MONA: Muon Optimizer with Nesterov Acceleration for Scalable Language Model Training

Anonymous ACL submission

Abstract

The Muon optimizer has recently offered a promising alternative to AdamW for large language model training, leveraging matrix orthogonalization to produce geometry-aware updates. However, like all first-order methods, Muon can become trapped in sharp local minima. In this work, we present **MONA**, an optimizer that bridges Muon's orthogonalization framework with curvature-aware acceleration. MONA adds an acceleration term directly into Muon's gradient processing pipeline. This term is calculated from the exponential moving average of gradient differences. We provide a detailed convergence analysis for MONA, showing that the acceleration term enables escape from sharp minima while preserving Muon's spectral-norm regularization. Empirically, MONA achieves better convergence and downstream task performance compared to both Muon and AdamW across three scales of Mixture-of-Experts pretraining, spanning from 1B to 68B parameters, with the largest model trained on 1 trillion tokens. Furthermore, we conduct supervised fine-tuning on the MOE-68B-A3B model and evaluate it on general capability, mathematical reasoning, and code generation benchmarks, where MONA achieves **SOTA** performance.

1 Introduction

The choice of optimization algorithm(Robbins and Monro, 1951) is one of the most important decisions in training large language models (LLMs). For over a decade, Adam (Kingma and Ba, 2014) and AdamW (Loshchilov and Hutter, 2017) have been the standard choice. However, as model sizes scale to the hundreds of billions of parameters (Brown et al., 2020; Liu et al., 2024b; Team et al., 2025b; Yang et al., 2025), the requirement for optimizers with superior sample efficiency has increased.

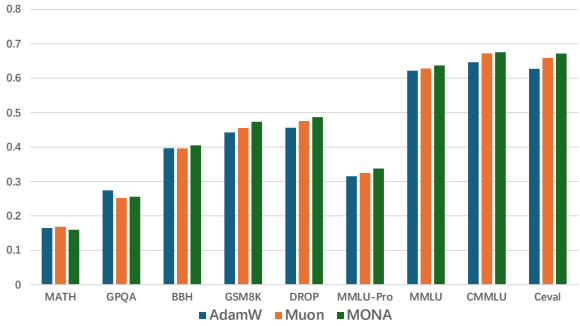

Figure 1: General capability evaluation results for pretraining MOE-68B-A3B at 700B tokens. MONA consistently outperforms Muon and AdamW across multiple benchmarks.

Recently, Muon (Jordan et al., 2024) has become a solid alternative. Instead of updating each parameter on its own, Muon views entire weight matrices as single geometric units. It then applies matrix orthogonalization to the momentum buffer. Muon is closely related to steepest descent under the spectral norm (Li and Hong, 2025). (Liu et al., 2025) demonstrated that Muon achieves approximately 2× computational efficiency compared to AdamW, training a 3B/16B-parameter MoE model on 5.7T tokens. Recent production deployments validate Muon's ability in large scale applications. Kimi K2 (Team et al., 2025a) and DeepSeek V4 (DeepSeek-AI, 2026) both used Muon to train trillion parameter MoE models.

Despite these successes, Muon, like all first-order gradient methods, lacks explicit mechanisms for exploring the loss landscape beyond local gradient information. Once the optimizer enters the basin of attraction, Muon's updates are guided solely by the momentum-averaged gradient, with no principled way to distinguish between flat and sharp minima (Keskar et al., 2016). This is particularly concerning for large-batch training (You et al., 2017, 2019).

(Zhao et al., 2026) proposed ALTO, an optimizer adaptor that introduces an acceleration term based on the exponential moving average of gradient differences. The key insight is that $g_k - g_{k-1}$ implicitly encodes curvature information via $g_k - g_{k-1} \approx H_k(\theta_k - \theta_{k-1})$. By adding this acceleration term to the gradient, ALTO enables escape from sharp minima and convergence to flatter solutions.

However, ALTO was designed as a general adaptor demonstrated on Adam and LAMB (You et al., 2019). Its integration with Muon presents challenges: Muon's orthogonalization is a highly nonlinear transformation, and it is unclear how curvature-aware acceleration interacts with this process. Moreover, ALTO's layer-wise learning rate regularization introduces complexity that may be unnecessary with Muon's inherently well-conditioned updates.

We present **MONA**, which seamlessly integrates the acceleration principle into Muon's orthogonalization framework. MONA applies the acceleration term before orthogonalization, transforming the raw gradient into a curvature-aware direction that is processed through Muon's standard pipeline.

Our contributions are threefold.

1. **Algorithm Design.** MONA combines Muon's matrix orthogonalization with an acceleration term derived from gradient differences, preserving Muon's geometric structure while adding curvature awareness to the updates.

2. **Convergence Analysis.** We establish convergence rates for MONA in non-convex settings and describe the acceleration term's effect on sharp minimum escape.

3. **Empirical Validation.** MONA outperforms both Muon and AdamW across three MoE model scales in pretraining, namely MOE-1B-A0d2B, MOE-6B-A0d5B, and MOE-68B-A3B, with the largest being trained on 1T tokens. We further conduct supervised fine-tuning on the MOE-68B-A3B model and evaluate it on general capability, mathematical reasoning, and code generation benchmarks, where MONA consistently achieves superior performance.

2 Related Work

2.1 Matrix-Aware Optimizers

The standard approach handles parameters as independent scalars, updating them based on their own gradient history. Second-order methods such as K-FAC (Martens and Grosse, 2015) and Shampoo (Gupta et al., 2018) use Kronecker-factored preconditioners, but their $O(N^{3/2})$ complexity limits scalability.

Muon (Jordan et al., 2024) occupies a unique position: by applying polar decomposition to the momentum matrix, it achieves $O(N)$ complexity while respecting matrix geometry. The updates are closely related to steepest descent under the spectral norm (Li and Hong, 2025).

2.2 Muon Variants

Scaling up Muon was explored by (Liu et al., 2025), who identified weight decay and per-parameter update scaling ($\gamma = 0.2\sqrt{\max(m,n)}$) as crucial for scaling Muon to billion-parameter models, along with distributed ZeRO-1-style optimization. Adaptive variants were also investigated. AdaMuon (Si et al., 2025) addresses Muon's lack of element-wise adaptivity by incorporating second-momentum estimation in the orthogonalized direction, achieving up to 40% efficiency gains over AdamW. ROOT (He et al., 2025) proposes adaptive Newton iterations with size-specific coefficients for consistent precision across layers, and proximal optimization with soft thresholding to suppress outlier gradients in the momentum buffer before orthogonalization.

