AMO: OPERATOR-LEVEL ADAPTIVE MUON ORTHOGONALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Muon has recently emerged as a competitive alternative to AdamW for large-scale pre-training, with orthogonalization via Newton–Schulz (NS) iteration as its core operation. Standard Muon applies a uniform NS schedule to all parameter matrices, overlooking possible differences in orthogonalization difficulty and its impact on performance. Through a systematic empirical study, we show that this per-matrix heterogeneity is pervasive and strongly associated with matrix geometry, which evolves dynamically across operator types, training stages, and network depths. Therefore, uniform NS schedules can lead to uneven orthogonalization quality across the model. Motivated by these findings, we propose Operator-level Adaptive Muon Orthogonalization (AMO), an observe-then-commit method that measures weight geometry by operator type early in training and then uses these signals to allocate the NS budget for the remainder of training. AMO delivers consistent improvements over uniform-schedule Muon across **standard**, **prolonged**, and **continual** pre-training, surpassing the strongest baseline by +0.76 on Llama3.1-1.4B and +0.51 on Qwen3-1.7B in average downstream performance of 12 evaluation tasks, with gains persisting at Llama3.1-4B scale.

1 INTRODUCTION

Muon (Jordan et al., 2024) has emerged as a competitive alternative to AdamW for large-scale model pre-training (Wang et al., 2026; Frans et al., 2025), with its core operation being the orthogonalization via Newton-Schulz (NS) iterations (DeepSeek-AI, 2026; Team, 2026; Team et al., 2026; AI et al., 2025; Liu et al., 2025a; Zhang et al., 2026; Si et al., 2025; Li et al., 2025). Two key design choices governing NS orthogonalization are the count of NS iteration steps T and the coefficient sequence $\{(a_k, b_k, c_k)\}_{k=1}^{T}$. Prior works have improved NS through globally shared coefficient sequences, including optimization-fitted coefficients and Chebyshev-based designs (Amsel et al., 2026; Grishina et al., 2026), while ROOT (He et al., 2025) takes an important step towards adaptive Muon by specializing NS coefficients to **matrix shape**. Yet ROOT's shape-level adaptation still treats matrices with the same shape as interchangeable, leaving an important question unresolved: *do matrices truly differ in how difficult they are to orthogonalize, and if so, along which dimension?*

We conduct a systematic pilot study to investigate this question (Sec. 2). **First**, a factorial study over two model families, two optimizers, four learning-rate (LR) schedulers, and three data orderings shows that monotone quality-based ordering does not reliably improve over random mixing (Fig. 2), suggesting that such data-side curricula are not a reliable remedy under the evaluated settings and motivating optimizer-side adaptation. **Second**, a model geometry analysis tracking momentum matrices updated by Muon reveals that heterogeneity is real and structured: a geometry-derived input statistic predicts NS quality (signal validity, Fig. 3(**a**)); operator types exhibit persistent, well-separated difficulty trajectories (operator-type structure, Fig. 3(**b**)); per-layer profiles evolve similarly and differ only marginally in magnitude (weak layer magnitude, Fig. 3(**c**)); and even those marginal layer differences reverse during training (non-stationary layer ordering, Fig. 3(**d**)). Together, these findings motivate **operator type** as an effective and compact unit of adaptation and support measuring the control signal from training dynamics rather than relying on a fixed matrix-level proxy.

Taken together, these observations motivate **operator-type adaptive NS allocation**: *how to allocate NS effort non-uniformly across parameter matrices based on their measured optimization difficulty?* To address this problem, we propose **Operator-level Adaptive Muon Orthogonalization (AMO)**,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

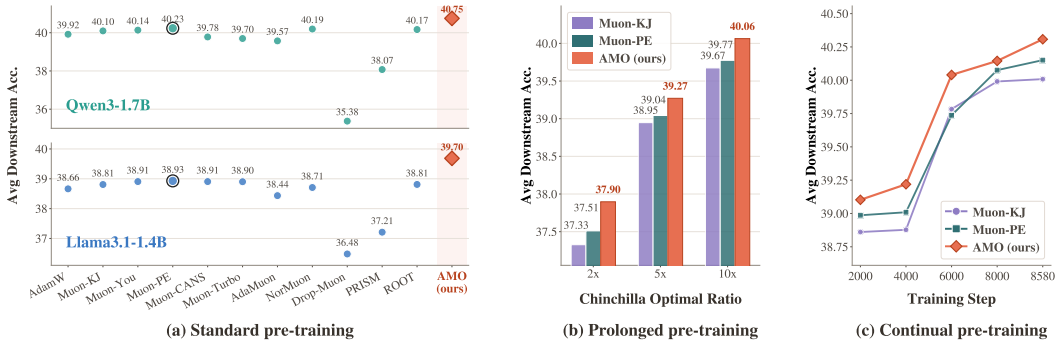


Figure 1: Results of AMO and baseline optimizers on **(a)** standard pre-training across model families and scales, **(b)** prolonged pre-training (2×/5×/10× Chinchilla optimal ratios) on Qwen3-0.6B, and **(c)** continual pre-training of Qwen3-0.6B on Slimpajama. AMO consistently leads in all three settings.

a simple observe-then-commit method that observes per-operator-type geometry signals under the common uniform schedule for a fixed window, computes a single allocation plan via greedy budget reallocation over projected NS error curves, then linearly transitions to that plan and locks it for the remainder of training. Extensive experiments on Llama3.1 and Qwen3 models ranging from 0.6B to 1.7B parameters show that AMO consistently outperforms uniform-schedule Muon and other baselines across model scales in standard pre-training, prolonged pre-training, and continual pre-training, demonstrating the effectiveness of AMO (Fig. 1).

Our contributions are threefold. First, we motivate **optimizer-side adaptation** by showing that monotone quality-based data curricula do not reliably improve over random mixing across the evaluated datasets and optimization settings, indicating that such data-side interventions are not a reliable remedy for the observed heterogeneity. Second, we **characterize its structure** through four geometry findings: signal validity, operator-type structure, weak layer magnitude, and non-stationary layer ordering, which together identify where and how heterogeneity manifests and provide the structural basis for operator-type adaptive NS allocation. Third, we **propose AMO**, a simple observe-then-commit method that achieves consistent gains over uniform-schedule Muon methods across standard, prolonged, and continual pre-training, demonstrating the effectiveness of adaptive Muon.

2 Pilot Analysis

A common design in Muon-family optimizers is to apply a shared NS schedule across updated parameter matrices in the network. Most of them (Amsel et al., 2026; Grishina et al., 2026; Boissin et al., 2025) share one schedule across all matrices and all training steps, while ROOT (He et al., 2025) specializes NS coefficients to **matrix shape** but still treats matrices of the same shape (e.g. `attn_k` and `attn_v` in Llama / Qwen models) as interchangeable. Whether this uniformity is adequate depends on whether per-matrix optimization difficulty is truly uniform, and if not, on *where* the heterogeneity comes from. We therefore ask two questions. **Q1**: Can curriculum learning reduce per-matrix heterogeneity enough to remove the need for optimizer-side adaptation? **Q2**: Is the structure of this heterogeneity **stable enough** across operator type, model depth, or training step to admit an adaptive scheme? Our experiments suggest no to **Q1** under the evaluated settings (Sec. 2.1) and yes to **Q2** (Sec. 2.2). The rest part provides concrete explorations and guides method designs.

2.1 Data Curriculum

Classical curriculum learning predicts that ordering examples by difficulty improves optimization (Luo et al., 2026). Here, we use quality scores (Zhuang et al., 2025) as practical difficulty-related signals for monotone curricula, whose effectiveness in mitigating the observed heterogeneity should manifest as reliable gains in final performance. We test this hypothesis with a factorial study over two base models (Llama3.1-760M, Qwen3-0.6B), two optimizers (both Muon and AdamW are considered), four LR schedulers (Warmup-Stable (WS), Warmup-Stable-Decay (WSD), Cosine, Constant), and three data orderings (`Uniform`, `Ascend`, `Descend`) on the `FineWeb-Edu` dataset (Penedo et al., 2024), where data are sorted by the *Educational Value* score field. The same data pool is used across

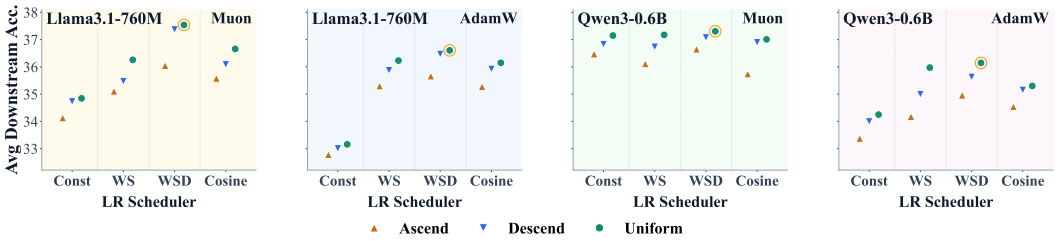


Figure 2: Average downstream accuracy under three data orderings, across 16 configurations. The best result in each subplot is marked with a yellow circle. `Ascend` and `Descend` denote curriculum learning in increasing/decreasing sample score order, while `Uniform` uses a random order.

all runs where only the **data presentation order** differs. Full hyperparameters, configurations, and additional results are provided in App. D.1.

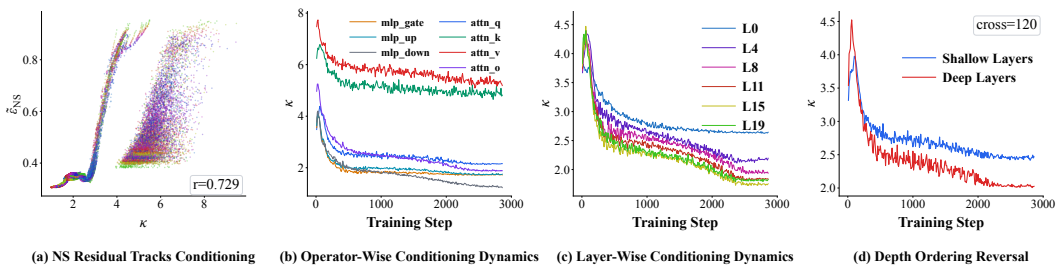
Fig. 2 reports the average downstream accuracy across a fixed evaluation suite for all 16 configurations. The performance ordering `Uniform` > `Descend` > `Ascend` holds consistently in **every** configuration, with no inversion across model, optimizer, or LR scheduler. Notably, `Ascend` and `Descend` traverse the *Educational-Value* score in opposite directions yet both underperform `Uniform`, indicating that neither direction provides a reliable advantage over random mixing in these experiments. This pattern suggests that monotone quality-based ordering is generally not a reliable remedy across the evaluated datasets and optimization settings, while also indicating a non-negligible interaction between curriculum ordering and the LR schedule. We therefore turn to optimizer-side geometry to directly characterize the observed per-matrix heterogeneity.

