblique manifold, we propose a novel, powerful, and efficient optimizer **Mano** that is the first to bridge the performance gap between manifold optimization and modern optimizers. Extensive experiments on the LLaMA and Qwen3 models demonstrate that Mano consistently and significantly outperforms AdamW and Muon even with less memory consumption and computational complexity, respectively, suggesting an expanded Pareto frontier in terms of space and time efficiency.

1. Introduction

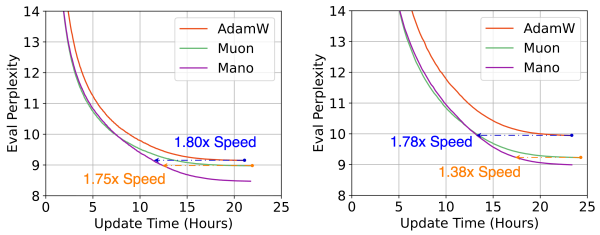
Large Language Models (LLMs) are represented as a major milestone in artificial intelligence and computing technology (Kumar, 2024; Gao et al., 2025). Nevertheless, training LLMs with billions of parameters requires specialized hardware accelerators, such as GPUs and NPUs, which consumes sunstantial energy and result in enormous computational costs (Samsi et al., 2023; Chowdhury et al., 2025). Consequently, advanced optimization methods are critical for reducing the training costs of LLMs.

Adam is widely regarded as one of the most popular optimizers in deep learning, with its variants being extensively used to train deep neural networks (DNNs) across a wide range of tasks (Kingma, 2014). In particular, AdamW, which decouples weight decay from the original gradient update of Adam, has been the dominant optimizer in training LLMs with broad infrastructure support for large-scale distributed training (Loshchilov & Hutter, 2017; Kunstner et al., 2024). However, by relying on the diagonal estimates of per-parameter curvature, Adam-based optimizers ignore spectral information and fail to capture the subspace structures of the parameter matrix. To address this limitation, Muon introduces the Newton-Schulz iteration to approximate matrix orthogonalization, allowing uniform exploration across all spectral directions in the loss landscape (Jordan et al., 2024). However, this spectral normalization strategy eliminates the curvature information encoded in the gradient and momentum, which may lead to sub-optimal performance on certain occasions (Su, 2025).

For optimization problems with a so-called manifold structure, algorithms have been developed to exploit this special structure from the perspective of differential geometry and numerical analysis, referred to as manifold optimization methods (Absil et al., 2008; Hu et al., 2020; Fei et al., 2025). However, these traditional manifold optimization methods have been overlooked for LLMs. In this study, we seek to adapt and extend manifold optimization methods to LLM training, where the underlying manifold structure of natural language or optimal parameters is not explicitly known or well understood, relaxing the constraints and assumptions typically required. Instead of retracting and constraining the parameters onto some manifold's surfaces, we demonstrate that by only projecting the momentum onto the tangent vector spaces of the parameters and mapping it to the Oblique manifold, the constrained gradient steps emerged as a simple yet effective learning direction in the solution space with strong escape-from-local-minima properties. Furthermore, we demonstrate the empirical advantages of periodically rotating the Oblique manifold, a process equivalent to an alternating column-row normalization scheme applied at each optimization step.

[1]xLeaF Lab, The Hong Kong University of Science and Technology (Guangzhou). †Correspondence to: Yufei Gu <ygu167@connect.hkust-gz.edu.cn>, Zeke Xie† <zekexie@hkust-gz.edu.cn>.

Preprint. January 29, 2026.



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

Figure 1. One day pretraining experiment of LLaMA-350M and -1.3B models on the `Pile` dataset for 10B and 2.8B tokens respectively. Our proposed optimizer Mano achieves $1.75\times$ and $1.38\times$ the convergence speed of Muon in terms of wall-clock time. This advantage is expected to further increase with reduced computational overhead and faster convergence speed.

By restriking manifold optimization with multiple reformed strategies, we propose a new class of optimizer, **MA**nifold-**N**ormalized **O**ptimizer (**Mano**). Throughout this paper, we seek to unravel the hidden potential of manifold optimization methods in today's infrastructure of training neural networks. This work mainly made three contributions.

- To the best of our knowledge, we are the first to revisit and restrike manifold optimization techniques for LLM training and highlight its promising potential when combining with the proposed reform strategies, while traditional manifold methods have been largely overlooked due to poor performance on LLMs.
- We design a novel, powerful, and efficient optimizer **Mano** for LLM training. It consumes less memory and has significantly lower computational complexity than popular modern optimizers, such as Adam and Muon.
- Mano is the first manifold optimizer that works well in LLM training, significantly outperforming Adam and Muon in test perplexity across both token consumption and wall-clock time. We also observe that it can effectively update model parameters with reduced gradient variance, theoretically suggesting better convergence. Mano restrikes a promising future of the reformed manifold optimization paradigm for LLM training.

2. Related Works

This section reviews prior works on efficient optimizers designed for pretraining LLMs and manifold optimization techniques in the field of deep learning.

2.1. Optimizers for LLM Pretraining

Adam-based optimizers remain the most widely used optimizers in the field of deep learning, including both the pretraining and fine-tuning of decoder-only transformers (Zhao et al., 2024b). Its popularity stems from its simplified design, flexibility in per-parameter adaptive learning rate, and robustness in performance across diverse domains. However, the first- and second-moment estimates of AdamW consume double the memory footprint of model weights or gradients, resulting in a significant memory overhead, especially for LLMs at scale. Several approaches have emerged to design more memory-efficient optimizers. Adam-mini leverages block-wise learning rate schedules based on Hessian partitions (Zhang et al., 2024a; Wang et al., 2025); Lion uses momentum-sign updates to eliminate the need of second moment (Chen et al., 2023), and Cautious-Adam/Lion applies gradient-aligned selective updates (Liang et al., 2024); SOAP instead applies AdamW updates in the Shampoo eigenbasis while amortizing eigendecomposition costs across multiple steps (Gupta et al., 2018; Vyas et al., 2024). Other notable methods include SWAN (Ma et al., 2024), MARS (Yuan et al., 2024), Sophia (Liu et al., 2023), and etc.

Another particularly promising line of research focuses on matrix-based spectral preconditioning methods. Muon, introduced by Jordan et al. (2024) in 2023, utilized the Newton-Schulz iteration to perform spectral normalization on the update steps. This approximation to the matrix-sign function produces a semi-orthogonal momentum update that normalized the magnitude at all spectral directions, including those low-magnitude but important directions for model generalization. Empirical studies have later extended Muon to scaled-up LLM training with stability, and demonstrated improved efficiency with halved memory consumption in comparison to AdamW (Liu et al., 2025; Shah et al., 2025; Team et al., 2025; Zeng et al., 2025). Benchmarking studies also demonstrate that matrix-based optimizers with spectral preconditioning (e.g., Kron, Muon, SOAP) often outperform scalar-based counterparts (e.g., AdamW, Lion, MARS), though no optimizer significantly outperforms in every tested scenario (Schlotthauer et al., 2025; Wen et al., 2025b; Semenov et al., 2025). We are reminded that empirical studies cannot investigate all possible regimes, even with theoretically guided insights and standardized experimental setups. In this paper, we offer empirical evidence across varied contexts, prioritizing hypothesis testing and investigation over the assertion of definitive conclusions.

2.2. Manifold Optimization in Deep Learning

Geometric optimization methods are designed to exploit the intrinsic geometric structures of the target objective function and have emerged as promising solutions to problems in various fields, including deep learning (Fei et al., 2025). For objective functions defined on a Riemannian manifold with a differentiable structure and a smooth inner product, various manifold optimization techniques are proposed to find optimal solutions on the manifold, including

diverse Riemannian optimizers that have been developed as geometrically-aware counterparts of conventional Euclidean methods, such as SGD, SGD-M, RMSProp, Adam, AdaGrad, AMSGrad, etc. (Bonnabel, 2013; Zhang et al., 2016; Bécigneul & Ganea, 2018; Roy et al., 2018). A broad spectrum of architecture-dependent manifold optimization techniques has been proposed for CNNs (Ozay & Okatani, 2016; Wang et al., 2020), RNNs (Arjovsky et al., 2016; Wisdom et al., 2016; Huang et al., 2018; Jing et al., 2019), GNNs (Zhu et al., 2020; Liu et al., 2021; de Ocáriz Borde et al., 2023), and other deep learning techniques (Zhang et al., 2018; Chaudhry et al., 2020).