Efficiency improvements have been proposed as well. DropMuon (Gruntkowska et al., 2025) introduces a randomized progressive training framework that updates only a subset of layers per step according to a randomized schedule, combining progressive training efficiency with layer-specific non-Euclidean updates. Dion (Ahn et al., 2025) replaces Newton-Schulz iteration with amortized power iteration on a momentum buffer, avoiding full-matrix reconstruction and integrating cleanly with weight sharding. MuonBP (Khaled et al., 2025) combines local block orthogonalization with global orthogonalization to eliminate the communication bottleneck under model parallelism while maintaining Muon's data efficiency.

2.3 Acceleration in Optimization

Adan (Xie et al., 2024) develops a Nesterov momentum estimation method that estimates the gradient's first- and second-order moments for convergence acceleration in adaptive algorithms. Lion (Chen et al., 2023) is a memory-efficient optimizer discovered through symbolic program search that uses only momentum tracking with sign-based updates at a uniform magnitude across all parameters.

ALTO (Zhao et al., 2026) is the closest precursor. It introduces $a_k = \beta_a a_{k-1} + (1 - \beta_a)(g_k - g_{k-1})$ and replaces g_k with $g_k + \alpha a_k$. The insight is that $-\nabla\|\nabla f(\theta_k)\|^2 \approx H_k(\theta_k - \theta_{k-1}) \approx \bar{g}_k - \bar{g}_{k-1}$, pointing away from sharp minima.

3 Preliminaries

3.1 Problem Setting

Consider the stochastic non-convex problem

$$\min_{\theta \in \mathcal{D}} f(\theta), \quad f(\theta) = \mathbb{E}_\zeta[\ell(\theta, \zeta)], \tag{1}$$

where $\ell(\theta, \zeta)$ is the loss on sample ζ drawn from distribution $\mathcal{P}$. The parameter $\theta \in \mathbb{R}^d$ partitions into matrix-valued parameters $\{W^{(i)} \in \mathbb{R}^{m_i \times n_i}\}$ and vector-valued parameters. We focus on a single matrix $W \in \mathbb{R}^{m \times n}$.

At iteration k, we observe stochastic gradient $G_k = \nabla_W \ell(W_k, \zeta_k)$. The goal is to iteratively update W_k toward a minimizer W^*.

3.2 Muon Optimizer

Muon (Jordan et al., 2024) updates matrix parameters as:

$$M_k = \mu M_{k-1} + G_k, \tag{2}$$

$$O_k = \text{Newton-Schulz}(M_k), \tag{3}$$

$$W_{k+1} = W_k - \eta\left(\gamma O_k + \lambda W_k\right). \tag{4}$$

The Newton-Schulz iteration approximates the polar decomposition. Starting from $X_0 = M_k / \|M_k\|_F$,

$$X_{t+1} = aX_t + bX_t X_t^\top X_t + cX_t X_t^\top X_t X_t^\top X_t, \tag{5}$$

where $a = 3.4445$, $b = -4.7750$, $c = 2.0315$ ensure convergence for singular values in $[0, 1]$. After T iterations (typically $T = 5$), $X_T \approx UV^\top$ where $M_k = U\Sigma V^\top$ is the SVD.

Following (Liu et al., 2025), the scaling factor matches Muon's RMS to AdamW:

$$\gamma = 0.2 \cdot \sqrt{\max(m, n)}. \tag{6}$$

3.3 Acceleration

Acceleration computes:

$$d_k = g_k - g_{k-1}, \tag{7}$$

$$a_k = \beta_a a_{k-1} + (1 - \beta_a) d_k, \tag{8}$$

$$\tilde{g}_k = g_k + \alpha a_k, \tag{9}$$

where β_a controls acceleration memory and α is the acceleration coefficient.

The theoretical motivation is that

$$-\nabla\|\nabla f(\theta_k)\|^2 = -2H_k \bar{g}_k \approx \bar{g}_k - \bar{g}_{k-1}, \tag{10}$$

where H_k is the Hessian and $\bar{g}_k$ the full-batch gradient. The direction $-\nabla\|\nabla f(\theta_k)\|^2$ points away from sharp minima. ALTO uses gradient differences as a computationally efficient proxy.

4 Methodology: MONA

4.1 Algorithm

Pseudocode is in Algorithm 1. For vector-valued parameters (embeddings, biases, etc.), MONA falls back to AdamW, following Muon's convention. Applying the acceleration term before momentum accumulation ensures that the momentum buffer captures curvature-aware directions for orthogonalization. Applying it after orthogonalization would destroy the orthogonal structure that Muon relies on.

Algorithm 1 MONA Optimizer

Require: η, μ, (β_a, α), λ, NS steps T, scaling γ
$M_0 \leftarrow 0$, $A_0 \leftarrow 0$, $G_0 \leftarrow 0$
for $k = 1, 2, 3, \ldots$ **do**
  $G_k \leftarrow \nabla_W \ell(W_k, \zeta_k)$
  $D_k \leftarrow G_k - G_{k-1}$
  $A_k \leftarrow \beta_a A_{k-1} + (1 - \beta_a) D_k$
  $\tilde{G}_k \leftarrow G_k + \alpha A_k$
  $M_k \leftarrow \mu M_{k-1} + \tilde{G}_k$
  $O_k \leftarrow \text{Newton-Schulz}(M_k, T)$
  $W_{k+1} \leftarrow W_k - \eta(\gamma O_k + \lambda W_k)$
  $G_{k-1} \leftarrow G_k$
end for

4.2 Geometric Intuition

MONA's effectiveness stems from the interplay of two mechanisms.

Spectral normalization (Muon). Newton-Schulz ensures O_k has singular values

close to 1, preventing over-commitment to large-gradient directions. Muon performs steepest descent on the spectral-norm unit ball.

Curvature-aware acceleration. The term A_k encodes how the gradient changes. Near sharp minima, $\|D_k\|$ is large, pushing toward flatter regions. In flat regions, A_k is small, allowing stable convergence.

The combination works well together. Orthogonalization ensures geometrically well-conditioned updates, while acceleration enriches the input with curvature information for more informed direction selection.

5 Theoretical Analysis

We provide convergence analysis for MONA under standard assumptions. Detailed proofs are deferred to Appendix A.