2.2 Model Geometry

Let $M_t \in \mathbb{R}^{m\times n}$ denote one Muon momentum matrix at step t. With Nesterov momentum, $M_t = G_t + \mu B_t$, $B_t = \mu B_{t-1} + G_t$, where G_t is the stochastic gradient and B_t is the momentum buffer. Muon orthogonalizes M_t via T-step NS iterations, producing M_t'. We compute, for each Muon-updated 2D matrix in every transformer block (seven operator types: `attn_{q,k,v,o}`, `mlp_{gate,up,down}`), the condition number of the NS input $\kappa(M_t)$ and the shape-normalized NS residual $\tilde{\epsilon}_{\mathrm{NS}} = \frac{\|M_t' M_t'^{\top} - I\|_F}{\sqrt{\min(m,n)}}$, which is comparable across matrices of different shapes. Fig. 3 summarizes four findings on a representative model Qwen3-0.6B, and analogous patterns hold for all seven remaining Llama3.1 and Qwen3 models with parameters ranging from 70M to 1.7B (App. D.2).

F1 and F2: A geometry signal predicts NS quality and concentrates at operator type. As shown in Fig. 3(**a**), the NS-input condition number $\kappa(M_t)$ is positively correlated with the shape-normalized residual $\tilde{\epsilon}_{\mathrm{NS}}$ with Pearson $r = 0.729$, confirming that a geometry-derived input-side statistic provides an informative proxy for NS quality (**F1**). App. D.2 shows that Pearson r lies in $[0.722, 0.813]$ for Llama3.1 and $[0.671, 0.912]$ for Qwen3. Operator-type trajectories of median κ are persistently separated throughout training (Fig. 3(**b**)), with between-type variation visibly dominating within-type variation across layers (**F2**). Moreover, the dominant hard operators differ across families: `attn_v` on Qwen3 and `attn_q` and `attn_o` on Llama3.1, with no single operator dominating across families. This observation suggests that matrix shape alone may miss operator-specific heterogeneity even when different operators share the same shape. These results therefore motivate **operator type** as a compact unit of adaptation.

F3 and F4: Per-layer differences are modest and non-stationary. Fig. 3(**c**) tracks the layer-wise median κ, aggregated across operator types, for six layers spanning the network depth (L0, L4, L8, L11, L15, L19). These trajectories evolve similarly throughout training and differ only modestly in magnitude, in contrast to the pronounced gaps across operator types in Fig. 3(**b**) (**F3**). Moreover, Fig. 3(**d**) compares shallow (first three layers) and deep (last three layers) groups of `attn_q`: their relative ordering *reverses* at step 120, so the group that is harder to orthogonalize early in training becomes the easier one later (more discussion in App. G.3) (**F4**). Depth-based difficulty ordering is therefore not fixed, and layer depth alone provides an insufficiently stable basis for prescribing the allocation schedule a priori. Instead, the allocation signal should be measured from the running model,



(a) NS Residual Tracks Conditioning (b) Operator-Wise Conditioning Dynamics (c) Layer-Wise Conditioning Dynamics (d) Depth Ordering Reversal

Figure 3: Geometry findings on Qwen3-0.6B. κ is shown at $\log_{10}$ scale. **(a)** NS-input conditioning $\kappa(M_t)$ vs. shape-normalized residual $\tilde{\epsilon}_{\mathrm{NS}}$ (Pearson's $r = 0.729$). Each point represents one parameter matrix at one training step, colored by layer depth. **(b)** Median κ trajectories per operator type: `attn_v` dominates the hardest regime, `mlp_down` the easiest. **(c)** Layer-wise median κ trajectories for six layers show similar evolution patterns. **(d)** Median κ trajectories of shallow (first three) vs. deep (last three) layers for `attn_q`; their relative ordering reverses at step 120, indicating that difficulty cannot be predicted from depth alone.

and the allocation unit should preserve its relative difficulty over training, a property that operator type exhibits but depth does not. Together, these four findings motivate the design requirements for adaptive Muon, which we instantiate as a simple observe-then-commit method in Sec. 3.

3 Method

3.1 Preliminaries

Muon and NS Iteration. Muon orthogonalizes momentum M via T-step NS iterations of the form

$$M \leftarrow a_k M + b_k M M^\top M + c_k (M M^\top)^2 M, \qquad k = 1, \ldots, T, \tag{1}$$

parametrized by the coefficient sequence $\{(a_k, b_k, c_k)\}_{k=1}^{T}$. The *only* degrees of freedom in NS design are therefore the iteration count T and the coefficient sequence. The input-side geometry signal $\ell(M) = \sigma_{\min}^{(\tau)}(M)/\|M\|_F$, where τ is a truncation threshold and singular values below τ are treated as τ and $\|\cdot\|_F$ denotes the matrix Frobenius norm, is monotone in $1/\kappa(M)$ in the well-conditioned regime when computed from exact Singular Value Decomposition (SVD).

Existing NS designs are global across matrices updated by Muon. Muon (Jordan et al., 2024) fits a single coefficient sequence, whereas You fits T coefficient sequences through optimization and applies them to every matrix at each training step. More recently, PE and CANS (Amsel et al., 2026; Grishina et al., 2026) derive T coefficient sequences from Chebyshev-based conservative bounds assuming a universal worst-case input ($\ell = 10^{-3}$, a conservative lower bound on the geometry signal across matrices and steps). ROOT (He et al., 2025) specializes coefficient sequences to *matrix shape* but treats matrices of the same shape identically, ignoring the operator-type heterogeneity documented in **F2**. None of these methods uses a runtime-measured signal, and none adapts at operator-type granularity, the two gaps Sec. 2 identifies.

From findings to design requirements. The four geometry findings (**F1-F4**) inspire the following three design requirements. **D1** (from **F1**): *use ℓ as the control signal for optimization difficulty* because it is a valid predictor of NS quality, integrates directly with PE-style polynomial composition (Amsel et al., 2026), and can be computed directly from SVD. **D2** (from **F2+F3+F4**): *adapt at operator-type granularity*. **F2** confirms operator type as a meaningful unit and **F3+F4** jointly rule out layer-wise adaptation. **D3** (from the dynamic nature of **F3**, **F4**): *acquire the signal through online observation*, since the geometry cannot be reliably prescribed from architectural priors (we discuss static NS coefficients in Sec. 5.2 and static NS step in App. G.2). Moreover, a fourth choice: committing to a single allocation rather than continuously re-tuning, is methodological: **F2**'s persistent operator-type separation (Fig. 3(**b**)) shows the cross-type ordering does not undergo destabilizing changes during training, while continuous re-tuning would cause optimizer dynamics to become unstable.

3.2 Adaptive Muon Orthogonalization

Following **D1-D3**, the core operation of **AMO** is to allocate a fixed NS step budget across the seven operator types $\mathcal{O} = \{\texttt{attn_}\{\texttt{q,k,v,o}\}, \texttt{mlp_}\{\texttt{gate,up,down}\}\}$ and recompute the coefficient

Algorithm 1 Adaptive Muon Optimizer (AMO)

Require: budget ratio r, baseline step count T_{base}, step range $[k_{\min}, k_{\max}]$, shrinkage α, baseline ℓ_{base}, observation interval Δ_{obs}, observation end step T_{obs}, observation sample size n_{smp}, transition duration Δ

compute total budget $B \leftarrow \text{round}(r \cdot |\mathcal{O}| \cdot T_{\text{base}})$

Phase 1 (Observation): run training under uniform T_{base} schedule; every Δ_{obs} steps, record per-type median $\ell_{\text{eff}}^{(i)}$ from n_{smp} matrices within each operator type $i \in \mathcal{O}$. Exit at step T_{obs}.

Plan: $\ell_{\text{target}}^{(i)} \leftarrow \alpha \cdot \text{median-of-medians}(\ell_{\text{eff}}^{(i)}) + (1-\alpha) \cdot \ell_{\text{base}}$ for each $i \in \mathcal{O}$

For each $i \in \mathcal{O}$ and $k \in [k_{\min}, k_{\max}]$: $E_i(k) \leftarrow 1 - \text{PE-SIMULATE}(\ell_{\text{target}}^{(i)}, k)$

$\{T_i^\star\}_{i \in \mathcal{O}} \leftarrow \text{BUDGETALLOCATE}(\{E_i\}, B)$ ▷ greedy *add* / *remove* / *transfer*

Phase 2 (Transition): for $u = 1, \ldots, \Delta$ **do**

  $p \leftarrow u/\Delta$

  for each $i \in \mathcal{O}$: $T_i(u) \leftarrow T_{\text{base}} + p\,(T_i^\star - T_{\text{base}})$, $\ell_{\text{eff}}^{(i)}(u) \leftarrow \ell_{\text{base}} + p\,(\ell_{\text{target}}^{(i)} - \ell_{\text{base}})$

  $B(u) \leftarrow \text{round}\big(\sum_{i \in \mathcal{O}} T_i(u)\big)$; round $\{T_i(u)\}$ s.t. $\sum_{i \in \mathcal{O}} T_i(u) = B(u)$

  regenerate PE coefficients at $(T_i(u), \ell_{\text{eff}}^{(i)}(u))$ for each $i \in \mathcal{O}$

end for

Lock: freeze the schedule and coefficients; stop geometry capture.

sequence for each operator type, following the observe-then-commit methodological design above. The complete workflow of AMO is provided in Alg. 1.

Phase 1: Observation (D3). Under the uniform baseline schedule, AMO captures the optimizer-internal momentum matrix M_t every Δ_{obs} steps, until reaching the observation end step T_{obs}. For each operator type, a fixed-size subset of n_{smp} matrices is uniformly sampled for observation, rather than using all matrices of that type, to improve efficiency. Per observation we record the per-type median of $\ell_{\text{eff}}^{(i)} = \ell_{\text{raw}}^{(i)}/s$, where $s = 1.01$ is a scale factor used for numerical safety. After a total of $T_{\text{obs}}/\Delta_{\text{obs}}$ observations, we have collected sufficient geometry observations to invoke planning.