While manifold optimization methods are well-established in other deep learning domains, they have been overlooked in the LLM literature and practices, leaving a relatively small fraction of studies exploring geometrically aware training strategies for LLMs. Recent literature has explored parameter-efficient low-rank training for LLMs through the lens of Riemannian manifolds (Jiang et al.; Zhang et al., 2024b; Mo et al., 2025; Park et al., 2025). Similar strategies have been extended to gradient tracking (Rajabi & Rambhatla, 2024; Rajabi et al., 2025), representation regularization (Zhang & Dong, 2025; Kingswell et al., 2025; Wren et al., 2025), and parameter pruning (Liu et al., 2024). Explicit discussions on Mano's relationship to other prior optimizers are provided in the Appendix D.

3. Preliminaries

This section revisits the traditional concept of manifold optimization and Riemannian stochastic gradient descent (SGD), discussing how our reformulated method is motivated.

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 $f : \mathcal{M} \to \mathbb{R}$. With the above definition, 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 $g_{ij} = \delta_{ij}$. Bonnabel (2013) generalizes SGD to perform gradient updates on Riemannian manifolds through the following operations:

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

First, the gradient vector g_t is orthogonally projected onto the tangent space $\mathcal{T}_{\theta_t}\mathcal{M}$ to determine the steepest ascent direction v_t for the objective function. A gradient step is then performed in the direction of $-\eta_t v_t$ and mapped back to the manifold surface. While the exponential map provides the geometrically exact update, its high computation cost is often replaced with numerical retractions. Retraction typically performs orthogonal projections back onto the manifold $\mathcal{M}$, serving as efficient first-order approximations that maintain the manifold constraint (Bonnabel, 2013).

However, traditional Riemannian manifold optimization strategies often fail to generalize to modern neural networks on general tasks, particularly LLMs. Certain manifolds–such as the Stiefel manifold–often require expensive matrix decompositions (e.g., SVD, QR, etc.), which impose inefficiency in optimization. Furthermore, manifold constraints can restrict the model's ability to explore the loss landscape, especially when the geometric structure of the chosen manifold does not align with the underlying objective function. Instead of framing natural language modeling or the optimal parameter solution θ^* within an arbitrary Riemannian manifold, we hypothesize that the learning trajectory and each constituent update step can be mapped onto some "smooth surfaces" with geometric structures that facilitate convergence and help to escape from local minima. Motivated by this hypothesis, we will discuss how we designed a new optimizer in the next section.

4. Methodology

In this section, we reformulate traditional manifold optimization strategies into a momentum-driven optimizer and detail the configuration of rotating manifold normalization.

4.1. Reformed Manifold Optimization

To formulate our manifold optimization methodology, we define a tangent space projector as $\mathbf{proj}_{\mathcal{T}_P\mathcal{M}}(Q)$ of matrix Q on the first-order approximation of the manifold surface $\mathcal{M}$ around P, and a manifold normalization operation $\mathcal{N}_{\mathcal{M}}(A)$ to constrain matrix A on the target manifold. 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 update rule as follow:

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

We emphasize that this reformed update rule with manifold constraint is different from the original definition of manifold optimization stated in Eq. 1, which defined the function f (parameters) on the Riemannian manifold. Our update rule can be viewed as imposing a soft manifold constraint:

it projects each update step onto the manifold surface defined by the parameters θ, while keeping the objective and solution unchanged in the Euclidean space.

4.2. Manifold Selection and Design

Among popular matrix manifolds, we select the Oblique manifold for our update rules due to its computational efficiency in manifold normalization. This choice is also driven by the hypothesis that smoother surface geometry reduces trajectory distances, thereby aiding convergence. Tab. 1 presents the average geodesic distance across 1000 consecutive update steps of a Qwen3-0.6B model trained with AdamW. Our observations reveal that the Oblique manifold yields the shortest geodesic distance compared to the Sphere and Stiefel manifolds, providing an intuitive geometric justification for our design, which better captures the model's natural learning trajectory.

Table 1. 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

Notations We denote the Oblique manifold as $\mathcal{OB}(n, m)$, the set of $\mathbb{F}-$valued matrices with unit norm column, endowed with the metric from the embedding (Roberts & Ursell, 1960; Absil & Gallivan, 2006; Huang et al., 2020). We define the following operators:

- 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 $\langle Q, P\rangle_d^{(j)} = \langle Q^{(j)}, P^{(j)}\rangle$.
- Dimension-wise norm ($\|\cdot\|_{2,k}$): For the k-th dimension, $\|P\|_{2,d}^{(i)} = \|P^{(i)}\|_2$. We further denote $\|P\|_{2,0}^{(i)} = \|P_{i,:}\|_2$ and $\|P\|_{2,1}^{(j)} = \|P_{:,j}\|_2$ for column- and row-wise norm respectively.

We thus define the orthogonal projection of a vector Q onto the tangent space $\mathcal{T}_P\mathcal{OB}$ at point P as

$$\mathbf{proj}_{\mathcal{T}_P\mathcal{OB}}(Q) = Q - \langle Q, P\rangle_d \odot P \tag{4}$$

and the normalization operator, which maps a vector A in the ambient space back to the Oblique manifold as

$$\mathcal{N}_{\mathcal{OB}}(A) = A \oslash \|A\|_{2,d} \tag{5}$$

Both operations are fully supported by modern machine learning frameworks, such as TensorFlow and PyTorch.

When integrating the Oblique manifold into the update rule (Eq. 3), we observe that enforcing the manifold constraint via column-wise normalization assumes that column directions dominate row directions in LLM parameter matrices. However, such an assumption remains unvalidated; for instance, the Muon optimizer demonstrates that all spectral directions hold comparable importance to model convergence. To address the potential insufficiency of the standard Oblique manifold, we introduce a **rotational manifold** scheme. The reformed approach alternates between column-wise normalization on odd iterations and row-wise normalization on even iterations. By consistently applying this rotation to both the tangent space of the parameters and the update step, we effectively create a custom manifold with oscillating orientation across iterations. Integrating this rotational scheme to the Oblique manifold with out update rule yields the **Mano** optimizer, as detailed in Alg. 1.

4.3. The Mano Optimizer

Algorithm 1 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 coefficient λ.

1: Initialize $M_0 \leftarrow \mathbf{0} \in \mathbb{R}^{m\times n}, t \leftarrow 0$.
2: **for** each step **do**
3: $\quad g_t \leftarrow \nabla f(\theta_t)$
4: $\quad M_t \leftarrow \mu\, M_{t-1} + g_t$
5: $\quad k \leftarrow t \bmod 2$ ▷ Rotating Manifold
6: $\quad \hat{\theta}_t \leftarrow \theta_t \oslash \|\theta_t\|_{2,k}$ ▷ Manifold Normalization
7: $\quad v_t \leftarrow M_t - \hat{\theta}_t \odot \langle M_t, \hat{\theta}_t\rangle_k$ ▷ Tangent Momentum
8: $\quad \hat{v}_t \leftarrow v_t \oslash \|v_t\|_{2,k}$ ▷ Manifold Normalization
9: $\quad \theta_{t+1} \leftarrow \theta_t - \eta_t(0.2\sqrt{n_k}\,\hat{v}_t + \lambda\theta_t)$
10: **end for**

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

- The parameters θ_t are not constrained on the manifold; the update process follows weight decay and Euclidean descent rather than retraction.
- Rotating Oblique manifold instead of a static geometric structure, alternating through each parameter dimension at every time step (Line 5).
- We first compute the tangent momentum via parameter-space manifold projection (Lines $6-7$), then apply a momentum-space manifold constraint to ensure the update step remains on the Oblique surface (Line 8).

In comparison to SGD-M, Mano only adds two-step column-/row-wise normalizations and one-step tangent-space projection, introduces no additional hyperparameters, and re-

quires no problem-specific assumptions from a geometric or differential perspective. Its implementation is highly streamlined, enabling ease of use and practical adoption from standard SGD-momentum. The memory overhead is comparable to SGD-momentum or Muon, halving the footprint of Adam-based optimizers. The computational cost of applying manifold normalization is also greatly reduced as no `MatMul` operations are involved, in comparison to the Newton-Schulz iteration. We proceed to discuss several key aspects of Mano's implementation and analyze its computational overhead w.r.t. SGD and Muon's Newton–Schulz iterations.

Rotational Manifold Scheme. We notice that the iterative procedure of alternating row and column normalization, known as the Sinkhorn-Knopp iteration, converges the matrix input to a doubly stochastic matrix (Knight, 2008). This set of matrices forms a convex manifold that has been widely studied in the context of manifold optimization (Douik & Hassibi, 2019). Nevertheless, our empirical results show that applying this iterative normalization procedure intermittently and constraining the tangent vectors only to the Oblique manifold serves as an efficient and effective regularization strategy in LLM training. By conducting ablation studies on static or dynamic manifold normalization in Sec 5.4, we hypothesized that our 'manifold rotation' strategy guarantees benign properties on the parameters θ indirectly, leaving sufficient space for future investigation.