5.1 Assumptions

Assumption 1 (L-smoothness). *The loss $\ell(W, \zeta)$ is L-smooth. For all W, W' and ζ,*

$$\|\nabla_W \ell(W, \zeta) - \nabla_W \ell(W', \zeta)\|_F \leq L\|W - W'\|_F. \quad (11)$$

Assumption 2 (Unbiased gradient with bounded variance). *The stochastic gradient satisfies $\mathbb{E}[G_k \mid W_k] = \nabla f(W_k)$ and $\mathbb{E}[\|G_k - \nabla f(W_k)\|_F^2] \leq \sigma^2$.*

Assumption 3 (Bounded gradient). *There exists $G > 0$ such that $\|G_k\|_F \leq G$ a.s.*

Assumption 4 (Expected directional alignment). *There exists $\rho > 0$ such that $\mathbb{E}[\langle \nabla f(W_k), O_k \rangle \mid W_k] \geq \rho \|\nabla f(W_k)\|_F^2$.*

Assumption 4 requires that the update direction O_k has a positive expected correlation with the full gradient. This holds because Newton-Schulz preserves the column space of M_k, and the stochastic gradient (in expectation) lies within this space.

5.2 Key Lemmas

Lemma 1 (Boundedness of acceleration). *Under Assumptions 2–3:*

$$\|A_k\|_F \leq 2G, \quad (12)$$

$$\|\tilde{G}_k\|_F \leq G(1 + 2|\alpha|). \quad (13)$$

Lemma 2 (Momentum bound). *Under the same assumptions,*

$$\|M_k\|_F \leq \frac{G(1 + 2|\alpha|)}{1 - \mu}. \quad (14)$$

Remark 1 (Newton-Schulz iteration). *Newton-Schulz iteration converges to the polar decomposition $UV^\top$ exponentially fast in the number of iterations T. After T steps, the approximation error satisfies $\|\tilde{O} - UV^\top\|_F \leq \epsilon_{NS}\sqrt{r}$ where $r = \mathrm{rank}(M)$ and ϵ_{NS} decays exponentially with T. This is a standard result in numerical analysis, and with the standard coefficients and $T = 5$, the error is negligible for practical purposes.*

5.3 Main Convergence Result

Theorem 1 (Non-convex convergence of MONA). *Let Assumptions 1–4 hold. Run MONA with learning rate $\eta > 0$, momentum $\mu \in [0, 1)$, acceleration (β_a, α) with $|\alpha| < 1/(1 - \beta_a)$. Define:*

$$\bar{G} = G(1 + 2|\alpha|), \quad C_1 = \frac{\bar{G}}{1-\mu} \quad (15)$$

$$C_2 = L\gamma^2 C_m/2, \quad C_3 = \rho\gamma. \quad (16)$$

If $\eta \leq \min\{1/L, C_3/C_2\}$, then after K iterations,

$$\frac{1}{K}\sum_{k=0}^{K-1} \mathbb{E}\left[\|\nabla f(W_k)\|_F^2\right] \leq \frac{f(W_0) - f^*}{\eta C_3 K} + \frac{\eta L C_4}{C_3}, \quad (17)$$

where $C_4 = \gamma^2 C_m/2$ for $C_m = O(r)$ with $r = \mathrm{rank}(M_k)$.

Proof Sketch. By L-smoothness,

$$f(W_{k+1}) \leq f(W_k) - \eta\gamma \langle \nabla f(W_k), O_k \rangle + \frac{\eta^2 L \gamma^2}{2}\|O_k\|_F^2. \quad (18)$$

By Assumption 4, $\mathbb{E}[\langle \nabla f(W_k), O_k \rangle \mid W_k] \geq \rho\|\nabla f(W_k)\|_F^2$. Since Newton-Schulz outputs approximately orthogonal matrices, $\|O_k\|_F^2 = O(r)$ where $r = \mathrm{rank}(M_k)$. Taking expectations and telescoping over $k = 0, \ldots, K - 1$ yields the claimed first-moment bound. The condition on α ensures stability. □

Remark 2 (Rate interpretation). *With $\eta = O(K^{-1/2})$, the bound becomes $O(K^{-1/2})$, matching SGD and AdamW. The acceleration affects constants, not the asymptotic rate.*

5.4 Acceleration and Sharp Minimum Escape

Near a stationary point W^*, with Hessian H^*,

$$f(W) \approx f(W^*) + \frac{1}{2}\langle W - W^*, H^*(W - W^*)\rangle, \quad (19)$$

and $\mathbb{E}[G_k - G_{k-1}] \approx H^*(W_k - W_{k-1}) = -\eta\gamma H^* O_{k-1}$.

Proposition 1 (Sharp minimum escape)**.** *Near a sharp minimum with* $\lambda_{\max}(H^*) \gg \lambda_{\min}(H^*) > 0$,

$$\mathbb{E}[A_k] \approx -\eta\gamma H^* \sum_{j=0}^{k} (1-\beta_a)\beta_a^{k-j} O_{j-1}. \quad (20)$$

For large eigenvalues (sharp directions), the acceleration is large, promoting escape. For small eigenvalues (flat directions), it is small, permitting convergence.

This formalizes that MONA selectively resists sharp minima. The selectivity arises from Hessian-dependent scaling—sharp directions amplify the acceleration; flat directions suppress it.

5.5 Comparison with Baseline Muon

In Muon, $O_k^{\text{Muon}} = \text{NS}(\mu M_{k-1} + G_k)$. In MONA, $O_k^{\text{MONA}} = \text{NS}(\mu M_{k-1} + G_k + \alpha A_k)$. The difference αA_k is approximately proportional to the negative Hessian-weighted average of past directions, biasing orthogonalization toward lower-curvature directions.

6 Experiments

6.1 Pretraining

We pretrain three MoE (Jacobs et al., 1991) language models of increasing scale based on the LongCat architecture (Team et al., 2025b), specifically ScMoE (Cai et al., 2024) with MLA (Liu et al., 2024a; Vaswani et al., 2017), comparing MONA against Muon. All models employ MLA and train on sequences of length 8192. We monitor validation loss on four held-out domains covering code, mathematical reasoning, general English text, and Chinese academic text to assess convergence behavior across diverse capabilities. For all the experimental curves shown in the figures, except for the optimizer selection, we kept all hyperparameters, including learning rate, learning rate scheduling, and batch size, consistent throughout the experiments.