Planning (D1, D2). Aggregation uses a median-of-medians, the median, over observations, of each observation's per-type median across sampled matrices, yielding a robust per-type estimate $\ell_{\text{robust}}^{(i)}$ that is shrunk toward a conservative baseline ℓ_{base},

$$\ell_{\text{target}}^{(i)} \;=\; \alpha \cdot \ell_{\text{robust}}^{(i)} \;+\; (1-\alpha) \cdot \ell_{\text{base}}, \tag{2}$$

where the shrinkage coefficient α stabilizes allocation against single-type overconfidence. For each type i and each candidate step count $k \in [k_{\min}, k_{\max}]$ (we empirically set $k_{\min} = 3$ and $k_{\max} = 7$), we invoke the PE composer at $(\ell_{\text{target}}^{(i)}, k)$ and simulate the resulting NS trajectory, giving a *projected* error curve $E_i(k) = 1 - \ell_{\text{after-final}}$, where $\ell_{\text{after-final}}$ is ℓ on the matrix produced by the final NS iteration. Searching over projected PE error curves rather than mapping ℓ directly to a step count correctly accounts for the joint effect of T and the PE coefficients fitted at that T. Under the budget constraint $\sum_i T_i = B$, a discrete greedy routine uses only the ranking of per-type marginal gains and losses: *add*, *remove*, *transfer*, with details and an optimality statement provided in App. C.

Phase 2: Transition and Lock. A discontinuous switch from the baseline schedule $(T_{\text{base}}, \ell_{\text{base}})$ to the planned schedule $(T_i^\star, \ell_{\text{target}}^{(i)})$ would cause the orthogonalized update M_t' to jump, destabilizing the momentum buffer B_t whose accumulated gradients were produced under the old schedule. AMO therefore distributes the change over a transition duration of Δ steps via joint linear interpolation. With $p(u) = u/\Delta$ for each $u = 1, 2, \ldots, \Delta$,

$$T_i(u) = T_{\text{base}} + p(u)\big(T_i^\star - T_{\text{base}}\big), \qquad \ell_{\text{eff}}^{(i)}(u) = \ell_{\text{base}} + p(u)\big(\ell_{\text{target}}^{(i)} - \ell_{\text{base}}\big). \tag{3}$$

Two consistency rules apply during transition. (i) *Two-stage rounding.* Rounding each $T_i(u)$ independently would let the running total drift off the planned trajectory, so we first round the interpolated total $B(u) = \text{round}(\sum_{i \in \mathcal{O}} T_i(u))$, then round per-type counts under the constraint $\sum_i T_i(u) = B(u)$. The total therefore advances monotonically from $|\mathcal{O}| \cdot T_{\text{base}}$ at $u = 0$ to B at $u = \Delta$. (ii) *Coefficient regeneration.* At every NS-applied step, PE coefficients are regenerated at the current interpolated state $(T_i(u), \ell_{\text{eff}}^{(i)}(u))$, so AMO adapts the full coefficient table, not only the step count. At step $u = \Delta$, the schedule and coefficients freeze and geometry capture stops; the persistent cross-type ordering established in **F2** ensures that locking does not discard a useful signal. Overall, observation horizon T_{obs}, observation interval Δ_{obs}, observation sample size n_{smp}, shrinkage coefficient α, and transition duration Δ are discussed in Sec. 4.4.

4 EXPERIMENTS AND RESULTS

4.1 EXPERIMENTAL SETUP

Optimizers. We compare AMO against AdamW and several Muon-style optimizers. AdamW uses $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay 0.1, and a WSD LR schedule. For all Muon variants (Muon (Jordan et al., 2024), AdaMuon (Si et al., 2025), You, PE (Amsel et al., 2026), CANS (Grishina et al., 2026), ROOT (He et al., 2025), and Turbo (Boissin et al., 2025)), we apply Muon to 2D matrices in Transformer blocks and AdamW to embeddings, output heads, biases, and normalization parameters. Muon uses Nesterov momentum 0.95, weight decay 0.1, and five NS iterations with the official KJ coefficients (Jordan et al., 2024), followed by a $0.2\sqrt{\max(m,n)}$ rescaling for an $m \times n$ matrix (Liu et al., 2025a). The base learning rate follows the empirical scaling rule of Zhou et al. (2026), with a cosine-decay schedule for all Muon-style runs. Implementation details of optimizers utilized are provided in App. F.1.

Pre-training. We pre-train eight dense decoder-only Transformers from the Llama-3.1 and Qwen3 families (AI@Meta, 2024; Yang et al., 2025), spanning four scales per family from 70M to 1.7B parameters. Except for continual pre-training, all models are trained from scratch with the official Gaussian initialization. All models use grouped-query attention (Ainslie et al., 2023), RMSNorm (Zhang & Sennrich, 2019), and SwiGLU feed-forward blocks (Shazeer, 2020). Training data are sampled from `FineWeb-Edu` (Penedo et al., 2024) and packed into 2048-token sequences with each family's native tokenizer. Standard pre-training uses the Chinchilla-optimal budget of 20 tokens per parameter (Hoffmann et al., 2022) (roughly 1B–34B tokens across scales), and prolonged pre-training uses $2\times$, $5\times$, and $10\times$ this budget. All runs use bfloat16 mixed precision, Flash-Attention-2 (Dao, 2024), gradient clipping at norm 1.0, and random seed 42, on 6–32 NVIDIA H100 (80GB) GPUs.

Evaluation. We evaluate all model checkpoints using the `lm-evaluation-harness` (Gao et al., 2024) framework on 12 standard downstream benchmarks spanning three capability axes: *commonsense reasoning* (HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), OpenBookQA (Mihaylov et al., 2018), CommonsenseQA (Talmor et al., 2019)), *scientific knowledge* (ARC-Easy, ARC-Challenge (Clark et al., 2018), SciQ (Welbl et al., 2017)), and *general language understanding* (MMLU (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024), RACE (Lai et al., 2017), AGIEval-en (Zhong et al., 2024)). All tasks are evaluated in zero-shot mode to isolate pre-training quality from in-context learning, except MMLU-Pro, which uses its official 5-shot setting. We report accuracy using the framework's default generation likelihood-based scoring, except MMLU-Pro, which uses greedy generation with exact-match per its official protocol. The unweighted average across the 12 benchmarks (*Avg.*) serves as the main metric.

4.2 MAIN RESULTS

Across the evaluated model families and scales, AMO consistently improves over uniform-schedule Muon and other baseline optimizers. Beyond the reported main comparisons, we further verify the robustness of these gains with paired **multi-seed** experiments on Llama3.1-1.4B (App. F.2) and extend the evaluation to **Llama3.1-4B** pre-trained for **100B** tokens (App. F.3). These additional results show that the observed improvements persist across random seeds and at a larger model scale.

AMO outperforms baseline optimizers across model scales and families. As shown in Fig. 1(**a**), AMO achieves the highest average downstream accuracy across two model families, improving over the strongest baseline by +0.76 on Llama3.1-1.4B (vs. Muon-PE, 39.69 vs. 38.93) and +0.51 on Qwen3-1.7B (vs. Muon-PE, 40.75 vs. 40.24) and the same ranking holds at the smaller scale, with +0.20 on Llama3.1-760M and +0.44 on Qwen3-0.6B (full details and results in App. F.4). The gain is also broadly distributed across 12 tasks rather than concentrated on a few benchmarks. Notably, the strongest static baseline varies across models, so no single hand-designed schedule wins everywhere, whereas AMO does so without per-model tuning. We interpret this as evidence that operator-type-aware budget allocation transfers across model scales (760M to 1.7B) and two architectural families, while online adaptation mitigates the brittleness of any fixed coefficient choice.

AMO is effective under prolonged pre-training. To test whether AMO's advantage persists beyond Chinchilla-optimal compute, we extend pre-training on Qwen3-0.6B to **2**$\times$, **5**$\times$, and **10**$\times$ Chinchilla optimal token ratios, consuming **24B**, **60B**, and **120B** tokens. Full results are provided in App. F.5.

Table 1: Practical time and memory cost on pre-training Qwen3-1.7B, measured over 6912 steps under an identical batch configuration. Arrows (↓ and ↑) indicate whether lower or higher is better.

Optimizer	**Step Time ↓** (s)	**Optimizer Update Time ↓** (ms)	**Throughput ↑** (M token/s)	**Peak Memory ↓** (GB)
AdamW	**5.96**	**4.48**	**0.82**	63.75
Muon	6.14	195.81	0.80	**61.12**
AdaMuon	6.19	241.65	0.79	69.00
AMO (ours)	6.17	191.47	0.80	**61.12**

Fig. 1(**b**) shows that AMO remains consistently better than both strong Muon baselines at all three prolonged-compute settings. At 2×, 5×, and 10× Chinchilla ratios, AMO reaches an average downstream accuracy of 37.90, 39.27, and 40.06, improving over the strongest baseline by +0.39, +0.23, and +0.29, respectively. Thus, the advantage of AMO is not only an early-training effect near the standard compute budget, but persists when the same model is trained substantially longer. This suggests that online operator-type-aware allocation continues to make useful budget decisions even as optimization enters later training stages, where a fixed coefficient schedule may become increasingly mismatched to the evolving geometry of different operator type groups.

AMO is effective under continual pre-training. We further evaluate AMO in continual pre-training (CPT), starting from a Qwen3-0.6B checkpoint pre-trained on 60B tokens and training on an additional 36B tokens from SlimPajama (Zhuang et al., 2025; Soboleva et al., 2023). As shown in Fig. 1(**c**), AMO outperforms both Muon-KJ and Muon-PE at every evaluated step. Its margin over the strongest baseline grows from +0.12 at step 2000 to +0.26 at step 6000 and remains positive at the end, where AMO reaches 40.31 versus 40.15 for Muon-PE and 40.01 for Muon-KJ. This indicates that AMO remains effective in the more practical CPT setting, where the optimizer must improve downstream performance without disrupting an already-trained model.

4.3 Cost Analysis

We compare the cost of AMO with baseline methods from both theoretical and practical perspectives. More details are provided in App. E and we discuss the quality-compute tradeoff of AMO in Sec. 5.1.

Theoretical complexity. Let $P = P_\mu + P_{\text{adam}}$ denote the total trainable parameters, where P_μ are 2D matrices updated by the Muon-family rule and P_{adam} are handled by AdamW. For the i-th matrix of layer type t, one NS iteration costs $C_{t,i} = \Theta\big(\min(m_{t,i}, n_{t,i})^2 \max(m_{t,i}, n_{t,i})\big)$. In terms of **memory**, AdamW and AdaMuon lie in the heavier $2P$ class, whereas Muon, Muon-PE, and AMO all fall into the lighter $P_\mu + 2P_{\text{adam}}$ class (AMO adds only $\mathcal{O}(|\mathcal{T}|)$ control metadata). In terms of **time**, every Muon-family method is dominated by $\sum_t k_t \sum_i C_{t,i}$; AdaMuon adds only a linear P_μ term on top. The key distinction of AMO is that it replaces the uniform rule $k_t \equiv 5$ with a per-type allocation $k_t^\star$ under a global budget $\sum_t k_t^\star = B$. Because the aggregate weights $\sum_i C_{t,i}$ differ substantially across layer types, fixing the nominal budget does *not* fix the real computation: AMO can reduce actual cost by shifting steps from expensive types to cheaper ones, optimizing the weighted objective $\sum_t k_t \sum_i C_{t,i}$ rather than the average NS depth. This *budget-aware cross-type reallocation* places AMO in a distinct complexity regime from static fixed-budget Muon variants.