Consistent Update RMS. Liu et al. (2025) proposed to set the update RMS of Muon to the range of 0.2 to 0.4 to be similar to that of AdamW, and uses a rescaling factor of 0.2 in their final implementation (Liu et al., 2025). We follow this conclusion and use the same rescaling factor of 0.2 in all of our experiments for sharing hyperparameters and enabling valid comparison among AdamW, Muon, and Mano. All update matrices $v_t \in \mathbb{R}^{m\times n}$ with column-wise normalization theoretically have an update RMS of $\sqrt{1/m}$ (row-wise normalization with update RMS of $\sqrt{1/n}$). Therefore, we set the rescaling variable of Mano to $0.2\sqrt{n_k}$, for the dimension $n_k \in \{m, n\}, n_0 = m, n_1 = n$.

4.4. Theoretical Analysis

Computational Overhead in FLOPs. For each matrix parameter $\theta \in \mathbb{R}^{m\times n}$, the Mano optimizer computes two column-wise normalization on the parameters θ_t and the update vectors v_t, each requiring $3mn$ FLOPs, which is identical to row-wise normalization. The tangent space projection consumes at most $5mn$ FLOPs due to no `MatMul` operations being involved. Therefore, the theoretical FLOPs of Mano's update rule are at most $11mn$. For the baseline amount of FLOPs being $6mnB$ for the number of inputs B passed through the layer, the FLOP overhead of Mano is at most $11/6B$, which is consistent for LLMs of different dimensions. In comparison to Muon's FLOP overhead of $5m/B$ (Jordan et al., 2024), the computational cost of Mano can be neglected in LLM training.

Convergence Analysis. Theorem 1 proves that Mano has convergence guarantees under common assumptions and a simplified setting with no momentum and a static Oblique manifold. The proof is relegated to Appendix E.

Theorem 1 (Convergence of Mano w/o Momentum)**.** *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 subsampling, $\sin(\phi_t^{(j)}) \geq \gamma > 0$ 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*

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

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

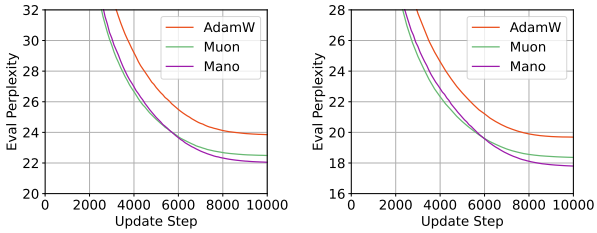
5. Experiments

5.1. Experiment Setup

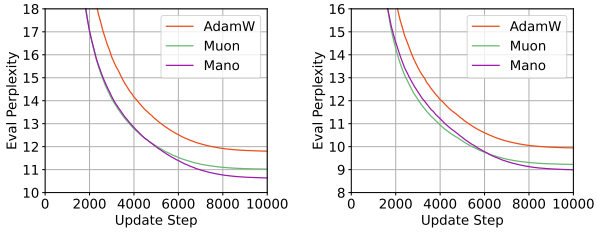
In this paper, we studied the pretraining performance of five popular models of two class of architectures, including LLaMA-$\{130\text{M}, 350\text{M}, 1.3\text{B}\}$ and Qwen3-$\{0.6\text{B}, 1.7\text{B}\}$, and two common text corpus, including `C4` and `Pile` (Raffel et al., 2020; Gao et al., 2020). We utilized a total batch size of 512 and evaluated models at 10000 training steps following the experimental setup described in Zhao et al. (2024a) and Raffel et al. (2020). We also report experimental results for the LLaMA-130M and -350M models, trained on 10B tokens from the `Pile` dataset, surpassing the Chincilla optimal scaling law recommendations (Hoffmann et al., 2022). We set $(\beta_1, \beta_2) = (0.9, 0.95)$ for AdamW and the same momentum coefficient $\mu = 0.95$ for Mano and Muon, with the other hyperparameters available in Appendix B.

5.2. Experiment Results

We present the pretraining dynamics of LLMs in test perplexity and compare the performance on Mano to the baseline optimizers AdamW and Muon. Fig. 2 reports a consistent advantage in sample efficiency of Mano of two LLaMA models on the `C4` and `Pile` dataset. We observe that Mano exhibits a distinct convergence pattern: though its initial convergence may be slower than that of Muon, its loss reduction in the later stages is surprisingly faster than both AdamW and Muon. While the loss curves of the two baseline optimizers plateau, Mano continues to progress at a nearly constant rate toward the global minimum and ultimately surpasses Muon, which may suggest that Mano is more effective at escaping local minima. We also observe



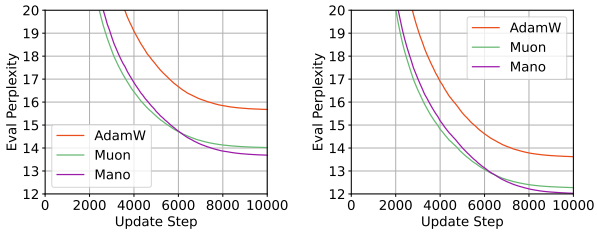
(a) LLaMA-350M / C4 (b) LLaMA-1.3B / C4



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

Figure 2. LLaMA-350M and -1.3B models trained on the C4/en and Pile dataset for 10000 steps with three different optimizers: AdamW, Muon, and Mano. Mano demonstrated a faster convergence speed than both popular optimizers with the simplest implementation and computational cost.

that, for larger models, the point at which Mano's loss descent rate surpasses that of Muon occurs later, potentially due to their larger data-scaling optima.

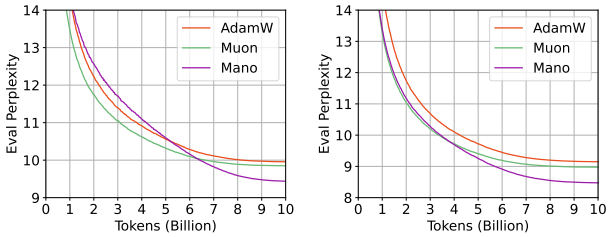


(a) Qwen3-0.6B / Pile (b) Qwen3-1.7B / Pile

Figure 3. Qwen3-0.6B and -1.7B models trained on the Pile dataset for 10000 steps with three different optimizers: AdamW, Muon, and Mano. The performance advantage of Mano is model-transferrable.

Fig. 3 reports the replicated experiments on the Qwen3 architecture and the Pile dataset, demonstrating that the performance advantage of Mano can be transferred across different model architectures. As Mano consistently improves the sample efficiency of pretraining LLMs in comparison to Muon, we hypothesize that projecting and constraining the training trajectory onto a manifold more accurately captures the steepest-descent path in the original solution space, without limiting the expressivity of LLM parameters. We further provide empirical results on the Qwen3-0.6B model with different maximum learning rate in Appendix B.2 to validate Mano's robustness across learning rate settings.

We further provide the experiment results with more training tokens. Due to computational constraints, we train LLaMA-130M and -350M models on the Pile dataset for $10B$ tokens, with results provided in Fig. 4. We notice that Mano performed worse than AdamW in the middle of training the LLaMA-130M model, but ultimately achieved the best performance across the three optimizers. 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.



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

Figure 4. LLaMA-130M and -350M models trained on the Pile dataset for 10B tokens. We demonstrated that with data scaling, Mano consistently performed better than Muon and AdamW in the ultimate convergence speed.

5.3. Learning Dynamics

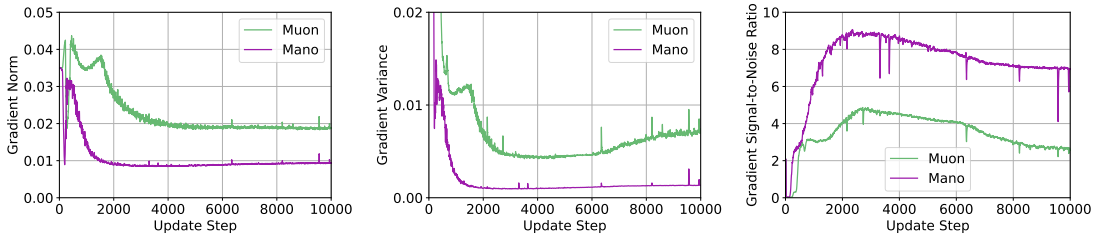
In this subsection, we will delve into the learning dynamics of the Mano optimizer and compare it to the baseline optimizers AdamW and Muon from multiple perspectives.

Gradient Stability. To understand the internal training dynamics of Mano, we reported the average gradient norm, variance, and the Signal-to-Noise (SNR) ratio in Fig. 5. Our empirical observations reveal the intrinsic advantage for Mano, that it consistently maintains a lower gradient variance compared to Muon, when operating under the same momentum coefficient and a similar update RMS. The SNR of Mano is notably higher than of Muon, suggesting superior training stability. We hypothesize that Mano's manifold normalization approach preserves the essential curvature information encoded within the original gradient step and promotes a more stable optimization landscape. This relationship is further evidenced by the spectral distributions of both optimizers, discussed in the following paragraph.