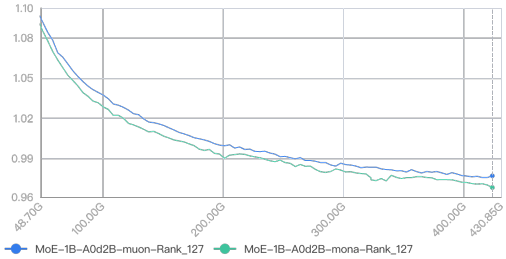

Figure 2: Validation loss on Code-Valid for MOE-1B-A0d2B.

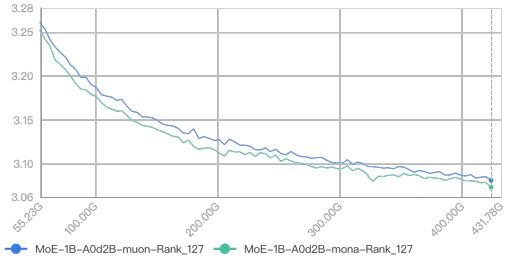

Figure 3: Validation loss on General-English-Text for MOE-1B-A0d2B.

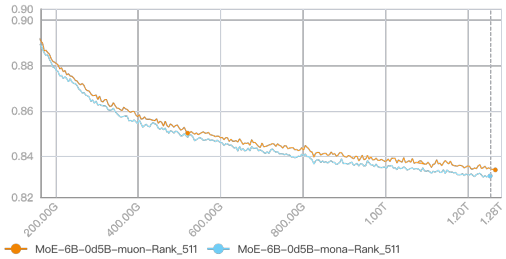

Figure 4: Validation loss on Code-Valid for MOE-6B-A0d5B.

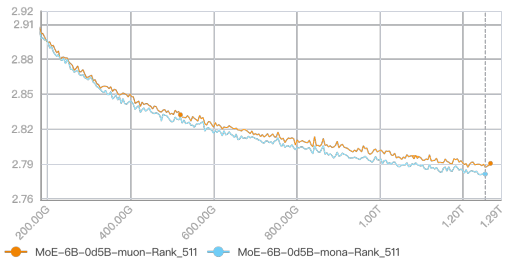

Figure 5: Validation loss on General-English-Text for MOE-6B-A0d5B.

MOE-1B-A0d2B. The smallest model uses 10 transformer layers with 768 hidden dimensions, 16 attention heads, 128 experts with 256

Table 1: General capability evaluation results for MOE-68B-A3B at 700B tokens. Scores are reported as mean ± std.

Benchmark	**AdamW**	**Muon**	**MONA**
MMLU-FewShot (Hendrycks et al., 2020)	0.6218±0.0098	0.6281±0.0098	**0.6373**±0.0097
MMLU-Pro-FewShot (Wang et al., 2024)	0.3150±0.0169	0.3250±0.0170	**0.3375**±0.0172
CMMLU-FewShot (Li et al.)	0.6464±0.0090	0.6723±0.0088	**0.6756**±0.0088
CEVAL-FewShot (Huang et al., 2023)	0.6274±0.0263	0.6586±0.0255	**0.6717**±0.0251
BBH-FewShot (Suzgun et al., 2023)	0.3966±0.0125	0.3962±0.0126	**0.4049**±0.0126
Math (CoT) (Hendrycks et al., 2021)	0.1648±0.0103	**0.1684**±0.0104	0.1594±0.0101
GPQA-FewShot (Rein et al., 2023)	**0.2743**±0.0365	0.2518±0.0355	0.2555±0.0356
DROP (CoT) (Dua et al., 2019)	0.4560±0.0100	0.4757±0.0100	**0.4868**±0.0100
GSM8K (CoT) (Cobbe et al., 2021)	0.4426±0.0270	0.4550±0.0271	**0.4734**±0.0272
Average	0.4382	0.4478	**0.4557**

Table 2: Code generation and mathematical reasoning evaluation results for MOE-68B-A3B at 700B tokens. Scores are reported as mean ± std. Knowledge-Specific is the average of hellaswag (Zellers et al., 2019), commonsenseqa (Talmor et al., 2019), openbookqa (Mihaylov et al., 2018), piqa (Bisk et al., 2020), siqa (Sap et al., 2019), and winogrande (Sakaguchi et al., 2019).

Benchmark	**AdamW**	**Muon**	**MONA**
HumanEval+ (Liu et al., 2024c)	**0.2927**±0.0699	0.2866±0.0694	0.2805±0.0690
Multiple (Cassano et al.)	0.2519±0.0326	0.2239±0.0313	**0.2848**±0.0343
MBPP+ (Liu et al., 2024c) 3-shot	**0.5265**±0.0504	0.5053±0.0505	0.5265±0.0504
BigCodeBench (Allal et al., 2022)	0.3123±0.0269	0.3061±0.0268	**0.3377**±0.0275
LiveCodeBench (Jain et al., 2025)	0.0358±0.0219	0.0430±0.0238	**0.0502**±0.0257
LeetCode (Xia et al., 2025)	0.0540±0.0209	**0.0847**±0.0249	0.0709±0.0232
Bigmath Mathematics (Albalak et al., 2025)	0.4627±0.0173	0.4493±0.0182	**0.4733**±0.0173
ProofWriter (Tafjord et al., 2021)	0.4995±0.0143	**0.5312**±0.0225	0.5169±0.0142
Knowledge-Specific	0.6575±0.0142	0.7021±0.0228	**0.7025**±0.0139
CRUXEval (Gu et al., 2024)	**0.3563**±0.0235	0.3362±0.0232	0.3500±0.0234
FullStackBench (Cheng et al., 2024)	0.2567±0.0145	**0.3044**±0.0153	0.2975±0.0153
HumanEval+ 3-shot	**0.3110**±0.0711	0.3049±0.0707	0.3049±0.0707
Average	0.3353	0.3397	**0.3495**

FFN hidden size each, and top-8 routing. We train for approximately 400B tokens. As shown in Figure 2 and Figure 3, MONA consistently achieves lower validation loss than Muon across both validation domains, with additional results on mathematical reasoning and Chinese academic text provided in Appendix B.

MOE-6B-A0d5B. The medium-scale model also uses 10 transformer layers but increases to 1536 hidden dimensions, 128 experts of 1024 FFN hidden size each, and top-6 routing, training for approximately 1.2T tokens. Figure 4 and Figure 5 show that MONA maintains its advantage at this scale, again outperforming Muon across both validation domains (additional results are provided in Appendix B). The more volatile curves compared to the MoE-1B-A0d2B model indicate a more complex optimization landscape.