Practical cost. Tab. 1 reports wall-clock measurements of pre-training Qwen3-1.7B for 6912 steps using different optimizers. Although Muon-family optimizer updates are roughly 43-54x more expensive than AdamW's element-wise step (191.47-241.65 ms vs 4.48 ms), end-to-end step time grows by only 3.0%-3.8%, because the forward/backward pass (∼5.82 s/step) dominate the training loop. Within the Muon family, AMO's optimizer update (191.47 ms) is lower than Muon (195.81 ms) and AdaMuon (241.65 ms), while its peak memory (61.12 GB) ties with Muon at the lowest value and saves 7.88 GB relative to AdaMuon. In this run the budget ratio is set to 1.0, so AMO does not reduce the nominal total NS budget but redistributes it across operator types, locking to an average of 5.0 NS steps across the seven operator types. The residual wall-clock gap to Muon stems from the low-frequency exact-SVD observation steps (∼15.63 s/observation step, of which 9.47 s is the post-step observation overhead) rather than from the locked-phase update rule; once the schedule is locked, AMO matches Muon at 6.13 s/step and 0.80M tokens/s (training loss curve in App. C.1).

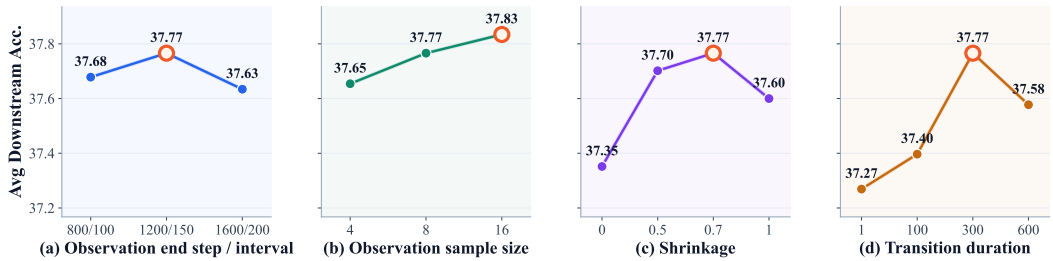


Figure 4: Downstream results of four ablations on pre-training Qwen3-0.6B. Each subplot isolates one AMO hyperparameter configuration, and orange circles mark the best-performing settings.

4.4 Ablation Study

We conduct an ablation study on four key design choices in AMO and report the average downstream accuracy of the resulting Qwen3-0.6B models after pre-training. Full results are provided in App. F.7.

Effect of observation. We jointly examine three observation-stage hyperparameters: the horizon T_{obs} and interval Δ_{obs}, which control when planning is invoked and how many snapshots feed the median-of-medians estimate; and the sample size n_{smp}, the number of matrices drawn per operator type to estimate $\ell_{\text{eff}}^{(i)}$. For the horizon and interval, we compare $(1200, 150)$ against $(800, 100)$ and $(1600, 200)$, keeping the total observation count fixed at $T_{\text{obs}}/\Delta_{\text{obs}} = 8$. Fig. 4(**a**) shows all settings within a narrow band (37.63–37.77) with the middle marginally best, indicating that AMO is largely insensitive to window placement once the observation count is fixed. For the sample size, Fig. 4(**b**) shows a monotone but saturating trend across $n_{\text{smp}} \in \{4, 8, 16\}$ (37.65 $\rightarrow$ 37.77 $\rightarrow$ 37.83): the estimator stabilizes with few samples, so the default $n_{\text{smp}} = 8$ balances stability against capture cost.

Effect of shrinkage. The shrinkage coefficient α interpolates between the measured $\ell_{\text{eff}}^{(i)}$ and the conservative baseline $\ell_{\text{base}} = 10^{-3}$. Notably, $\alpha = 1$ trusts the measurement fully while $\alpha = 0$ entirely recovers the non-adaptive baseline. Fig. 4(**c**) shows a pronounced inverted-U curve over $\alpha \in \{0, 0.5, 0.7, 1.0\}$: $\alpha = 0$ falls to the non-adaptive floor (37.35), $\alpha = 1$ recovers only 37.60, and the shrunk estimators dominate, peaking at $\alpha = 0.7$ (37.77). The two endpoints fail for opposite reasons: $\alpha = 0$ discards the measured signal entirely, while $\alpha = 1$ inherits the residual variance of per-operator estimates, so partial shrinkage yields a better bias-variance trade-off.

Effect of transition duration. The transition duration Δ controls how abruptly AMO moves from the uniform baseline to the AMO target schedule: a very short Δ approximates a step change and risks momentum-buffer inconsistency, while a very long Δ delays the benefit of the optimized allocation. Comparing $\Delta \in \{1, 100, 300, 600\}$, Fig. 4(**d**) shows that the near-instant switch $\Delta = 1$ is clearly weakest (37.27) and $\Delta = 100$ remains suboptimal (37.40), confirming that abrupt transitions destabilize the momentum buffers; performance peaks at $\Delta = 300$ (37.77) and slightly retreats at $\Delta = 600$ (37.58). Therefore, the benefit is captured once the transition is smooth enough to absorb the schedule change and further extension only delays entry into the optimized regime.

5 Analysis and Discussion

5.1 Budget Control

To test whether AMO's gains come *from better allocation or simply from more orthogonalization compute*, we vary the NS budget ratio r on Llama3.1-760M and Qwen3-0.6B, keeping all other settings unchanged (full results in App. F.8). Since the Muon baseline uses 5 NS steps for each of the 7 operator types (a total budget of 35), we set the execution-phase budget to $B = \text{round}(35r)$ with $r \in \{0.8, 0.9, 1.0, 1.1, 1.2\}$. All runs share the same observation phase under the uniform 5-step baseline and differ only in the planning budget, so orthogonalization FLOPs scale approximately linearly with r while all other training compute stays fixed. As shown in Fig. 5(**a**), AMO yields a monotone compute–quality curve on both models, improving from 37.53 to 37.88 on Qwen3-0.6B and from 36.79 to 37.39 on Llama3.1-760M as r increases from 0.8 to 1.2. The curve is also asymmetric:

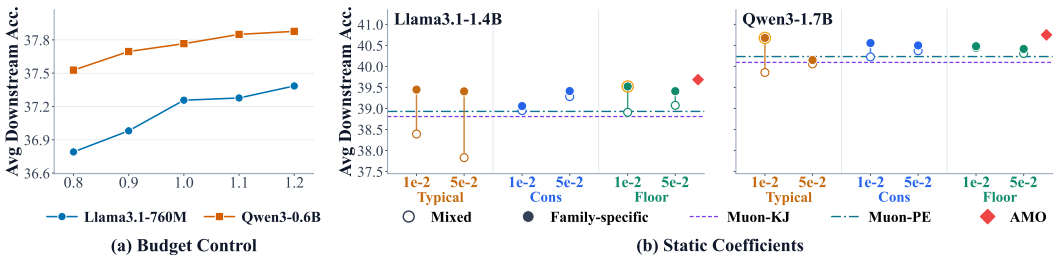


Figure 5: Downstream results of Llama / Qwen models pre-trained **(a)** with five different AMO budget ratios (0.8, 0.9, 1.0, 1.1, 1.2); **(b)** under 12 *mixed* and *family-specific* static NS schedules.

reducing the budget to 0.9 saves ∼8.6% FLOPs at a cost of only 0.07 (Qwen) and 0.28 (Llama) points, whereas increasing it to $r = 1.2$ adds 20% FLOPs for gains of only 0.11 and 0.13.

5.2 Static Coefficients

To isolate the role of *adaptivity*, we compare AMO with static coefficient schedules inspired by PE and CANS, precomputed from the observation data in Sec. 2.2 (details in App. F.9). For Llama3.1-1.4B and Qwen3-1.7B, we sweep three aggregation rules (`Typical`, `Conservative`, `Floor`) and two NS error tolerances (10^{-2}, 5×10^{-2}), estimating each operator-type-wise schedule from either a *mixed* pool across all models or a *family-specific* pool, yielding six paired variants per model. Fig. 5(**b**) reveals three patterns. *(i) Family-specific estimation consistently beats mixed estimation:* it wins all 12 matched pairs, with mean gains of +0.64 on Llama3.1-1.4B and +0.25 on Qwen3-1.7B. *(ii) Even among family-specific schedules, the best aggregation rule is model-dependent:* `Floor-1e-2` is best on Llama3.1-1.4B (39.52), while `Typical-1e-2` is best on Qwen3-1.7B (40.68). *(iii) AMO surpasses the best static schedule on both models without model-specific tuning,* reaching 39.69 (+0.17) on Llama3.1-1.4B and 40.75 (+0.07) on Qwen3-1.7B. These results expose the brittleness of static designs, which must commit before training to an aggregation rule, error tolerance, and estimation pool that do not transfer robustly across model families; AMO sidesteps these choices by deriving its schedule online from the model's own geometry.

5.3 Allocation Granularity

Our geometry analysis motivates operator type as a compact allocation unit, but the modest layer-wise differences leave open whether a finer, depth-aware subdivision could help. We therefore construct a **Type × Depth AMO** variant that splits each of the seven operator types into shallow and deep groups, yielding 14 allocation units instead of 7. On Llama3.1-1.4B, Type × Depth AMO improves the average score over Muon (38.91 vs. 38.66) but trails operator-type AMO (39.55) by 0.64 points. Finer granularity therefore does not help under the same observation and budget setting, suggesting that the depth split does not contribute enough allocation signal to offset the finer partition. This result does not imply that depth information is inherently uninformative or that operator type is universally optimal. Rather, it shows that **operator-type allocation outperforms the tested shallow/deep subdivision under our fixed-budget setting**. Combined with the weak and non-stationary layer differences in Sec. 2.2, this supports operator type as an effective and compact granularity for AMO.