Spectral Distribution. Spectral preconditioning has attracted widespread interest following the empirical success of Muon. We analyze Mano from a spectral perspective by comparing the spectra of its update matrices with those of AdamW and Muon in Fig. 6. We observe that Mano achieved efficient spectral regularization through manifold

Table 2. 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. Mano yields consistent gains in sample efficiency.

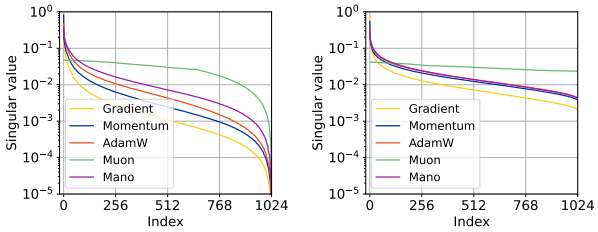
Datasets	`C4/en`		`Pile`			
Models	Llama-350M	Llama-1.3B	Llama-350M	Llama-1.3B	Qwen3-0.6B	Qwen3-1.7B
AdamW	23.852	19.690	11.803	9.945	15.679	13.624
Muon	22.491	18.365	11.022	9.227	14.020	12.276
Mano	**21.182**	**17.800**	**10.549**	**8.994**	**13.689**	**12.028**



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

Figure 5. The average (a) Gradient norm, (b) Gradient variance, and (c) Gradient Signal-to-Noise Ratio (SNR) of LLaMA-350M model parameters trained on the `Pile` dataset. The SNR is calculated as the norm-to-variance ratio. As an indicator of internal training dynamics, Mano exhibits lower gradient variance and a higher SNR than Muon, both under the same momentum coefficient $\mu = 0.95$.

normalization that increases the relative magnitude of rare directions with a monotone transformation of singular values in the momentum. While Muon performs whitening and flattens the spectrum, it discards the singular order information, which can be suboptimal from a theoretical perspective, as suggested by Su (2025).



(a) Attention Layer (b) MLP Layer

Figure 6. The spectral distributions of (a) all attention layers and (b) all MLP layers from an LLaMA-350M model at the 1000 step on the `C4/en` corpus, including the model gradient, momentum, and the update matrix of AdamW, Muon, and Mano. The manifold normalization of Mano may also be viewed as an efficient spectral regularization method that lifted the update spectra while preserving the singular values' original ordering.

Wall-clock Time Comparison. We have previously derived the theoretical FLOPs overhead of Mano, that without `MatMul` operations, the computational cost of the proposed manifold normalization operation is neglectable for LLM training. To assess the practical computational efficiency, we conduct a performance analysis of the normalization operations used in Mano and Muon, reporting their respective wall-clock times in Tab. 3. The observations suggest that the computational time of Mano grows linearly with the increase in the LLM's dimension, in contrast to the exponential growth observed for Muon. Fig. 1 compares the practical training performance measured in wall-clock time. In an experiment of one-day on the LLaMA-350M and -1B models, Mano achieves $1.75\times$ and $1.38\times$ faster convergence than Muon with continuously growing advantage.

5.4. Ablation Studies

In this subsection, we will present various ablation study results on the critical design choices of Mano and Manifold optimization methods, aiming to provide a complete understanding of how our strategy advanced.

Riemannian SGD-M. We first compare Mano and our reformed strategies to standard Riemannian SGD with momentum (RSGD-M) on the Oblique manifold. While the implementation of the two optimizers shares many similarities, their performance in training LLMs diverges significantly. As illustrated in Fig. 7, standard RSGD-M struggles to optimize the LLaMA-350M model, failing to reach the optimal loss range of 2.0 to 3.0 or show any sign of convergence. In contrast, Mano can significantly reduce the loss beyond RSGD-M. We posit that because traditional manifold methods rely on retractions to map the parameters onto smooth surfaces, they constrain LLMs' expressivity and hinder exploration of the loss landscape. By avoiding the assumption that the objective or solution must reside on a specific matrix manifold, Mano enables more flexible training dynamics necessary for LLMs.

Table 3. Computational cost comparison of Newton-Schulz iteration (NS, $T = 5$) and the manifold normalization enforced by Mano on Attention and MLP matrices from LLaMA-1B, -7B, and -70B models in `BFloat16`. Reported values denote the average over 1000 PyTorch runs, with peak GPU memory usage measured via `TORCH.CUDA`. Mano incurs significantly lower computational overhead than Muon, both theoretically with constant-time complexity and empirically in LLM experiments.

Module	Metric	Newton-Schulz	Mano
LLaMA-1B (dim=2048)			
Attention	Time	2.01 (ms)	0.14 (ms)
	Mem	64.1 (MB)	56.0 (MB)
MLP	Time	4.68 (ms)	0.17 (ms)
	Mem	119.3 (MB)	87.3 (MB)
LLaMA-7B (dim=4096)			
Attention	Time	14.83 (ms)	0.34 (ms)
	Mem	224.0 (MB)	192.0 (MB)
MLP	Time	30.22 (ms)	1.45 (ms)
	Mem	472.0 (MB)	344.0 (MB)
LLaMA-70B (dim=8192)			
Attention	Time	110.79 (ms)	2.19 (ms)
	Mem	896.0 (MB)	512.0 (MB)
MLP	Time	184.33 (ms)	4.35 (ms)
	Mem	1536.0 (MB)	1024.0 (MB)

Dynamic or Static Oblique Manifold? A key feature of Mano's implementation is the rotational Oblique manifold scheme. To understand how this feature functioned in optimization, we provide ablation experiment results in Tab. 4 with a static Oblique manifold fixed at the 0th dimension for parameters and update steps. We show that, under fixed column-wise normalization, Mano achieves comparable test perplexity on LLaMA-350M but performs significantly worse on LLaMA-1B, indicating a poor model-wise scaling behavior.

Table 4. Ablation results on LLaMA-350M and -1.3B models' trained on the `Pile` dataset, with test perplexity reported at the 10000 steps. While the static Oblique manifold hindered LLaMA-1.3B performance, momentum retraction yielded performance gains on LLaMA-350M, leaving space for further investigation into scale-dependent behavior.

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

Momentum with or without Retraction? Combining the traditional Manifold optimization strategies and our reformulated Mano optimizer, we examined whether the manifold constraints are extended from the update steps to the buffered momentum as well. By simply replacing v_t to M_t in Alg. 1, namely $M_t = v_t$, the momentum buffer is updated as the tangent momentum. Results in Tab. 4 suggest essentially identical results for both LLaMA-350M and -1B models; additional experiments are required to fully validate this design. Ultimately, the integration of manifold optimization into modern frameworks offers a vast design space. While this study cannot exhaustively explore every aspect, Mano introduces a compelling design philosophy with the potential to redefine optimization rules in high-dimensional regimes.

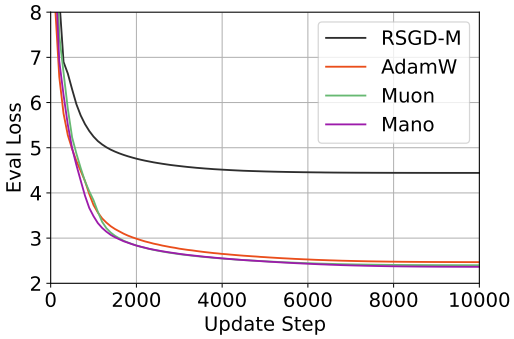


Figure 7. Comparing conventional Riemannian SGD-M and Mano on LLaMA-350M models trained on the `Pile` dataset. Unlike traditional manifold optimization methods that impose constraints on model parameters and expressivity during training, Mano provides a more flexible approach and superior performance.

6. Conclusion

Limitations. The empirical scope of this work is constrained by available computational resources. We aim to leave additional experiments, such as hyperparameter fine-tuning and over-training experiments of larger models, as future works. On the theoretical side, we frankly note that our current convergence analysis holds for a simplified version of the Mano optimizer, while recent LLM training methods often lacked convergence analysis. Extending the theory to cover momentum dynamics and broader optimization regimes remains an important future direction.

Summary. To the best of our knowledge, this is the first study to reformulate manifold optimization methods for efficient training LLMs. The proposed optimizer, Mano, departs from traditional manifold optimization techniques and modern optimizers that perform spectral preconditioning or second-moment estimates. Empirical results demonstrate that Mano outperforms the existing baseline of AdamW and Muon in training LLMs with significantly lower computational overhead than Muon and a reduced memory footprint compared to AdamW. Based on the hypothesis that mapping learning trajectories to smooth manifold surfaces can accelerate training convergence, this study highlights the potential of utilizing geometrically aware manifold techniques in conjunction with modern optimization strategies.