MOE-68B-A3B. The largest model scales to 14 transformer layers with 3072 hidden dimensions, 32 attention heads, 256 experts with 1024 FFN hidden size each, and top-12 routing (Liu et al., 2026). We train for approximately 700B tokens and evaluate the intermediate checkpoint on a comprehensive suite of benchmarks covering general capability, mathematical reasoning, and code generation, comparing MONA against both Muon and AdamW.

Table 1 reports the results on general capability benchmarks. MONA achieves the highest average score (0.4557) across all three optimizers, outperforming Muon (0.4478) and AdamW (0.4382). Notably, MONA shows consistent improvements on MMLU-FewShot, MMLU-Pro, CMMLU-FewShot, CEVAL-FewShot, BBH-FewShot, DROP, and GSM8K, demonstrating that curvature-aware acceleration helps the model develop stronger general reasoning and mathematical capabilities.

Table 3: BigCode evaluation results for MOE-68B-A3B. Scores are reported as mean ± std. ckpt_−1 is the checkpoint before SFT.

Task	**ckpt_−1**	**Muon**	**MONA**
MULTIPLE	0.6082±0.0366	0.7836±0.0302	**0.7851**±0.0298
MBPP+	0.5423±0.0503	0.7778±0.0420	**0.7857**±0.0414
HUMANEVAL+	0.6829±0.0714	0.8720±0.0513	**0.8841**±0.0491
TORCHDATAEVAL (Zan et al., 2022)	0.7740±0.0328	**0.8109**±0.0308	0.8077±0.0309
DS1000 (Lai et al., 2023)	0.2860±0.0332	0.3395±0.0350	**0.3703**±0.0359
BIGCODEBENCH	0.0658±0.0144	0.5526±0.0289	**0.5553**±0.0289
HUMANEVAL+ 3-shot	0.0061±0.0120	0.8598±0.0533	**0.8659**±0.0523

Table 2 reports the results on code generation and specialized mathematics benchmarks. MONA again achieves the highest average, compared to Muon and AdamW. MONA delivers particularly strong gains on MULTIPLE, BIGCODEBENCH, and LIVECODEBENCH, indicating that the acceleration term's exploration of flatter minima enables the model to learn more transferable code representations.

6.2 Supervised Fine-Tuning and Evaluation

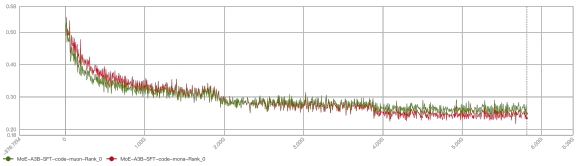

Figure 6: SFT training loss on code data for the MOE-68B-A3B model. MONA-pretrained checkpoint (red) achieves lower loss than Muon (green) throughout training, with a larger gap emerging in later epochs.

To assess the practical utility of MONA-optimized models beyond pretraining, we conduct supervised fine-tuning (SFT) on the MOE-68B-A3B model using high-quality code data. The SFT stage employs Adam with a peak learning rate of cosine decay, training for 3 epochs on approximately 2B tokens with a maximum sequence length of 32k. We compare the MONA-pretrained and Muon-pretrained checkpoints under identical SFT settings.

Figure 6 shows the SFT training loss curves. The MONA-pretrained model consistently achieves lower training loss than the Muon-pretrained baseline throughout all three epochs, with the gap widening in the later stages of training. This suggests that the curvature-aware acceleration in MONA not only improves pretraining convergence but also produces initializations that are better suited for downstream adaptation.

Table 3 reports the results on the BigCode evaluation suite (Zhuo et al., 2025). The ckpt_−1 column shows that the pretrained base model performs poorly on most code generation tasks, confirming that the subsequent SFT stage is essential for unlocking programming capabilities. After SFT, both Muon and MONA pretrained models achieve large improvements across all benchmarks. Comparing the two, MONA achieves higher scores than Muon on 6 out of 7 tasks. Notably, MONA delivers substantial gains on MBPP+, HUMANEVAL+, and DS1000, etc., indicating that the curvature-aware acceleration during pretraining enables the model to learn more robust code representations that transfer better to downstream programming tasks.

6.3 Efficient Deployment with Reduced Overhead

MONA introduces two additional state buffers compared to Muon: the previous gradient G_{k-1} and the acceleration buffer A_k, both stored in full precision during training. While this overhead is manageable on large-scale training clusters, it can be prohibitive for researchers and practitioners with limited GPU memory. To address this, we explore two complementary strategies for reducing the memory footprint of the acceleration term.

The first strategy is low-precision quantization. We implement **MONA-Lite**, a variant that stores G_{k-1} and A_k in bfloat16 (BF16) rather than float32 (FP32). This reduces the memory overhead of these auxiliary states by approximately 50% without requiring any changes to the update equations or the orthogonalization pipeline.

The second strategy is streaming gradient

computation, an engineering optimization that eliminates the G_{k-1} buffer entirely. Instead of storing the previous gradient separately, we compute the gradient difference in-place. After backpropagation produces G_k, we immediately compute $G_k - G_{\text{slot}}$ against the gradient stored from the previous step, update A_k with this difference, and then overwrite the slot with G_k. This streaming approach removes the need to maintain a dedicated G_{k-1} buffer, leaving only A_k as auxiliary state.

When combined, BF16 quantization and streaming computation reduce the total extra memory overhead by about 75% compared to standard FP32 MONA. Streaming cuts it by 50%, and quantization cuts it by another 50%. This makes the accelerated optimizer practical for resource-constrained settings without sacrificing its curvature-aware benefits.

We evaluate both strategies by pretraining the MOE-1B-A0d2B model under identical hyperparameters. Figure 7 builds on the code validation plot in Figure 2, adding the MONA-Lite curve (yellow) alongside Muon (blue) and FP32 MONA (green). Similarly, Figure 8 extends Figure 3 to the general English text domain. In both cases, MONA-Lite closely tracks the FP32 MONA curve throughout training, while both optimizers maintain a clear advantage over Muon. This demonstrates that the acceleration term can be safely compressed to BF16 and that streaming computation does not affect training quality across diverse evaluation domains.

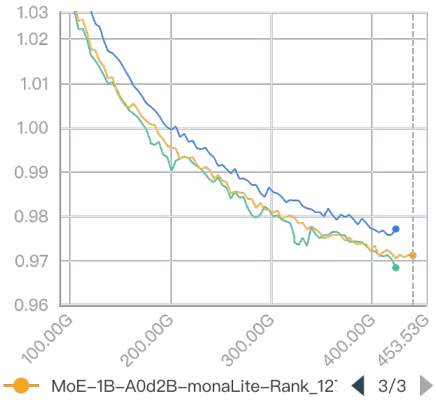

Figure 7: Validation loss on Code_Valid for the MOE-1B-A0d2B model, extending Figure 2 with the MONA-Lite curve (yellow). MONA-Lite closely tracks FP32 MONA while both maintain a clear advantage over Muon.