6 Concluding Remarks

In this paper, we challenge the assumption that Muon's NS orthogonalization should be applied uniformly to all parameter matrices. Our geometric analysis shows that per-matrix heterogeneity is intrinsic, structured at the operator-type level, and stable in cross-type ranking. Building on this, we propose AMO, which reallocates a fixed NS budget across operator types based on the model's own geometry, without extra orthogonalization compute or model-specific tuning. Across Llama-3.1 and Qwen3 models up to 1.7B parameters, AMO consistently outperforms uniform-schedule Muon in both pre-training and continual pre-training, and surpasses tuned static schedules. These results suggest treating orthogonalization not as a fixed routine, but as a compute resource to be allocated according to each operator's geometry.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experimental design, assist with the implementation of methods, support the interpretation of experimental results, and assist with the writing, editing, and translation of the manuscript. We have not used generative AI tools to propose and refine research hypotheses, generate synthetic datasets, formulate mathematical claims, provide ingredients for or assist with mathematical proofs, clean or reformat datasets, or conduct qualitative or thematic data analysis. The remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to improve the organization, clarity, and presentation of the manuscript. All AI-assisted written content were independently reviewed by the authors. AI-assisted code and experimental configurations were verified through manual inspection and testing, and AI-assisted interpretations and research suggestions were checked against experimental evidence and relevant literature. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We have made every effort to ensure the reproducibility of our results. Sec. 4.1 describes the experimental setup, including model configurations, training data, token budgets, and optimizer hyperparameters. The complete AMO procedure, including the observation, planning, and execution phases and the NS budget allocation algorithm, is detailed in App. C.1. Pilot analysis setups are provided in App. D.1 and App. D.2 Implementation details and hyperparameters of all baseline optimizers are given in App. F.1.

REFERENCES

Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates, 2025. URL https://arxiv.org/abs/2504.05295. 18

Essential AI, Ishaan Shah, Anthony M. Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, Khoi Nguyen, Kurt Smith, Michael Callahan, Michael Pust, Mohit Parmar, Peter Rushton, Platon Mazarakis, Ritvik Kapila, Saurabh Srivastava, Somanshu Singla, Tim Romanski, Yash Vanjani, and Ashish Vaswani. Practical efficiency of muon for pretraining, 2025. URL https://arxiv.org/abs/2505.02222. 1, 19

AI@Meta. Llama 3 model card, 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md. 6, 34

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, Singapore, 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.emnlp-main.298/. 6, 34

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yRtgZ1K8hO. 1, 2, 4, 6, 18, 19, 20, 21, 37, 42

Kang An, Yuxing Liu, Rui Pan, Yi Ren, Shiqian Ma, Donald Goldfarb, and Tong Zhang. ASGO: Adaptive structured gradient optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=fru52tkjHf. 19

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L Leavitt, and Mansheej Paul. Perplexed by perplexity: Perplexity-based data pruning with small reference models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=1GTARJhxtq. 19

Tianyi Bai, Ling Yang, Zhen Hao Wong, Fupeng Sun, Xinlin Zhuang, Jiahui Peng, Chi Zhang, Lijun Wu, Qiu Jiantao, Wentao Zhang, Binhang Yuan, and Conghui He. Efficient pretraining data selection for language models via multi-actor collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9465–9491. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.acl-long.466/. 19

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239. 6, 37

Thibaut Boissin, Thomas Massena, Franck Mamalet, and Mathieu Serrurier. Turbo-muon: Accelerating orthogonality-based optimization with pre-conditioning, 2025. URL https://arxiv.org/abs/2512.04632. 2, 6, 18, 37

Dan A. Calian, Gregory Farquhar, Iurii Kemaev, Luisa Zintgraf, Matteo Hessel, Jeremy Shar, Junhyuk Oh, András György, Tom Schaul, Jeff Dean, Hado van Hasselt, and David Silver. Datarater: Meta-learned dataset curation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=vUtQFnlDyv. 19

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457. 6, 37

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec. 6, 35

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro. 1

Logan Engstrom, Axel Feldmann, and Aleksander Madry. DsDm: Model-aware dataset selection with datamodels. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12491–12526. PMLR, 2024. URL https://proceedings.mlr.press/v235/engstrom24a.html. 19

Kevin Frans, Pieter Abbeel, and Sergey Levine. What really matters in matrix-whitening optimizers?, 2025. URL https://arxiv.org/abs/2510.25000. 1, 19

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602. 6, 37

GLM-5-Team. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763. 18

Ekaterina Grishina, Matvey Smirnov, and Maxim Rakhuba. Accelerating newton-schulz iteration for orthogonalization via chebyshev-type polynomials, 2026. URL https://arxiv.org/abs/2506.10935. 1, 2, 4, 6, 18, 21, 37, 42

Kaja Gruntkowska, Yassine Maziane, Zheng Qu, and Peter Richtárik. Drop-muon: Update less, converge faster, 2025. URL https://arxiv.org/abs/2510.02239. 18, 37

Yuxian Gu, Li Dong, Hongning Wang, Yaru Hao, Qingxiu Dong, Furu Wei, and Minlie Huang. Data selection via optimal control for language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=dhAL5fy8wS. 19

Wei He, Kai Han, Hang Zhou, Hanting Chen, Zhicheng Liu, Xinghao Chen, and Yunhe Wang. Root: Robust orthogonalized optimizer for neural network training, 2025. URL https://arxiv.org/abs/2511.20626. 1, 2, 4, 6, 19, 37

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ. 6, 37

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556. 6, 35, 36

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Predicting predictions from training data, 2022. URL https://arxiv.org/abs/2202.00622. 19

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/. 1, 4, 6, 18, 36

Ahmed Khaled, Kaan Ozkara, Tao Yu, Mingyi Hong, and Youngsuk Park. Muonbp: Faster muon via block-periodic orthogonalization, 2025. URL https://arxiv.org/abs/2510.16981. 18

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980. 18

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://aclanthology.org/D17-1082. 6, 37

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. Normuon: Making muon more efficient and scalable, 2025. URL https://arxiv.org/abs/2510.05491. 1, 18, 37

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3xHDeA8Noi. 18

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, Yanru Chen, Huabin Zheng, Yibo Liu, Shaowei Liu, Bohong Yin, Weiran He, Han Zhu, Yuzhi Wang, Jianzhou Wang, Mengnan Dong, Zheng Zhang, Yongsheng Kang, Hao Zhang, Xinran Xu, Yutao Zhang, Yuxin Wu, Xinyu Zhou, and Zhilin Yang. Muon is scalable for llm training, 2025a. URL https://arxiv.org/abs/2502.16982. 1, 6, 18, 36

Liming Liu, Zhenghao Xu, Zixuan Zhang, Hao Kang, Zichong Li, Chen Liang, Weizhu Chen, and Tuo Zhao. Cosmos: A hybrid adaptive optimizer for memory-efficient training of llms, 2025b. URL https://arxiv.org/abs/2502.17410. 19

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *The Seventh International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7. 18

Kairong Luo, Zhenbo Sun, Haodong Wen, Xinyu Shi, Jiarui Cui, Chenyi Dang, Kaifeng Lyu, and Wenguang Chen. How learning rate decay wastes your best data in curriculum-based LLM pretraining. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=T5wkZJqzkz. 2

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, 2018. Association for Computational Linguistics. URL https://aclanthology.org/D18-1260. 6, 37

Konstantin Mishchenko and Aaron Defazio. Prodigy: An expeditiously adaptive parameter-free learner. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35779–35804. PMLR, 2024. URL https://proceedings.mlr.press/v235/mishchenko24a.html. 18

Antonio Orvieto and Robert M. Gower. In search of adam's secret sauce. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=CH72XyZs4y. 19

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf. 2, 6, 35

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. 6, 37

Noam Shazeer. Glu variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202. 6, 34

KaShun SHUM, Yuzhen Huang, Hongjian Zou, dingqi, YiXuan Liao, Xiaoxin Chen, Qian Liu, and Junxian He. Predictive data selection: The data that predicts is the data that teaches. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=tTVYR82Iz6. 19

Chongjie Si, Debing Zhang, and Wei Shen. Adamuon: Adaptive muon optimizer, 2025. URL https://arxiv.org/abs/2507.11005. 1, 6, 19, 37

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B. 7

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 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)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL https://aclanthology.org/N19-1421. 6, 37

Shohei Taniguchi, Keno Harada, Gouki Minegishi, Yuta Oshima, Seongcheol Jeong, Go Nagahara, Tomoshi Iiyama, Masahiro Suzuki, Yusuke Iwasawa, and Yutaka Matsuo. ADOPT: Modified adam can converge with any β_2 with the optimal rate. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=rzvVm0LsyK. 18

Kimi Team. Kimi k2: Open agentic intelligence, 2026. URL https://arxiv.org/abs/2507.20534. 1, 18

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, M. C., Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y. Charles, H. S. Che, Guanduo Chen, Guangyu Chen, Guanzheng Chen, Huarong Chen, Jia Chen, Jianlong Chen, Jun Chen, Kexin Chen, Peng Chen, Ruijue Chen, Wentao Chen, Xin Chen, Yang Chen, Yanru Chen, Yifei Chen, Yingjiang Chen, Yuankun Chen, Yujie Chen, Yutian Chen, Zhirong Chen, Dazhi Cheng, Yean Cheng, Jialei Cui, Jingbing Cui, Anqi Dai,