Acknowledgment

We gratefully thank Juanxi Tian for early discussions and hypotheses contributed to this work.

References

Absil, P.-A. and Gallivan, K. A. Joint diagonalization on the oblique manifold for independent component analysis. In *2006 IEEE international conference on acoustics speech and signal processing proceedings*, volume 5, pp. V–V. IEEE, 2006.

Absil, P.-A., Mahony, R., and Sepulchre, R. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2008.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pp. 1120–1128. PMLR, 2016.

Bécigneul, G. and Ganea, O.-E. Riemannian adaptive optimization methods. *arXiv preprint arXiv:1810.00760*, 2018.

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

Chaudhry, A., Khan, N., Dokania, P., and Torr, P. Continual learning in low-rank orthogonal subspaces. *Advances in Neural Information Processing Systems*, 33:9900–9911, 2020.

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., Lu, Y., et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

Chowdhury, M. N. U. R., Haque, A., and Soliman, H. The hidden cost of ai: Unraveling the power-hungry nature of large language models. 2025.

de Ocáriz Borde, H. S., Kazi, A., Barbero, F., and Lio, P. Latent graph inference using product manifolds. In *The eleventh international conference on learning representations*, 2023.

Douik, A. and Hassibi, B. Manifold optimization over the set of doubly stochastic matrices: A second-order geometry. *IEEE Transactions on Signal Processing*, 67 (22):5761–5774, 2019.

Fei, Y., Liu, Y., Jia, C., Li, Z., Wei, X., and Chen, M. A survey of geometric optimization for deep learning: from euclidean space to riemannian manifold. *ACM Computing Surveys*, 57(5):1–37, 2025.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gao, M., Hu, X., Yin, X., Ruan, J., Pu, X., and Wan, X. Llm-based nlg evaluation: Current status and challenges. *Computational Linguistics*, pp. 1–27, 2025.

Gupta, V., Koren, T., and Singer, Y. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Hu, J., Liu, X., Wen, Z.-W., and Yuan, Y.-X. A brief introduction to manifold optimization. *Journal of the Operations Research Society of China*, 8(2):199–248, 2020.

Huang, L., Liu, X., Lang, B., Yu, A., Wang, Y., and Li, B. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Huang, L., Liu, X., Qin, J., Zhu, F., Liu, L., and Shao, L. Projection based weight normalization: Efficient method for optimization on oblique manifold in dnns. *Pattern Recognition*, 105:107317, 2020.

Jiang, X., Wang, X., and Stich, S. U. Loram: Low-rank adaptation of large language models on manifold. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*.

Jing, L., Gulcehre, C., Peurifoy, J., Shen, Y., Tegmark, M., Soljacic, M., and Bengio, Y. Gated orthogonal recurrent units: On learning to forget. *Neural computation*, 31(4): 765–783, 2019.

Jordan, K., Jin, Y., Boza, V., Jiacheng, Y., Cesista, F., Newhouse, L., and Bernstein, J. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Kingma, D. P. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingswell, J., Whitstable, L., Troughton, O., Blumberg, K., and Sutherland, A. Sequential manifold regularization for large language model contextual stability. 2025.

Knight, P. A. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

Kumar, P. Large language models (llms): survey, technical frameworks, and future challenges. *Artificial Intelligence Review*, 57(10):260, 2024.

Kunstner, F., Milligan, A., Yadav, R., Schmidt, M., and Bietti, A. Heavy-tailed class imbalance and why adam outperforms gradient descent on language models. *Advances in Neural Information Processing Systems*, 37: 30106–30148, 2024.

Large, T., Liu, Y., Huh, M., Bahng, H., Isola, P., and Bernstein, J. Scalable optimization in the modular norm. *Advances in Neural Information Processing Systems*, 37: 73501–73548, 2024.

Liang, K., Chen, L., Liu, B., and Liu, Q. Cautious optimizers: Improving training with one line of code. *arXiv preprint arXiv:2411.16085*, 2024.

Liu, D., Qin, Z., Wang, H., Yang, Z., Wang, Z., Rong, F., Liu, Q., Hao, Y., Li, B., Chen, X., et al. Pruning via merging: Compressing llms via manifold alignment based layer merging. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17817–17829, 2024.

Liu, H., Li, Z., Hall, D., Liang, P., and Ma, T. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv:2305.14342*, 2023.

Liu, J., Su, J., Yao, X., Jiang, Z., Lai, G., Du, Y., Qin, Y., Xu, W., Lu, E., Yan, J., et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

Liu, W., Fu, S., Zhou, Y., Zha, Z.-J., and Nie, L. Human activity recognition by manifold regularization based dynamic graph convolutional networks. *Neurocomputing*, 444:217–225, 2021.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ma, C., Gong, W., Scetbon, M., and Meeds, E. Swan: Sgd with normalization and whitening enables stateless llm training. *arXiv preprint arXiv:2412.13148*, 2024.

Mo, Z., Huang, L.-K., and Pan, S. J. Parameter and memory efficient pretraining via low-rank riemannian optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Ozay, M. and Okatani, T. Optimization on submanifolds of convolution kernels in cnns. *arXiv preprint arXiv:1610.07008*, 2016.

Park, J., Kang, M., Lee, S., Lee, H., Kim, S., and Lee, J. Riemannian optimization for lora on the stiefel manifold. *arXiv preprint arXiv:2508.17901*, 2025.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.

Rajabi, S. and Rambhatla, S. Optimizing fine-tuning efficiency: Gradient subspace tracking on grassmann manifolds for large language models. In *NeurIPS 2024 Workshop on Mathematics of Modern Machine Learning*, 2024.

Rajabi, S., Nonta, N., and Rambhatla, S. Subtrack++: Gradient subspace tracking for scalable llm training. *arXiv preprint arXiv:2502.01586*, 2025.

Roberts, P. H. and Ursell, H. D. Random walk on a sphere and on a riemannian manifold. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 252(1012):317–356, 1960.

Roy, S. K., Mhammedi, Z., and Harandi, M. Geometry aware constrained optimization techniques for deep learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4460–4469, 2018.

Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D., and Gadepally, V. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–9. IEEE, 2023.

Schlotthauer, J., Kroos, C., Hinze, C., Hangya, V., Hahn, L., and Küch, F. Pre-training llms on a budget: A comparison of three optimizers. *arXiv preprint arXiv:2507.08472*, 2025.

Semenov, A., Pagliardini, M., and Jaggi, M. Benchmarking optimizers for large language model pretraining. *arXiv preprint arXiv:2509.01440*, 2025.

Shah, I., Polloreno, A. M., Stratos, K., Monk, P., Chaluvaraju, A., Hojel, A., Ma, A., Thomas, A., Tanwer, A., Shah, D. J., et al. Practical efficiency of muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

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

Team, K., Bai, Y., Bao, Y., Chen, G., Chen, J., Chen, N., Chen, R., Chen, Y., Chen, Y., Chen, Y., et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Vyas, N., Morwani, D., Zhao, R., Kwun, M., Shapira, I., Brandfonbrener, D., Janson, L., and Kakade, S. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

Wang, J., Chen, Y., Chakraborty, R., and Yu, S. X. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11505–11515, 2020.

Wang, J., Wang, M., Zhou, Z., Yan, J., Wu, L., et al. The sharpness disparity principle in transformers for accelerating language model pre-training. *arXiv preprint arXiv:2502.19002*, 2025.

Wen, K., Dang, X., Lyu, K., Ma, T., and Liang, P. Fantastic pretraining optimizers and where to find them ii: From weight decay to hyperball optimization, 11 2025a. URL https://whenwen.github.io/wd_blog/public/hyperball-part-1.html.

Wen, K., Hall, D., Ma, T., and Liang, P. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025b.

Wisdom, S., Powers, T., Hershey, J., Le Roux, J., and Atlas, L. Full-capacity unitary recurrent neural networks. *Advances in neural information processing systems*, 29, 2016.

Wren, A., Loxley, B., Cadwallader, H., Beckwith, S., Pargeter, F., and Blades, J. Contextual subspace manifold projection for structural refinement of large language model representations. *arXiv preprint arXiv:2502.08026*, 2025.

Xie, T., Luo, H., Tang, H., Hu, Y., Liu, J. K., Ren, Q., Wang, Y., Zhao, W. X., Yan, R., Su, B., et al. Controlled llm training on spectral sphere. *arXiv preprint arXiv:2601.08393*, 2026.

Yang, G., Simon, J. B., and Bernstein, J. A spectral condition for feature learning. *arXiv preprint arXiv:2310.17813*, 2023.