Figure 8: Validation loss on General-English-Text for the MOE-1B-A0d2B model, extending Figure 3 with the MONA-Lite curve (yellow). As in the code domain, MONA-Lite remains indistinguishable from FP32 MONA while outperforming Muon.

We also measured the training speed overhead. Appendix C shows MONA is about 1% slower inside the optimizer step, but this difference disappears at the iteration level, so overall training time is essentially the same as Muon.

7 Conclusion

We propose MONA, an improved variant of the Muon optimizer, which integrates curvature-aware acceleration into its matrix orthogonalization framework. By augmenting gradients with an exponential moving average of gradient differences before orthogonalization, MONA endows Muon with the ability to escape sharp minima while preserving all of Muon's geometric benefits. Our theoretical analysis proves convergence under standard assumptions and shows how the acceleration term avoids sharp minima.

Empirically, MONA achieves lower validation loss than both Muon and AdamW across three scales of MoE models (1B to 68B parameters) on code, mathematical reasoning, and general text. At the 68B scale, MONA delivers superior general capability and code generation scores, and its pretrained models achieve higher BigCode evaluation results after code-specific SFT. To reduce the overhead of the acceleration term, we further introduce MONA-Lite, which combines BF16 quantization with streaming gradient computation to cut the extra memory by approximately 75% without sacrificing training quality.

8 Limitations

Hyperparameter tuning. MONA introduces two additional hyperparameters (β_a, α) compared to Muon, which adds a tuning cost. However, we observe a consistent relationship between them across all experiments: $\alpha = -1/(2(1-\beta_a))$. In practice, we set $\beta_a = 0.99$ for the MOE-1B-A0d2B model, $\beta_a = 0.98$ for the MOE-6B-A0d5B model, and $\beta_a = 0.975$ for the MOE-68B-A3B model. This shared rule largely alleviates the tuning overhead.

Memory overhead. Even with BF16 quantization and streaming gradient computation (MONA-Lite), the acceleration buffer A_k still introduces approximately half a gradient's worth of extra memory. Users should balance this residual overhead against the observed training gains when deploying MONA in memory-constrained environments.

References

Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. 2025. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*.

Alon Albalak, Duy Phung, Nathan Lile, Rafael Rafailov, Kanishk Gandhi, Louis Castricato, Anikait Singh, Chase Blagden, Violet Xiang, Dakota Mahan, and 1 others. 2025. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models. *arXiv preprint arXiv:2502.17387*.

Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro Von Werra. 2022. A framework for the evaluation of code generation models.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Weilin Cai, Juyong Jiang, Le Qin, Junwei Cui, Sunghun Kim, and Jiayi Huang. 2024. Shortcut-connected expert parallelism for accelerating mixture-of-experts. *arXiv preprint arXiv:2404.05019*.

Federico Cassano, John Gouwar, Daniel Nguyen, Sydney Nguyen, Luna Phipps-Costin, Donald Pinckney, Ming-Ho Yee, Yangtian Zi, Carolyn Jane Anderson, Molly Q Feldman, and 1 others. Multipl-e: A scalable and extensible approach to benchmarking neural code generation, 2022. *URL https://arxiv. org/abs/2208.08227*.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and 1 others. 2023. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233.

Yao Cheng, Jianfeng Chen, Jie Chen, Li Chen, Liyu Chen, Wentao Chen, Zhengyu Chen, Shijie Geng, Aoyan Li, Bo Li, and 1 others. 2024. Full-stack bench: Evaluating llms as full stack coders. *arXiv preprint arXiv:2412.00535*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378.

Kaja Gruntkowska, Yassine Maziane, Zheng Qu, and Peter Richtárik. 2025. Drop-muon: Update less, converge faster. *arXiv preprint arXiv:2510.02239*.

Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I Wang. 2024. Cruxeval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*.

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

Wei He, Kai Han, Hang Zhou, Hanting Chen, Zhicheng Liu, Xinghao Chen, and Yunhe Wang. 2025. Root: Robust orthogonalized optimizer for neural network training. *arXiv preprint arXiv:2511.20626*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob

Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Yao Fu, and 1 others. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *Advances in neural information processing systems*, 36:62991–63010.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Naman Jain, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. 2025. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *International Conference on Learning Representations*, volume 2025, pages 58791–58831.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. 2024. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan. github. io/posts/muon*, 6(3):4.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Ahmed Khaled, Kaan Ozkara, Tao Yu, Mingyi Hong, and Youngsuk Park. 2025. Muonbp: Faster muon via block-periodic orthogonalization. *arXiv preprint arXiv:2510.16981*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pages 18319–18345. PMLR.

Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmlu: Measuring massive multitask language understanding in chinese, 2024. *URL https://arxiv. org/abs/2306.09212*.

Jian Li and Mingyi Hong. 2025. A note on the convergence of muon and further. *arXiv preprint arXiv:2502.16982*.

Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, and 1 others. 2024a. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024b. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Hong Liu, Jiaqi Zhang, Chao Wang, Xing Hu, Linkun Lyu, Jiaqi Sun, Xurui Yang, Bo Wang, Fengcun Li, Yulei Qian, Lingtong Si, Yerui Sun, Rumei Li, Peng Pei, Yuchen Xie, and Xunliang Cai. 2026. Scaling embeddings outperforms scaling experts in language models. *Preprint*, arXiv:2601.21204.

Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. 2024c. Evaluating language models for efficient code generation. *arXiv preprint arXiv:2408.06450*.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, and 1 others. 2025. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

James Martens and Roger Grosse. 2015. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and 1 others. 2017. Mixed precision training. *arXiv preprint arXiv:1710.03740*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*.

Herbert Robbins and Sutton Monro. 1951. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Social iqa: Commonsense reasoning about social interactions. In *EMNLP*.

Chongjie Si, Debing Zhang, and Wei Shen. 2025. Adamuon: Adaptive muon optimizer. *arXiv preprint arXiv:2507.11005*.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. pmlr.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and 1 others. 2023. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, and 150 others. 2025a. Kimi k2: Open agentic intelligence. *Preprint*, arXiv:2507.20534.