Jiaqi Deng, Hao Ding, Rui Ding, Shaofeng Ding, Mengfan Dong, Mengnan Dong, Yuhao Dong, Yuxin Dong, Angang Du, Chenzhuang Du, Dikang Du, Jusen Du, Yulun Du, Yu Fan, Jing Feng, Qiulin Feng, Yichen Feng, Kelin Fu, Qiang Fu, Fuxuan Gao, Hongcheng Gao, Jingyue Gao, Tong Gao, Weijia Gao, Shangyi Geng, Jie Gong, Linhu Gong, Shengao Gong, Xiaochen Gong, Qizheng Gu, Yicheng Gu, Shuhao Guan, Haiqing Guo, Shiqi Guo, Xiang Guo, Zhengyan Guo, Beixi Hao, Wenxin Hao, Xiaoru Hao, Dailan He, Haotian He, Lehan He, Qi He, Weiran He, Xinran He, Xinyi He, Yibo He, Yunjia He, Chao Hong, Tiange Hong, Hao Hu, Jiaxi Hu, Ruikun Hu, Weiming Hu, Yangyang Hu, Zhenxing Hu, Liang Hua, Jinbin Huang, Ke Huang, Ruiyuan Huang, Siying Huang, Weixiao Huang, Yan Huang, Zhengjie Huang, Zhiqi Huang, Yulong Hui, Chaobo Jia, Yutong Jiang, Zhejun Jiang, Zuoyou Jiang, Wenyi Jin, Xinyi Jin, Yu Jing, Huanjun Kong, Guokun Lai, Aidi Li, Cheng Li, Chengyuan Li, Cong Li, Fang Li, Guanyu Li, Haoyang Li, Jia Li, Junxiong Li, Lei Li, Letian Li, Lincan Li, Weihong Li, Wentao Li, Xintong Li, Yang Li, Yishen Li, Yiwei Li, Yuxiao Li, Zhaowei Li, Zhaoxi Li, Zheming Li, Zhengxiao Li, Zhiyuan Li, Jiawei Lin, Xiaohan Lin, Yibo Lin, Zichao Lin, Ziyan Lin, Bill Liu, Boxiao Liu, Chuan Liu, Liang Liu, Shaowei Liu, Shudong Liu, Shuran Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yanming Liu, Yibo Liu, Yipeng Liu, Zhengying Liu, Zhiheng Liu, Enzhe Lu, Haoyu Lu, Linqiang Lu, Tingzhan Lu, Zhiyuan Lu, Aotian Luo, G. Luo, Junyu Luo, Yifan Luo, B. Lyu, Wenzhou Lyu, Shaoguang Mao, Yuan Mei, Xin Men, Minqing Ni, Yixuan Niu, Siyuan Pan, Shujun Peng, Zhangyang Qi, Ruoyu Qin, ZeChao Qin, Zeyu Qin, Haiquan Qiu, Jianxin Qiu, Jiezhong Qiu, Bowen Qu, Yuhao Qu, Zeyu Shang, Youbo Shao, Han Shen, Jincheng Shi, Juanfeng Shi, Lidong Shi, Shengyuan Shi, Wingchun Siu, Pengwei Song, Xiaoxi Song, Jianlin Su, Yunfeng Su, Zhaochen Su, Lin Sui, Jingsong Sun, Junyao Sun, Shaoning Sun, Shuzhe Sun, Tongyu Sun, Yujun Sun, Yunpeng Tai, Chuning Tang, Heyi Tang, Sirui Tang, Zecheng Tang, Chaoran Tian, Rongpeng Tian, Yu Tian, Wei Tu, Chensi Wang, Chuang Wang, Chunjie Wang, Dinglu Wang, Feng Wang, Hailong Wang, Haiming Wang, Hao Wang, Hao Wang, Huaqing Wang, Hui Wang, Jiayi Wang, Jinglong Wang, Jinhong Wang, Jiuzheng Wang, Linian Wang, Shaobo Wang, Shenzhi Wang, Shuyi Wang, Si Wang, Siyuan Wang, Tianfu Wang, Wenjue Wang, Xingran Wang, Xinmei Wang, Xinyuan Wang, Xusheng Wang, Yalin Wang, Yangkun Wang, Yao Wang, Yaoyu Wang, Yejie Wang, Yiqin Wang, Yucheng Wang, Yuzhi Wang, Zhaoji Wang, Zhaowei Wang, Zhengtao Wang, Zhenhao Wang, Zhongsheng Wang, Zifan Wang, Chu Wei, Ming Wei, Shouxin Wei, Zichen Wen, Fan Wu, Haoning Wu, Rucong Wu, Wenhao Wu, Xiaoxue Wu, Yingcong Wu, Yongqi Wu, Yuxin Wu, Zijian Wu, Xinglang Xian, Chenxuan Xiang, Yuye Xiang, Bocheng Xiao, Chenjun Xiao, Xin Xiao, Jin Xie, Xiaotong Xie, Yifeng Xie, Zhe Xie, Bowei Xing, Yiming Xiong, Baosheng Xu, Boyu Xu, Jiale Xu, Jianfan Xu, Jing Xu, Jinjing Xu, L. H. Xu, Qingtao Xu, Shuyao Xu, Suting Xu, Tiantian Xu, Tianxiang Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ye Xu, Yueni Xu, Ziyao Xu, Haonan Xue, Junjie Yan, Yaoyao Yan, Fan Yang, Guangyao Yang, Hao Yang, Junwei Yang, Ruoyu Yang, Wenjie Yang, Xiaofei Yang, Xinyu Yang, Yi Yang, Yiling Yang, Ying Yang, Yuchen Yang, Zhen Yang, Zhilin Yang, Zian Yang, Zuhao Yang, Haotian Yao, Dan Ye, Haoran Ye, Wenjie Ye, Zhanbo Ye, Bohong Yin, Haoxiang Yin, Xietong Yin, Chengzhen Yu, Haozhen Yu, Longhui Yu, Shengnan Yu, Shuying Yu, Tianxiang Yu, Enming Yuan, Mengjie Yuan, Tongtian Yue, Wei Yue, Yang Yue, Dunyuan Zha, Haobing Zhan, B. H. Zhang, Dehao Zhang, Fei Zhang, Hao Zhang, Haoyuan Zhang, Huanyu Zhang, Jiapei Zhang, Jiaxuan Zhang, Jin Zhang, Kaiyi Zhang, Miaozhen Zhang, Puqi Zhang, Qinglei Zhang, Rong Zhang, Rui Zhang, Shaoshuai Zhang, Shiyi Zhang, Xiaobin Zhang, Xiaoyun Zhang, Y. Zhang, Yangkun Zhang, Ye Zhang, Yichi Zhang, Yikun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Zijing Zhang, Bin Zhao, Chenguang Zhao, Feifan Zhao, Jinglun Zhao, Jinxiang Zhao, Shuai Zhao, Wenshuo Zhao, Xiangyu Zhao, Xuanle Zhao, Yikai Zhao, Zijia Zhao, Haozhi Zheng, Huabin Zheng, Ruihan Zheng, Shaojie Zheng, Tengyang Zheng, Haofeng Zhong, Lei Zhong, Longguang Zhong, M. Zhou, Qiankang Zhou, Runjie Zhou, Ruozhang Zhou, Xinyu Zhou, Yiqiao Zhou, Zaida Zhou, Jinguo Zhu, Liya Zhu, Xinhao Zhu, Yangjunfeng Zhu, Yuxuan Zhu, Zhen Zhu, Chen Zhuang, Weiyu Zhuang, and Xinxing Zu. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653. 1

Kushal Tirumala, Daniel Simig, Armen Aghajanyan, and Ari S. Morcos. D4: Improving LLM pretraining via document de-duplication and diversification. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL https://openreview.net/forum?id=CG0L2PFrb1. 19

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: Improving and stabilizing shampoo using adam for language

modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf. 18

Shuche Wang, Fengzhuo Zhang, Jiaxiang Li, Cunxiao Du, Chao Du, Tianyu Pang, Zhuoran Yang, Mingyi Hong, and Vincent Y. F. Tan. Muon outperforms adam in tail-end associative memory learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=twbMFL0DMp. 1, 18

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 95266–95290. Curran Associates, Inc., 2024. doi: 10.52202/079017-3018. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ad236edc564f3e3156e1b2feafb99a24-Paper-Datasets_and_Benchmarks_Track.pdf. 6, 37

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://aclanthology.org/W17-4413. 6, 37

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them, 2025. URL https://arxiv.org/abs/2509.02046. 19

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. Qurating: Selecting high-quality data for training language models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=GLGYYqPwjy. 19

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=boSqwdvJVC. 19

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=uPSQv0leAu. 19

Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9508–9520, 2024. 18

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388. 6, 34

Shenghao Yang, Zhichao Wang, Oleg Balabanov, N. Benjamin Erichson, and Michael W. Mahoney. Prism: Distribution-free adaptive computation of matrix functions for accelerating neural network training, 2026. URL https://arxiv.org/abs/2601.22137. 18, 37

Zichun Yu, Spandan Das, and Chenyan Xiong. MATES: Model-aware data selection for efficient pretraining with data influence models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=6gzPSMUAz2. 19

Huizhuo Yuan, Yifeng Liu, Shuang Wu, zhou Xun, and Quanquan Gu. MARS: Unleashing the power of variance reduction for training large models. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=NrcKQ3ASLZ. 18

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. URL https://aclanthology.org/P19-1472. 6, 37

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf. 6, 34

Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Qiu Jiantao, Lei Cao, Ju Fan, Ye Yuan, Guoren Wang, and Conghui He. Harnessing diversity for important data selection in pretraining large language models. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=bMC1t7eLRc. 19

Jack Zhang, Noah Amsel, Berlin Chen, and Tri Dao. Gram newton-schulz, 2026. URL https://dao-ailab.github.io/blog/2026/gram-newton-schulz/. 1, 18

Minxin Zhang, Yuxuan Liu, and Hayden Schaeffer. Adagrad meets muon: Adaptive stepsizes for orthogonal updates, 2025b. URL https://arxiv.org/abs/2509.02981. 19

Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why transformers need adam: A hessian perspective. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=X6rqEpbnj3. 18

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. In *The Thirteenth International Conference on Learning Representations*, 2025c. URL https://openreview.net/forum?id=iBExhaU3Lc. 18

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham M. Kakade. Deconstructing what makes a good optimizer for autoregressive language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=zfeso8ceqr. 19

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2299–2314, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.findings-naacl.149/. 6, 37

Yunhua Zhou, Shuhao Xing, Junhao Huang, Xipeng Qiu, and Qipeng Guo. How to set the learning rate for large-scale pre-training?, 2026. URL https://arxiv.org/abs/2601.05049. 6, 24, 35

Xinlin Zhuang, Jiahui Peng, Ren Ma, Yinfan Wang, Tianyi Bai, Xingjian Wei, Qiu Jiantao, Chi Zhang, Ying Qian, and Conghui He. Meta-rater: A multi-dimensional data selection method for pre-training language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2025. 2, 7, 19

Appendix

Table of Contents

A Limitations **18**

B Related Work **18**

C Details of AMO **19**
C.1 Algorithm 20
C.2 Optimality Analysis 21

D Pilot Analysis **23**
D.1 Data Curriculum 23
D.2 Model Geometry 24

E Cost Analysis **31**
E.1 Theoretical Complexity 31
E.2 Practical Cost 33

F Experiments **34**
F.1 Experimental Settings 34
F.2 Multi-seed Experiments 38
F.3 Large-scale Experiments 38
F.4 Main Experiments 39
F.5 Prolonged Pre-training Experiments 39
F.6 Continual Pre-training Experiments 39
F.7 Ablation Experiments 39
F.8 Budget Control Experiments 40
F.9 Static Coefficient Experiments 42

G Further Analysis **42**
G.1 Prolonged Pre-training 42
G.2 NS Step Allocation 42
G.3 Post-warmup Stability 44

A LIMITATIONS

Our study has three main limitations. First, AMO adopts an observe-then-commit strategy: the NS allocation is fixed after the observation phase and cannot respond to substantial geometry shifts later in training, which periodic re-observation could address. Second, although operator-type allocation outperforms the tested Type $\times$ Depth variant, it is not necessarily optimal, and other granularities or richer allocation signals may prove more effective. Finally, our experiments focus on dense Transformers up to 4B parameters. Extending AMO to other architectures, such as mixture-of-experts models, and to substantially larger-scale pre-training remains future work.