Yuan, H., Liu, Y., Wu, S., Zhou, X., and Gu, Q. Mars: Unleashing the power of variance reduction for training large models. *arXiv preprint arXiv:2411.10438*, 2024.

Zeng, A., Lv, X., Zheng, Q., Hou, Z., Chen, B., Xie, C., Wang, C., Yin, D., Zeng, H., Zhang, J., et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

Zhang, H., J Reddi, S., and Sra, S. Riemannian svrg: Fast stochastic optimization on riemannian manifolds. *Advances in Neural Information Processing Systems*, 29, 2016.

Zhang, T., Zheng, W., Cui, Z., and Li, C. Deep manifold-to-manifold transforming network. In *2018 25th IEEE international conference on image processing (ICIP)*, pp. 4098–4102. IEEE, 2018.

Zhang, Y. and Dong, Q. Multi-scale manifold alignment: A unified framework for enhanced explainability of large language models. *arXiv preprint arXiv:2505.20333*, 2025.

Zhang, Y., Chen, C., Li, Z., Ding, T., Wu, C., Kingma, D. P., Ye, Y., Luo, Z.-Q., and Sun, R. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*, 2024a.

Zhang, Y., Hu, J., Cui, J., Lin, L., Wen, Z., and Li, Q. Retraction-free optimization over the stiefel manifold with application to the lora fine-tuning. 2024b.

Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., and Tian, Y. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024a.

Zhao, R., Morwani, D., Brandfonbrener, D., Vyas, N., and Kakade, S. Deconstructing what makes a good optimizer for language models. *arXiv preprint arXiv:2407.07972*, 2024b.

Zhu, S., Pan, S., Zhou, C., Wu, J., Cao, Y., and Wang, B. Graph geometry interaction learning. *Advances in Neural Information Processing Systems*, 33:7548–7558, 2020.

A. Impact Statement

This paper aims to advance the understanding of deep learning optimization. While our findings may contribute to improving the sustainability of LLM training and, more broadly, societal welfare and environmental outcomes, we do not discuss specific social impact in this work.

B. Details for Reproducibility

B.1. Hyperparameters

In this paper, we follow the experimental setup described in Zhao et al. (2024a) and Raffel et al. (2020). The model architecture and respective hyperparameters are presented in Tab. 5. Besides the learning rate and batch size settings, we use a cosine decay learning rate scheduler with a minimum learning rate ratio of 0.1 for all experiments. Weight decay is set to 0.1, and gradients are clipped at 1.0. The LLaMA models are tokenized using the T5 tokenizer, and the Qwen3 models use the generative Qwen3 tokenizer. For optimizer hyperparameters, we use the default $(\beta_1, \beta_2) = (0.9, 0.95)$ for AdamW, number of Newton-Schulz iterations $T = 5$ for Muon, and the momentum coefficient $\mu = 0.95$ for both Muon and Mano. All training is performed in *BFloat16* mixed precision. All experiments are conducted with data distributed parallel (DDP) on $4\times$ NVIDIA H800-PCIe-80G GPUs, except for the LLaMA-130M experiments, which are performed on $4\times$ NVIDIA RTX-4090 GPUs.

Model	LLaMA-130M	LLaMA-350M	LLaMA-1.3B	Qwen-0.6B	Qwen-1.7B
Layer num	12	24	24	28	28
Hidden dim size	768	1024	2048	1024	2048
FFN dim size	1024	2736	5461	3072	6144
Attention heads	12	16	32	16	16
Seq-len	1024	1024	1024	1024	1024
Max Learning Rate	6.0×10^{-4}	3.0×10^{-4}	3.0×10^{-4}	3.0×10^{-4}	3.0×10^{-4}
Batch Size	16				
GradAcc	8				
Total Batch Size	512				
Iterations	10000 / 36000		10000		
Warmup iterations	1000				

Table 5. Training configurations for different LLaMA and Qwen model scales, including architecture details, sequence length, learning rate, batch size and gradient accumulation steps, and training schedule. For the over-trained LLaMA-130M and -350M models with 10B training corpus, the training iterations are extended to 36000 with all other hyperparameters unchanged.

B.2. Additional Empirical Designs for Mano

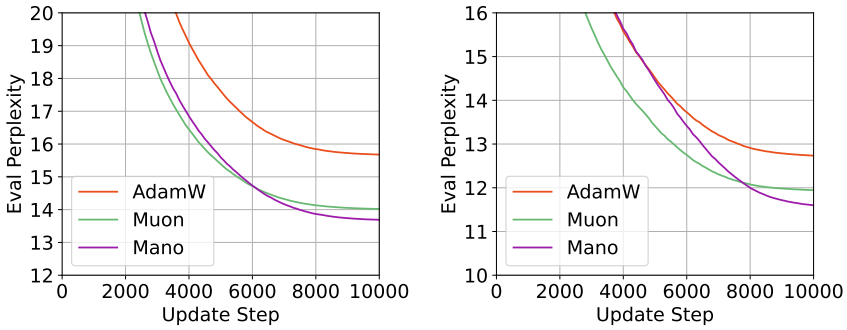
In this section, we report the complete hyperparameter configurations to facilitate reproducibility and further discuss the empirical designs for Mano.

Nesterov Accelerated Gradient. Empirical studies demonstrated that Nesterov-style momentum performs better than normal SGD-momentum for Muon, thus it has been made the default in the public implementation of Muon (Jordan et al., 2024; Liu et al., 2025). In our experiments of Mano, Nesterov Accelerated Gradient (NAG) may yield better performance than standard momentum for large-scale models, but can occasionally degrade it for smaller models, while having an overall minor effect on the ultimate training trajectory. For consistency, all experiment results in this paper use the standard momentum implementation and include NAG as an option in our implementation.

Input and Output Parameters. The update rule of Mano may apply to parameters of arbitrary dimensionality, for which the rotational manifold scheme can be implemented by iteratively traversing the Oblique manifold along each dimension. However, we followed the implementation of Muon to optimize the LLM's input and output parameters and $1 - D$ bias using AdamW Jordan et al. (2024). The modular norm theory stated that the optimization dynamics of the embedding layer should be different from other layers, which applies to the lm head layer as well, according to empirical studies of

Muon (Large et al., 2024). We hypothesize that the structural properties of the input embedding and output head layers in LLMs are constrained by the high sparsity of vocabulary activations, such that neither matrix orthogonalization nor manifold normalization outperforms AdamW adaptive learning per-parameter.

Learning Rate Independence. For all models, we used a uniform learning-rate schedule across experiments. Although different optimization algorithms may have different optimal learning rates, we controlled for this factor by constraining the RMS magnitude of parameter updates to AdamW's range of 0.2 to 0.4, as proposed by Liu et al. (2025) and discussed in the main paper. This normalization ensures that the optimizers operate at similar effective step sizes, allowing a fair comparison of their optimization behavior without optimizer-specific learning-rate tuning. Fig. 8 shows the training performance of Qwen3-0.6B models on the `Pile` dataset under a fixed learning-rate schedule with varying maximum learning-rate values. We find that Mano converges more slowly than Muon and AdamW during the early training phase, but achieves faster convergence in the later stage when using a higher learning rate. Despite the different learning-rate value, Mano achieves a noticeably lower final test perplexity than both baseline optimizers across the two experiments.



(a) Qwen3-0.6B / LR=3.0×10^{-4} (b) Qwen3-0.6B / LR=6.0×10^{-4}

Figure 8. We train Qwen3-0.6B models on the `Pile` dataset for 10000 steps using different baseline learning rates and the same learning-rate schedule. We observe that for a higher learning rate,

C. Mano for General Tensor

We provided the Mano optimizer in its general form for order-d tensor in Alg. 2. At step t, the tangent vector projection and manifold normalization are applied to the $t \bmod d$-th dimension of the parameters and the update step.

Algorithm 2 The Mano Optimizer for general tensor

Require: Weight $\theta_t \in \mathbb{R}^{n_1 \times \cdots \times n_k}$, momentum $M_t \in \mathbb{R}^{n_0 \times \cdots \times n_{d-1}}$, learning rate η_t at step t, momentum coefficient μ, and weight decay coefficient λ.