Meituan LongCat Team, Bei Li, Bingye Lei, Bo Wang, Bolin Rong, Chao Wang, Chao Zhang, Chen Gao, Chen Zhang, Cheng Sun, and 1 others. 2025b. Longcat-flash technical report. *arXiv preprint arXiv:2509.01322*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290.

Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. 2025. Leetcodedataset: A temporal dataset for robust evaluation and efficient training of code llms. *arXiv preprint arXiv:2504.14655*.

Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. 2024. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9508–9520.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Yang You, Igor Gitman, and Boris Ginsburg. 2017. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2019. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

Daoguang Zan, Bei Chen, Zeqi Lin, Bei Guan, Yongji Wang, and Jian-Guang Lou. 2022. When language model meets private library. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 277–288.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Tong Zhao, Jiacheng Li, Yuanchang Zhou, Guangming Tan, and Weile Jia. 2026. Exploring landscapes for better minima along valleys. *Advances in Neural Information Processing Systems*, 38:171496–171547.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, and 1 others. 2025. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations*, volume 2025, pages 66602–66656.

A Proofs of Theoretical Results

A.1 Proof of Lemma 1

Proof. From $A_k = \beta_a A_{k-1} + (1-\beta_a)(G_k - G_{k-1})$ and the triangle inequality,

$$\|A_k\|_F \le \beta_a \|A_{k-1}\|_F + 2(1-\beta_a)G. \quad (21)$$

Unrolling with $A_{-1} = 0$,

$$\|A_k\|_F \le 2G(1-\beta_a^k) \le 2G. \quad (22)$$

For $\tilde{G}_k = G_k + \alpha A_k$,

$$\|\tilde{G}_k\|_F \le G + |\alpha| \cdot 2G = G(1+2|\alpha|). \quad \square \quad (23)$$

A.2 Proof of Lemma 2

Proof. The update $M_k = \mu M_{k-1} + \tilde{G}_k$ gives

$$\|M_k\|_F \le \mu \|M_{k-1}\|_F + G(1+2|\alpha|). \quad (24)$$

Unrolling with $M_0 = 0$,

$$\|M_k\|_F \le \frac{G(1+2|\alpha|)}{1-\mu}. \quad (25)$$

$\square$

A.3 Proof of Theorem 1

Proof. By L-smoothness,

$$f(W_{k+1}) \le f(W_k) + \langle \nabla f(W_k), W_{k+1} - W_k \rangle + \frac{L}{2}\|W_{k+1} - W_k\|_F^2. \quad (26)$$

Substituting $W_{k+1} - W_k = -\eta\gamma O_k$,

$$f(W_{k+1}) \le f(W_k) - \eta\gamma \langle \nabla f(W_k), O_k \rangle + \frac{\eta^2\gamma^2 L}{2}\|O_k\|_F^2. \quad (27)$$

By Assumption 4, $\mathbb{E}[\langle \nabla f(W_k), O_k \rangle \mid W_k] \ge \rho \|\nabla f(W_k)\|_F^2$. Thus:

$$\mathbb{E}[f(W_{k+1}) \mid W_k] \le f(W_k) - \eta\gamma\rho\|\nabla f(W_k)\|_F^2 + \frac{\eta^2\gamma^2 L}{2}\mathbb{E}[\|O_k\|_F^2 \mid W_k]. \quad (28)$$

For $\|O_k\|_F$: since Newton-Schulz outputs approximately orthogonal matrices with singular values close to 1,

$$\|O_k\|_F^2 \le C_m \quad \text{for some constant} \quad C_m = O(r). \quad (29)$$

Rearranging and taking full expectation,

$$\eta\gamma\rho\,\mathbb{E}[\|\nabla f(W_k)\|_F^2] \le \mathbb{E}[f(W_k) - f(W_{k+1})] + \frac{\eta^2\gamma^2 L C_m}{2}. \quad (30)$$

Summing over $k = 0, \ldots, K-1$ and using $f(W_k) \ge f^*$,

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(W_k)\|_F^2] \le \frac{f(W_0) - f^*}{\eta\rho\gamma K} + \frac{\eta\gamma L C_m}{2\rho}, \quad (31)$$

which matches the stated bound with $C_4 = \gamma^2 C_m/2$. With $\eta = O(K^{-1/2})$, this gives $O(K^{-1/2})$ rate.

$\square$

A.4 Proof of Proposition 1

Proof. Near W^*, $\nabla f(W) \approx H^*(W - W^*)$, so

$$\mathbb{E}[G_k - G_{k-1} \mid W_k, W_{k-1}] \approx H^*(W_k - W_{k-1}) = -\eta\gamma H^* O_{k-1}. \quad (32)$$

The acceleration $A_k = \beta_a A_{k-1} + (1-\beta_a)(G_k - G_{k-1})$ yields, taking expectation and unrolling,

$$\mathbb{E}[A_k] \approx -\eta\gamma H^* \sum_{j=0}^{k}(1-\beta_a)\beta_a^{k-j} O_{j-1}. \quad (33)$$

With $H^* = Q\Lambda Q^\top$ and eigenvalues $\lambda_1 \ge \cdots \ge \lambda_d > 0$, in the eigenbasis,

$$\mathbb{E}[A_k^{(i)}] \approx -\eta\gamma\lambda_i \sum_{j=0}^{k}(1-\beta_a)\beta_a^{k-j} O_{j-1}^{(i)}. \quad (34)$$

For large λ_i (sharp directions), $|\mathbb{E}[A_k^{(i)}]|$ is large, pushing the optimizer away. For small λ_i (flat directions), it is small, allowing settlement. With $\alpha < 0$, $\tilde{G}_k = G_k + \alpha A_k$ selectively amplifies flat-direction updates. $\square$

B Additional Validation Curves

Figures 9–12 present supplementary validation loss curves for the mathematical reasoning and Chinese academic text domains.

Figure 9: Validation loss on Mathematical-Reasoning for MOE-1B-A0d2B.

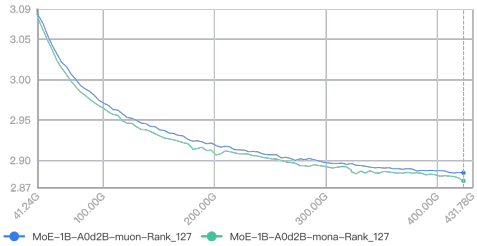

Figure 10: Validation loss on Chinese-Academic-Text for MOE-1B-A0d2B.