B RELATED WORK

Adam optimizer. Adam Kingma & Ba (2017) and its decoupled weight decay variant AdamW Loshchilov & Hutter (2019) have long served as the default optimizers for LLM pre-training. Zhang et al. (2024) provides a theoretical justification from a Hessian perspective, showing that the loss landscape of Transformers exhibits highly heterogeneous curvature across parameter dimensions, which necessitates the coordinate-wise adaptivity that Adam provides and that SGD fundamentally lacks. Building on this understanding, one line of work seeks to go beyond Adam's diagonal second-moment preconditioner by incorporating richer curvature information. Sophia (Liu et al., 2024) uses a lightweight diagonal Hessian estimator paired with element-wise clipping to achieve a 2$\times$ speedup over Adam in pre-training steps with negligible per-step overhead. SOAP (Vyas et al., 2025) establishes a formal equivalence between Shampoo and Adafactor in a rotated basis, and runs Adam in Shampoo's eigenbasis to combine the benefits of Kronecker-factored preconditioning with the simplicity of Adam. A second line of work improves the update rule itself. ADOPT (Taniguchi et al., 2024) identifies the correlation between the current gradient and the second-moment estimate as the root cause of Adam's theoretical non-convergence, and resolves it by excluding the current gradient from the second-moment computation and reordering the momentum and normalization steps, achieving the optimal $\mathcal{O}(1/\sqrt{T})$ rate with any β_2. Adan (Xie et al., 2024) integrates Nesterov momentum into the adaptive gradient framework, and MARS (Yuan et al., 2025) is the first to successfully bring variance reduction into large-scale training by reconciling preconditioned gradient methods with a scaled stochastic recursive momentum, consistently outperforming AdamW on GPT-2 pre-training. Finally, a practical but important direction aims at reducing the overhead of Adam-style optimizers. Adam-mini (Zhang et al., 2025c) exploits the block-diagonal Hessian structure to partition parameters into groups that share a single learning rate, substantially cutting optimizer memory while matching Adam's performance. Prodigy (Mishchenko & Defazio, 2024) takes a complementary approach by eliminating the need for learning rate tuning altogether through a D-adaptation mechanism that automatically estimates the optimal step size during training.

Muon optimizer. Muon Jordan et al. (2024) is a recently proposed optimizer designed for the hidden-layer parameters of neural networks. It orthogonalizes the momentum buffer via Newton-Schulz (NS) iteration, replacing the gradient's singular values with uniform ones so that the update utilizes all directions in parameter space equally. This design has demonstrated substantial speedups over AdamW in large-scale LLM pre-training (Team, 2026; GLM-5-Team, 2026; Liu et al., 2025a). Wang et al. (2026) provide a theoretical explanation for Muon's advantage from the perspective of associative memory learning. A central practical concern with Muon is the cost of the multi-step NS iteration process. Therefore, several works explore accelerating the NS iteration itself. Polar Express (Amsel et al., 2026) derives optimal polynomial coefficients for the matrix sign iteration, CANS (Grishina et al., 2026) replaces hand-tuned coefficients with Chebyshev-type polynomials for faster spectral convergence, Turbo-Muon (Boissin et al., 2025) preconditions the input matrix to concentrate its spectrum and thus reduce the required number of NS steps, and PRISM (Yang et al., 2026) introduces a distribution-free adaptive framework for general matrix function computation including orthogonalization. More recently, Gram Newton-Schulz (Zhang et al., 2026) exploits Gram-matrix structure for a hardware-aware implementation. A parallel line of work reduces or replaces the NS step altogether to lower communication and computation overhead. Dion (Ahn et al., 2025) substitutes NS with amortized power iteration and low-rank approximation, enabling faster updates without full-matrix reconstruction under FSDP and tensor parallelism. Drop-Muon (Gruntkowska et al., 2025) and MuonBP (Khaled et al., 2025) perform orthogonalization only periodically while provably maintaining convergence. NorMuon Li et al. (2025) replaces NS with column normalization to

achieve comparable performance at a fraction of the cost. Finally, because Muon's orthogonalization discards singular-value magnitudes, it forgoes the coordinate-wise or curvature-aware adaptivity that Adam provides. Several methods address this by hybridizing orthogonal updates with adaptive mechanisms. AdaMuon (Si et al., 2025) reintroduces element-wise adaptive learning rates in Adam into the Muon framework, and Zhang et al. (2025b) equips orthogonal updates with an AdaGrad-Norm step size. COSMOS (Liu et al., 2025b) fuses ideas from SOAP and Muon by running Adam in Muon's eigenbasis for memory-efficient hybrid preconditioning. ASGO (An et al., 2025) designs a single-sided adaptive preconditioner that exploits the low-rank gradient and block-diagonal Hessian structure observed in Transformers, and ROOT (He et al., 2025) augments orthogonalized updates with robustness mechanisms to improve training stability.

Benchmarking and understanding optimizers. Alongside the development of new optimizers, a parallel line of work focuses on systematically benchmarking, deconstructing, and understanding what makes an optimizer effective for LLM pre-training. Zhao et al. (2025) deconstructs the design choices behind optimizers for autoregressive language models, identifying the key components, such as sign-based updates and momentum, that most strongly influence performance. In a similar spirit, Wen et al. (2025) conducts a comprehensive empirical study across a range of pre-training optimizers, characterizing the conditions under which different methods excel and offering practical guidelines for optimizer selection. In an investigation of orthogonalization-based methods, Frans et al. (2025) isolate key design variables in whitening optimizers. Their findings suggest that empirical improvement is attributable to only a few critical choices, particularly the application of momentum before the orthogonalization step. From a more theoretical perspective, Orvieto & Gower (2025) investigates the mechanisms underlying Adam's robustness, seeking to identify the secret sauce that makes it consistently competitive despite its simplicity. Similarly, AI et al. (2025) evaluates the practical wall-clock efficiency of Muon at scale, reporting that while Muon achieves faster convergence in terms of steps, its per-step overhead from NS iteration can offset gains depending on hardware and parallelism configuration.

Data selection for pre-training LLMs. Recently, a growing body of work has explored how to curate high-quality subsets from large-scale corpora for pre-training LLMs, and existing methods can be broadly organized into several categories. The first line of work performs *reference-model-based scoring*, using a pre-trained model to assign quality signals to individual documents. DSIR (Xie et al., 2023) selects examples whose distribution aligns with a curated target corpus via importance resampling, while Ankner et al. (2025) investigate perplexity-based pruning and find that small reference models can serve as effective proxies. Beyond single-score filtering, *multi-dimensional quality rating* methods assess data along multiple interpretable dimensions. QuRating (Wettig et al., 2024) scores documents on facets such as writing style and expertise, Meta-Rater (Zhuang et al., 2025) learns a joint multi-dimensional scoring function, and DataRater (Calian et al., 2025) meta-learns a curation policy that generalizes across tasks. A third category, *model-aware selection*, ties data quality to the evolving state of the model being trained. DsDm (Engstrom et al., 2024) estimates each example's marginal contribution via datamodels (Ilyas et al., 2022), MATES (Yu et al., 2024) periodically updates lightweight influence models during pre-training, Gu et al. (2025) dynamically adjusts the training distribution as a sequential decision problem, and SHUM et al. (2025) identifies the most instructive examples by their ability to predict learning outcomes. Complementing instance-level methods, several approaches address *corpus-level diversity*. D4 (Tirumala et al., 2023) combines document de-duplication with diversification, Wettig et al. (2025) constructs coherent domain clusters for balanced curation, and the diversity-aware framework of Zhang et al. (2025a) explicitly optimizes distributional coverage. Finally, *system-level strategies* utilize multiple signals within a unified pipeline. Bai et al. (2025) coordinates specialized selection actors to jointly curate the corpus.

C Details of AMO

This appendix fills in the PE-SIMULATE from Amsel et al. (2026) and our BUDGETALLOCATE operation invoked in Alg. 1. Also, we provide an optimality guarantee for BUDGETALLOCATE output under standard convexity assumptions. Sec. C.1 gives the full algorithm and implementation details, and Sec. C.2 states and proves the optimality theorem.

Algorithm 2 BUDGETALLOCATE

Require: projected error curves $\{E_i(k)\}_{i=1}^{|\mathcal{T}|}$, budget B, step range $[k_{\min}, k_{\max}]$, baseline T_{base}

Optional relaxation: if $B < |\mathcal{T}| \cdot k_{\min}$, decrease $k_{\min}$ (floor at 1) until feasible; if $B > |\mathcal{T}| \cdot k_{\max}$, increase $k_{\max}$ until feasible.
Initialize $T_i \leftarrow \text{clip}(T_{\text{base}}, k_{\min}, k_{\max})$ for all i
***// Add* pass**
while $\sum_i T_i < B$ **do**
    $i^\star \leftarrow \arg\max_{i\,:\,T_i < k_{\max}} \big[E_i(T_i) - E_i(T_i + 1)\big]$
    $T_{i^\star} \leftarrow T_{i^\star} + 1$
***// Remove* pass**
while $\sum_i T_i > B$ **do**
    $i^\star \leftarrow \arg\min_{i\,:\,T_i > k_{\min}} \big[E_i(T_i - 1) - E_i(T_i)\big]$
    $T_{i^\star} \leftarrow T_{i^\star} - 1$
***// Transfer* pass**
while $\exists (i, j)$ with $T_i < k_{\max}$, $T_j > k_{\min}$, and $\big[E_i(T_i) - E_i(T_i + 1)\big] > \big[E_j(T_j - 1) - E_j(T_j)\big]$ **do**
    $T_i \leftarrow T_i + 1$; $T_j \leftarrow T_j - 1$
return $\{T_i^\star\} \leftarrow \{T_i\}$

C.1 ALGORITHM

PE-SIMULATE(ℓ, k) is a scalar simulation of the PE-composed NS trajectory: starting from an input-side ℓ, it applies k NS iterations with coefficients $\{(a_j, b_j, c_j)\}_{j=1}^{k}$ produced by the PE composer at (ℓ, k), and returns the resulting post-NS effective ℓ, which is close to 1 when orthogonalization succeeds (Amsel et al., 2026). BUDGETALLOCATE takes as input the seven projected error curves $\{E_i(k)\}_{i=1}^{|\mathcal{T}|}$ with $|\mathcal{T}| = 7$, where $E_i(k) = 1 - \text{PE-SIMULATE}(\ell_{\text{target}}^{(i)}, k)$ is the post-NS error for operator type i at step count k, together with the total budget B and the step range $[k_{\min}, k_{\max}]$, and returns a per-type allocation $\{T_i^\star\}_{i=1}^{|\mathcal{T}|}$ satisfying $\sum_i T_i^\star = B$ and $T_i^\star \in [k_{\min}, k_{\max}]$. Alg. 2 gives the full process of BUDGETALLOCATE: it initializes all types at T_{base} clipped to the range, then applies three passes: *add* until the budget is met, *remove* if over budget, and *transfer* to eliminate any remaining pairwise improvement. The *add* and *remove* passes alone cannot resolve cases where a lower-error reallocation exists without crossing the budget boundary; the *transfer* pass is what lets BUDGETALLOCATE terminate at a global rather than merely local optimum (Sec. C.2).