Initialize $M_0 \leftarrow \mathbf{0} \in \mathbb{R}^{n_0 \times \cdots \times n_{d-1}}, t \leftarrow 0$.
for each step **do**
  $g_t \leftarrow \nabla f(\theta_t)$
  $M_t \leftarrow \mu\, M_t + g_t$
  $k \leftarrow t \bmod k$
  $\hat{\theta}_t \leftarrow \theta_t \oslash \|\theta_t\|_{2,k}$
  $v_t \leftarrow M_t - \hat{\theta}_t \odot \langle M_t, \hat{\theta}_t \rangle_k$
  $\hat{v}_t \leftarrow v_t \oslash \|v_t\|_{2,k}$
  $\theta_{t+1} \leftarrow \theta_t - \eta_t(0.2\sqrt{n_k}\,\hat{v}_t + \lambda\theta_t)$
end for

D. Relationship to Existing Optimizers

Adafactor: By employing a factored rank-1 approximation of the second-order moments, Adafactor is closely related to Mano, which both explicitly normalize updates along parameter dimensions than globally or per-coordinate (Shazeer &

Stern, 2018). However, Adafactor relies on EMA-based second-moment normalization and controls per-parameter RMS in a manner similar to Adam, whereas Mano enforces manifold-based normalization and explicitly constrains the update steps to lie on an Oblique manifold. Consequently, Mano regulates update magnitudes geometrically rather than statistically, yielding stronger stability guarantees.

Spectral Optimizers: Mano fundamentally differs from existing spectral optimizers (e.g., Shampoo, SOAP, Muon, Conda) in that it does not rely on matrix-wide spectral information or second-order preconditions, but instead performs only vector-based operations to apply geometric constraints at each step. Although the Oblique manifold admits a spectral interpretation, we view Mano as a computationally efficient alternative to the current paradigm of extensive spectral preconditioning employed in optimization.

SSO and Hyperball: Two recent optimizers have integrated manifold constraints to accelerate LLM pretraining. The Hyperball optimizer (Wen et al., 2025a) employs manifold normalization to regulate the effective step sizes and weight norms, serving as an effective alternative to the weight decay scheme. Similarly, the Spectral Sphere Optimizer (SSO) (Xie et al., 2026) constrains both model weights and updates to a spectral sphere, aligning with maximal update parameterization (μP) (Yang et al., 2023). Our proposed methodology departs from both approaches: Mano applies manifold normalization to the momentum and requires no spectral preconditioning.

E. Proofs

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

$$\begin{cases} g_t \leftarrow \nabla f(\theta_t) \\ \hat{\theta}_t \leftarrow \theta_t \oslash \|\theta_t\|_{2,0} \\ v_t \leftarrow g_t - \hat{\theta}_t \odot \langle g_t, \hat{\theta}_t \rangle_0 \\ \hat{v}_t \leftarrow v_t \oslash \|v_t\|_{2,0} \\ \theta_{t+1} \leftarrow \theta_t - \eta\sqrt{m}\hat{v}_t \end{cases} \tag{7}$$

The mathematical operations involved are defined as follow:

- 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 $\langle Q, P\rangle_d^{(j)} = \langle Q^{(j)}, P^{(j)}\rangle$.
- Dimension-wise norm ($\|\cdot\|_{2,k}$): For the k-th dimension, $\|P\|_{2,d}^{(i)} = \|P^{(i)}\|_2$.

Before we present the proof of Theorem 1, we first propose and proof a Lemma on the lower bound on the inner product of the true gradient $g_t = \nabla f(\theta_t)$ and the normalized tangent $\hat{v}_t$.

Lemma 2. *Under the conditions of the update rule stated in Eq. 7, for $\phi_t^{(j)}$ be the angle between $g_t^{(j)} = \nabla f(\theta_t)^{(j)}$ and the parameter $\theta_t^{(j)}$, let $\sin(\phi_t^{(j)}) \geq \gamma > 0$ for tangential component γ, we have*

$$\langle \nabla f(\theta_t), \hat{v}_t \rangle \geq \gamma \|\nabla f(\theta_t)\|. \tag{8}$$

Proof. Denote the inner product as $S_t = \langle \nabla f(\theta_t), \hat{v}_t \rangle = \langle g_t, \hat{v}_t \rangle$, we have

$$S_t = \sum_{j=1}^{m} \langle g_t^{(j)}, \frac{v_t^{(j)}}{\|v_t^{(j)}\|} \rangle = \sum_{j=1}^{m} \langle v_t^{(j)} + \hat{\theta}_t^{(j)} \odot \langle g_t^{(j)}, \hat{\theta}_t^{(j)} \rangle, \frac{v_t^{(j)}}{\|v_t^{(j)}\|} \rangle \tag{9}$$

Because $v_t^{(j)}$ and $\hat{\theta}_t^{(j)}$ are orthogonal, the second part of the inner product is zero:

$$S_t = \sum_{j=1}^{m} \langle \frac{v_t^{(j)}, v_t^{(j)}}{\|v_t^{(j)}\|} \rangle = \sum_{j=1}^{m} \frac{\|v_t^{(j)}\|^2}{\|v_t^{(j)}\|} = \sum_{j=1}^{m} \|v_t^{(j)}\| \tag{10}$$

Because v_t is defined as the component of the gradient g_t orthogonal to the parameter θ_t, let $\phi_t^{(j)}$ be the angle between $g_t^{(j)}$ and the parameter $\theta_t^{(j)}$, we have $\|v_t^{(j)}\| = \|g_t^{(j)}\| \sin(\phi_t^{(j)})$. Thus, we can further express S_t as,

$$S_t = \sum_{j=1}^{m} \|g_t^{(j)}\| \sin(\phi_t^{(j)}) \tag{11}$$

We derive $S_t \to 0$ when the full gradient vanishes $\|g_t\| = 0$ or when the gradient is perfectly parallel to the weight vector $\sin(\phi_t^{(j)}) = 0$. If we assume the gradient is never perfectly aligned with the weights with a tangential component, we have $\sin(\phi_t^{(j)}) \geq \gamma > 0$, we have a lower bound for S_t:

$$S_t = \langle \nabla f(\theta_t), \hat{v}_t \rangle \geq \gamma \sum_{j=1}^{m} \|g_t^{(j)}\| = \gamma \|g_t\| = \gamma \|\nabla f(\theta_t)\| \tag{12}$$

The proof is now complete. □

E.1. Deterministic Setting

We first consider the convergence of the update rule stated in Eq. 7 under a deterministic setting for the true gradient $\nabla f(\theta_t) = g_t$. Assuming the function f is L_2-smooth (L-Lipschitz Continuity), i.e., for all x, y,

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2} \|y - x\|^2 \tag{13}$$

For $y = \theta_{t+1} = \theta_t - \eta\sqrt{m}\hat{v}_t$ applied to the L-smoothness:

$$\begin{aligned} f(\theta_{t+1}) &\leq f(\theta_t) + \nabla f(\theta_t)^\top (\theta_{t+1} - \theta_t) + \frac{L}{2} \|\theta_{t+1} - \theta_t\|^2 \\ &= f(\theta_t) + \langle \nabla f(\theta_t), (\theta_{t+1} - \theta_t) \rangle + \frac{L}{2} \|\eta\sqrt{m}\hat{v}_t\|^2 \\ &= f(\theta_t) - \eta\sqrt{m} \langle \nabla f(\theta_t), \hat{v}_t \rangle + \frac{L}{2} \eta^2 m^2 \end{aligned} \tag{14}$$

We now substitute $S_t = \langle \nabla f(\theta_t), \hat{v}_t \rangle$ to the deterministic analogue of the L-smoothness,

$$\begin{aligned} f(\theta_{t+1}) &\leq f(\theta_t) - \eta\sqrt{m} S_t + \frac{L}{2} \eta^2 m^2 \\ \eta\sqrt{m} S_t &\leq f(\theta_t) - f(\theta_{t+1}) + \frac{L}{2} \eta^2 m^2 \\ \sum_{t=0}^{T} \eta\sqrt{m} S_t &\leq \sum_{t=0}^{T} (f(\theta_t) - f(\theta_{t+1})) + \sum_{t=0}^{T} \frac{L}{2} \eta^2 m^2 \\ \sqrt{m}\eta \sum_{t=0}^{T} S_t &\leq f(\theta_0) - f(\theta_{T+1}) + (T+1) \frac{L}{2} m^2 \eta^2 \end{aligned} \tag{15}$$

Since $f(\theta_0) \geq f_{\inf}$, we have $\sqrt{m} \sum_{t=0}^{T} S_t \leq f(\theta_0) - f_{\inf} + (T+1)\frac{L}{2} m^2 \eta^2$. According to Lemma 2 that we have $S_t \geq \gamma \|g_t\|$ for tangential component γ, we arrive at

$$\begin{aligned} \sqrt{m}\eta \sum_{t=0}^{T} \gamma \|\nabla f(\theta_t)\| &\leq f(\theta_0) - f_{\inf} + (T+1) \frac{L}{2} m^2 \eta^2 \\ \sum_{t=0}^{T} \|\nabla f(\theta_t)\| &\leq \frac{f(\theta_0) - f_{\inf}}{m^{\frac{1}{2}} \gamma \eta} + (T+1) \frac{L m^{\frac{3}{2}} \eta}{2\gamma} \\ \frac{1}{T+1} \sum_{t=0}^{T} \|\nabla f(\theta_t)\| &\leq \frac{1}{T+1} \frac{f(\theta_0) - f_{\inf}}{m^{\frac{1}{2}} \gamma \eta} + \frac{L m^{\frac{3}{2}} \eta}{2\gamma} \end{aligned} \tag{16}$$