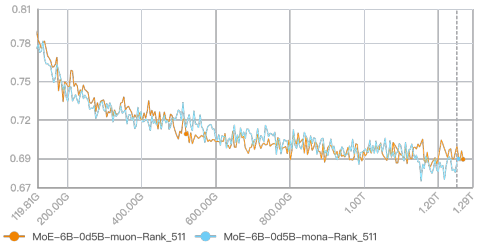

Figure 11: Validation loss on Mathematical-Reasoning for MOE-6B-A0d5B.

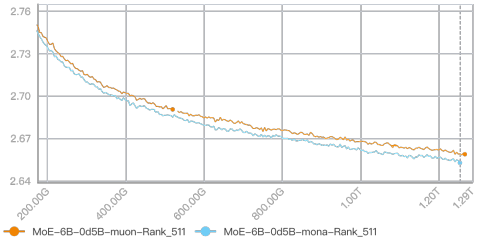

Figure 12: Validation loss on Chinese-Academic-Text for MOE-6B-A0d5B.

C Computational Overhead Analysis

The memory overhead of MONA's additional buffers was already discussed in Section 6.3. Here, we add measurements of computational time overhead from the MOE-6B-A0d5B pretraining run.

Figure 13 shows the optimizer inner step time. This is the wall-clock time spent inside the optimizer update at each training step, not counting communication or data loading. MONA adds several simple operations while updating the acceleration item. These add a small amount of time inside the optimizer. By sampling points across the training run, we find that MONA's optimizer inner step time is about **1%** higher than Muon's on average.

Figure 14 reports the end-to-end iteration time, covering the full training step from the forward pass to gradient synchronization and data loading. At this level, the difference between MONA and Muon is too small to notice. The tiny overhead inside the optimizer is completely hidden by the natural variation in communication and data loading between steps. Overall, MONA and Muon run at essentially the same speed in practice.

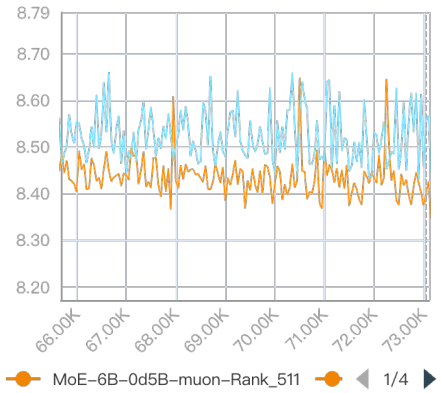

Figure 13: Optimizer inner step time for MOE-6B-A0d5B pretraining. MONA runs about 1% slower than Muon at the optimizer step level.

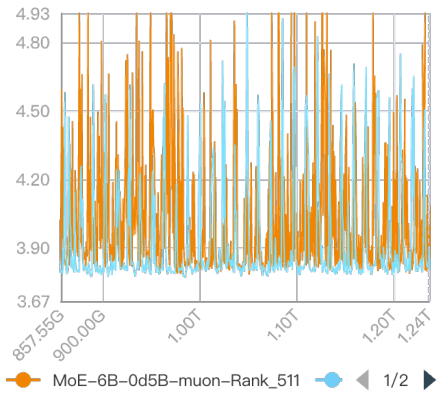

Figure 14: End-to-end iteration time for MOE-6B-A0d5B pretraining. MONA and Muon show no practical difference in overall training speed.

D Comparison with Accelerated AdamW

To better understand where the acceleration gains come from, we compare MONA against not only Muon and AdamW but also an AdamW variant equipped with the same acceleration term. We call this variant AdamW-Acc. It is adapted from ALTO's acceleration mechanism, but with one key change for fair comparison in the pretraining setting.

ALTO uses layer-wise learning rate regularization, which assigns different learning rates to different layers. In practice, pretraining typically uses a relatively large fixed learning rate, while post-training applies lower learning rates along with various schedulers. If pretraining already introduces layer-wise learning rate dynamics, it effectively consumes some of the learning rate scheduling benefits that would otherwise belong to the post-training stage. To avoid this, we replace the layer-wise regularization with a default uniform learning rate. This makes AdamW-Acc a clean baseline that isolates the acceleration benefit.

We run all four optimizers, AdamW, AdamW-Acc, Muon, and MONA, on the MOE-1B-A0d2B model under identical settings. Figure 15 shows the training loss curves, and Figure 16 shows the validation loss on the General-English-Text domain.

The results follow a consistent ordering. MONA achieves the lowest loss, followed by Muon, then AdamW-Acc, and finally AdamW. This confirms two observations. First, the acceleration term itself brings measurable improvement, since AdamW-Acc outperforms AdamW. Second, Muon already provides a more stable training foundation than AdamW, with smoother loss curves and no spike issues during pretraining. The acceleration term builds on this stable base and pushes performance further, rather than merely recovering from instability.

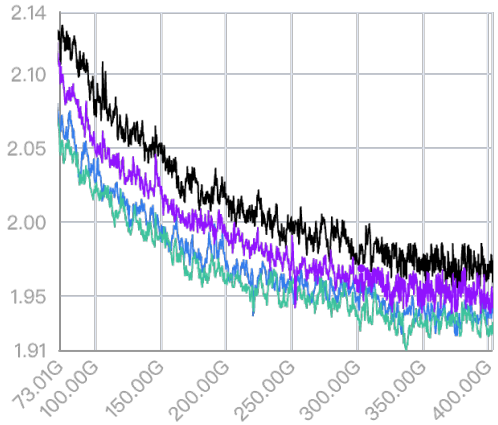

Figure 15: Training loss for MOE-1B-A0d2B with four optimizers: AdamW(black), AdamW-Acc(purple), Muon(blue), and MONA(green).

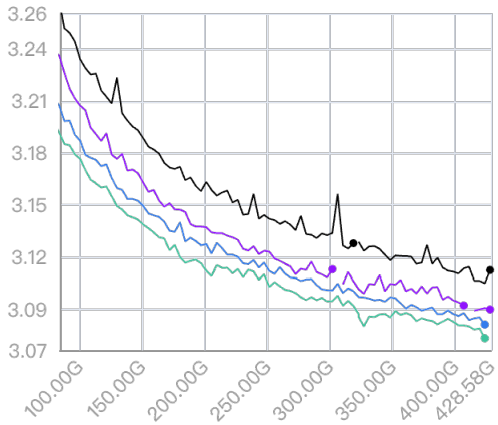

Figure 16: Validation loss on General-English-Text for MOE-1B-A0d2B. MONA outperforms Muon, which in turn outperforms AdamW-Acc and AdamW.