Let $\eta \leq \frac{C}{\sqrt{T+1}}$, we have

$$\begin{aligned}
&\frac{1}{T+1}\sum_{t=0}^{T}\|\nabla f(\theta_t)\| \leq \frac{1}{(T+1)\frac{C}{\sqrt{T+1}}}\left(\frac{f(\theta_0)-f_{\inf}}{m^{\frac{1}{2}}\gamma}\right)+\frac{C}{\sqrt{T+1}}\left(\frac{Lm^{\frac{3}{2}}}{2\gamma}\right)\\
&\Rightarrow \min_{t\in[0,T]}\|\nabla f(\theta_t)\| \leq \frac{1}{\sqrt{T+1}}\left(C_1+C_2\right),\ C_1=\frac{f(\theta_0)-f_{\inf}}{m^{\frac{1}{2}}\gamma C},\ C_2=\frac{Lm^{\frac{3}{2}}C}{2\gamma}
\end{aligned} \tag{17}$$

Thus, we derived the speed of convergence for Eq. 7 with deterministic gradient as $\min_{t\in[0,T]}\|\nabla f(\theta_t)\| \leq O(\frac{Lm^{\frac{3}{2}}}{\gamma\sqrt{T}})$. We now attempt to extend this result to stochastic gradient.

E.2. Stochastic Setting

We now attempt to extend the above proof to the stochastic setting, assuming that $\mathbb{E}[\xi_k]=0$ for gradient noise ξ of sub-sampling. This commonly used assumption is equivalent to the equality that the stochastic gradient $\tilde{g}_t=\nabla f(\theta_k,\xi_k)=\nabla f(\theta_k)+\xi_k$ is an unbiased estimator of the true gradient $g_t=\nabla f(x_k)$, as demonstrated by the following derivation:

$$\begin{aligned}
\mathbb{E}_{\xi_k}[\nabla f(x_k,\xi_k)] &= \mathbb{E}_{\xi_k}[\nabla f(x_k)+\xi_k]\\
\mathbb{E}_{\xi_k}[\nabla f(x_k,\xi_k)] &= \mathbb{E}_{\xi_k}[\nabla f(x_k)]+\mathbb{E}_{\xi_k}[\xi_k]\\
\mathbb{E}_{\xi_k}[\nabla f(x_k,\xi_k)] &= \nabla f(x_k)+0=\nabla f(x_k),
\end{aligned} \tag{18}$$

or equivalently $\mathbb{E}_{\xi_k}[\tilde{g}_k]=g_k$. We now extend Lemma 2 under the stochastic setting as detailed below.

Lemma 3. *Under the conditions of the update rule stated in Eq. 7, assume that $\mathbb{E}_{\xi_k}[\nabla f(x_k,\xi_k)]=\nabla f(x_k)$, for $\phi_t^{(j)}$ be the angle between $\tilde{g}_t$ and the parameter $\theta_t^{(j)}$, let $\sin(\phi_t^{(j)})\geq\gamma\geq 0$ for tangential component γ, we have*

$$\mathbb{E}_{\xi_t}[\langle\nabla f(\theta_t+\xi_t),\hat{v}_t\rangle] \geq \gamma\mathbb{E}_{\xi_t}[\|\nabla f(\theta_t)\|] \tag{19}$$

Proof. By the linearity of expectation, we have

$$\mathbb{E}_{\xi_t}[\langle\nabla f(\theta_t+\xi_t),\hat{v}_t\rangle] = \langle\mathbb{E}_{\xi_t}[\nabla f(\theta_t+\xi_t)],\hat{v}_t\rangle = \langle\nabla f(\theta_t),\hat{v}_t\rangle \tag{20}$$

According to Lemma 2, we have

$$\begin{aligned}
\langle\nabla f(\theta_t),\hat{v}_t\rangle &\geq \gamma\|\nabla f(\theta_t)\|\\
\mathbb{E}_{\xi_t}[\langle\nabla f(\theta_t),\hat{v}_t\rangle] &\geq \mathbb{E}_{\xi_t}[\gamma\|\nabla f(\theta_t)\|]\\
\mathbb{E}_{\xi_t}[\langle\nabla f(\theta_t),\hat{v}_t\rangle] &\geq \mathbb{E}_{\xi_t}[\gamma\|\nabla f(\theta_t)\|]\\
\mathbb{E}_{\xi_t}[\mathbb{E}_{\xi_t}[\langle\nabla f(\theta_t+\xi_t),\hat{v}_t\rangle]] &\geq \gamma\mathbb{E}_{\xi_t}[\|\nabla f(\theta_t)\|]\\
\mathbb{E}_{\xi_t}[\langle\nabla f(\theta_t+\xi_t),\hat{v}_t\rangle] &\geq \gamma\mathbb{E}_{\xi_t}[\|\nabla f(\theta_t)\|]
\end{aligned} \tag{21}$$

The proof is now complete. □

Proof. We now present the complete proof of Theorem 1, starting with the standard descent lemma for L-smoothness w.r.t. the true gradient g_t, and the stochastic inner product $\tilde{S}_t=\langle\nabla f(\theta_t+\xi_t),\hat{v}_t\rangle$

$$\begin{aligned}
f(\theta_{t+1}) &\leq f(\theta_t)-\eta\sqrt{m}\tilde{S}_t+\frac{L}{2}\eta^2m^2\\
\mathbb{E}_{\xi_t}[f(\theta_{t+1})] &\leq \mathbb{E}_{\xi_t}[f(\theta_t)]-\eta_t\sqrt{m}\,\mathbb{E}_{\xi_t}[\tilde{S}_t]+\frac{L}{2}\eta^2m^2\\
\sum_{t=0}^{T}\mathbb{E}_{\xi_t}[f(\theta_{t+1})] &\leq \sum_{t=0}^{T}\mathbb{E}_{\xi_t}[f(\theta_t)]-\sum_{t=0}^{T}\eta\sqrt{m}\,\mathbb{E}_{\xi_t}[\tilde{S}_t]+\sum_{t=0}^{T}\frac{L}{2}\eta^2m^2\\
\eta\sqrt{m}\sum_{t=0}^{T}\mathbb{E}_{\xi_t}[\tilde{S}_t] &\leq f(\theta_0)-f_{\inf}+(T+1)\frac{L}{2}\eta^2m^2
\end{aligned} \tag{22}$$

According to Lemma 3, we have

$$\begin{aligned}
\eta\sqrt{m}\sum_{t=0}^{T}\gamma\mathbb{E}_{\xi_t}[\|\nabla f(\theta_t)\|] &\leq f(\theta_0) - f_{\inf} + (T+1)\frac{L}{2}\eta^2 m^2 \\
\sum_{t=0}^{T}\mathbb{E}_{\xi_t}[\|\nabla f(\theta_t)\|] &\leq \frac{f(\theta_0) - f_{\inf}}{m^{\frac{1}{2}}\gamma\eta} + (T+1)\frac{Lm^{\frac{3}{2}}\eta}{2\gamma} \\
\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}_{\xi_t}[\|\nabla f(\theta_t)\|] &\leq \frac{1}{T+1}\frac{f(\theta_0) - f_{\inf}}{m^{\frac{1}{2}}\gamma\eta} + \frac{Lm^{\frac{3}{2}}\eta}{2\gamma}
\end{aligned} \tag{23}$$

Let $\eta \leq \frac{C}{\sqrt{T+1}}$, we have

$$\begin{aligned}
&\frac{1}{T+1}\sum_{t=0}^{T}\mathbb{E}_{\xi_t}[\|\nabla f(\theta_t)\|] \leq \frac{1}{(T+1)\frac{C}{\sqrt{T+1}}}\left(\frac{f(\theta_0) - f_{\inf}}{m^{\frac{1}{2}}\gamma}\right) + \frac{C}{\sqrt{T+1}}\left(\frac{Lm^{\frac{3}{2}}}{2\gamma}\right) \\
&\Rightarrow \min_{t\in[0,T]}\mathbb{E}_{\xi_t}[\|\nabla f(\theta_t)\|] \leq \frac{1}{\sqrt{T+1}}\left(C_1 + C_2\right),\ C_1 = \frac{f(\theta_0) - f_{\inf}}{m^{\frac{1}{2}}\gamma C},\ C_2 = \frac{Lm^{\frac{3}{2}}C}{2\gamma}
\end{aligned} \tag{24}$$

The proof is now complete. □