Two details are worth noting. The optional relaxation step is triggered only when the derived budget $B = r \cdot |\mathcal{T}| \cdot T_{\text{base}}$ is infeasible under $[k_{\min}, k_{\max}]$, which happens when r falls outside $[k_{\min}/T_{\text{base}},\ k_{\max}/T_{\text{base}}]$. In all experiments reported in this paper, $[k_{\min}, k_{\max}] = [3, 7]$, $T_{\text{base}} = 5$, and $r = 1.0$, so the derived budget $B = 35$ sits strictly inside the feasible range $[21, 49]$ and no relaxation is ever triggered. Computationally, BUDGETALLOCATE operates entirely on scalar projected errors and terminates within a few dozen iterations in all cases, so its runtime is **negligible** relative to a single training step.

Implementations of AMO. For AMO, all runs start from the uniform PE schedule with 5-step NS iteration for each of the seven Muon layer types (`attn_q`, `attn_k`, `attn_v`, `attn_o`, `mlp_gate`, `mlp_up`, `mlp_down`). The observation horizon is 1200 steps for Qwen3-0.6B and Llama-3.1-760M, and 3000 steps for Qwen3-1.7B and Llama-3.1-1.4B. We estimate $\sigma_{\min}$ with exact SVD in fp32, use budget ratio 1.0, and search PE schedules over NS step counts $[3, 7]$. Before allocation, the robust observed ℓ is shrunk toward the baseline by $\ell_{\text{target}} = 0.7\ell_{\text{obs}} + 0.3 \cdot 10^{-3}$. The resulting target schedules are reached through a 300-step transition duration and then locked for the remainder of training, with no post-lock refresh. All configurations of AMO are provided in Tab. 3 and details of final AMO NS schedules locked after transition are provided in Tab. 2.

Training loss dynamics. To gain deeper insights into the training dynamics, we present a comparison of the training loss curves of AMO and Muon during the pre-training of Qwen3-1.7B in Fig. 6. It can be clearly observed that AMO consistently attains a lower training loss than Muon from the transition stage onward, indicating that the adaptive update yields more effective optimization under the same training budget (with the default budget ratio set to $r = 1.0$).

Table 2: Details of final locked NS schedules of AMO under budget ratio $r = 1.0$. ℓ_{target} is the effective target value logged at transition planning and coefficients are recomputed with PE.

Model	Operator	NS Step T	ℓ_{target}	NS Coefficient $(a_k, b_k, c_k)_{k=1}^{T}$
Qwen3-0.6B	attn_q	5	1.36334×10^{-3}	(8.2244,-23.121,16.654); (4.0628,-2.8881,0.52601); (3.8427,-2.8118,0.52947); (3.0757,-2.2601,0.46810); (2.1522,-1.5322,0.40416)
	attn_k	6	3.02222×10^{-4}	(8.2612,-23.225,16.729); (4.1194,-2.9025,0.52399); (4.0687,-2.8896,0.52579); (3.8707,-2.8312,0.53163); (3.1377,-2.3064,0.47322); (2.1930,-1.5705,0.40824)
	attn_v	6	3.00606×10^{-4}	(8.2612,-23.225,16.729); (4.1194,-2.9025,0.52399); (4.0691,-2.8897,0.52577); (3.8723,-2.8324,0.53176); (3.1415,-2.3092,0.47353); (2.1956,-1.5730,0.40850)
	attn_o	5	1.48010×10^{-3}	(8.2204,-23.110,16.646); (4.0567,-2.8865,0.52624); (3.8145,-2.7922,0.52728); (3.0160,-2.2152,0.46314); (2.1167,-1.4983,0.40058)
	mlp_gate	4	4.51844×10^{-3}	(8.1170,-22.819,16.437); (3.9063,-2.8501,0.53278); (3.2316,-2.3758,0.48091); (2.2633,-1.6351,0.41517)
	mlp_up	5	3.92888×10^{-3}	(8.1369,-22.875,16.477); (3.9343,-2.8566,0.53143); (3.3254,-2.4443,0.48852); (2.3218,-1.6582,0.40255); (1.8981,-1.2755,0.37756)
	mlp_down	4	5.87911×10^{-3}	(8.0715,-22.692,16.345); (3.8286,-2.8020,0.52837); (3.0454,-2.2374,0.46559); (2.1338,-1.5147,0.40230)
Qwen3-1.7B	attn_q	6	3.00377×10^{-4}	(8.2613,-23.225,16.729); (4.1195,-2.9026,0.52399); (4.0691,-2.8897,0.52577); (3.8726,-2.8325,0.53178); (3.1420,-2.3096,0.47357); (2.1960,-1.5733,0.40854)
	attn_k	5	1.26601×10^{-3}	(8.2278,-23.131,16.661); (4.0679,-2.8893,0.52582); (3.8667,-2.8284,0.53132); (3.1287,-2.2996,0.47247); (2.1868,-1.5647,0.40762)
	attn_v	5	7.42367×10^{-4}	(8.2459,-23.182,16.698); (4.0956,-2.8964,0.52481); (3.9760,-2.8666,0.52953); (3.4770,-2.5538,0.50069); (2.5062,-1.8479,0.43834)
	attn_o	6	3.00361×10^{-4}	(8.2613,-23.225,16.729); (4.1195,-2.9026,0.52399); (4.0691,-2.8897,0.52577); (3.8726,-2.8325,0.53178); (3.1421,-2.3096,0.47357); (2.1960,-1.5734,0.40854)
	mlp_gate	4	2.05821×10^{-3}	(8.2005,-23.054,16.606); (4.0268,-2.8791,0.52740); (3.6822,-2.6995,0.51693); (2.7982,-2.0880,0.46496)
	mlp_up	5	1.73407×10^{-3}	(8.2116,-23.086,16.629); (4.0435,-2.8832,0.52674); (3.7549,-2.7506,0.52263); (2.8987,-2.1259,0.45330); (2.0567,-1.4398,0.39443)
	mlp_down	4	4.25243×10^{-3}	(8.1259,-22.845,16.455); (3.9189,-2.8530,0.53216); (3.2730,-2.4061,0.48427); (2.2978,-1.6663,0.41853)
Llama-3.1-760M	attn_q	6	3.00856×10^{-4}	(8.2612,-23.225,16.729); (4.1194,-2.9025,0.52399); (4.0690,-2.8896,0.52578); (3.8721,-2.8322,0.53174); (3.1409,-2.3087,0.47348); (2.1952,-1.5726,0.40846)
	attn_k	4	3.07460×10^{-3}	(8.1658,-22.957,16.536); (3.9758,-2.8665,0.52954); (3.4762,-2.5532,0.50063); (2.5052,-1.8471,0.43825)
	attn_v	5	1.70416×10^{-3}	(8.2127,-23.088,16.631); (4.0450,-2.8836,0.52668); (3.7618,-2.7554,0.52317); (2.9117,-2.1359,0.45439); (2.0627,-1.4458,0.39505)
	attn_o	6	3.00986×10^{-4}	(8.2612,-23.225,16.729); (4.1194,-2.9025,0.52399); (4.0690,-2.8896,0.52578); (3.8719,-2.8321,0.53173); (3.1406,-2.3085,0.47345); (2.1950,-1.5724,0.40844)
	mlp_gate	5	2.54507×10^{-3}	(8.1838,-23.007,16.572); (4.0022,-2.8730,0.52841); (3.5796,-2.6270,0.50884); (2.6139,-1.9020,0.42880); (1.9545,-1.3357,0.38367)
	mlp_up	5	1.97798×10^{-3}	(8.2033,-23.062,16.612); (4.0309,-2.8801,0.52724); (3.6999,-2.7119,0.51832); (2.8003,-2.0498,0.44494); (2.0150,-1.3981,0.39009)
	mlp_down	4	3.76471×10^{-3}	(8.1424,-22.891,16.488); (3.9422,-2.8585,0.53106); (3.3529,-2.4644,0.49074); (2.3714,-1.7317,0.42562)
Llama-3.1-1.4B	attn_q	6	3.00393×10^{-4}	(8.2613,-23.225,16.729); (4.1195,-2.9026,0.52399); (4.0691,-2.8897,0.52577); (3.8725,-2.8325,0.53178); (3.1420,-2.3095,0.47357); (2.1960,-1.5733,0.40854)
	attn_k	4	2.93098×10^{-3}	(8.1707,-22.970,16.546); (3.9829,-2.8683,0.52923); (3.5034,-2.5727,0.50280); (2.5386,-1.8753,0.44135)
	attn_v	5	1.20276×10^{-3}	(8.2300,-23.137,16.666); (4.0712,-2.8902,0.52570); (3.8824,-2.8394,0.53254); (3.1647,-2.3264,0.47544); (2.2121,-1.5883,0.41014)
	attn_o	6	3.00379×10^{-4}	(8.2613,-23.225,16.729); (4.1195,-2.9026,0.52399); (4.0691,-2.8897,0.52577); (3.8726,-2.8325,0.53178); (3.1420,-2.3096,0.47357); (2.1960,-1.5733,0.40854)
	mlp_gate	5	1.98534×10^{-3}	(8.2030,-23.061,16.611); (4.0306,-2.8800,0.52725); (3.6982,-2.7108,0.51819); (2.7975,-2.0476,0.44470); (2.0139,-1.3970,0.38998)
	mlp_up	5	1.58149×10^{-3}	(8.2169,-23.100,16.639); (4.0514,-2.8852,0.52644); (3.7904,-2.7754,0.52540); (2.9672,-2.1782,0.45906); (2.0903,-1.4727,0.39788)
	mlp_down	4	4.32035×10^{-3}	(8.1237,-22.838,16.450); (3.9156,-2.8522,0.53232); (3.2623,-2.3982,0.48340); (2.2886,-1.6581,0.41764)

C.2 Optimality Analysis

We analyze BudgetAllocate under four assumptions. Three are by construction or follow from established results: (A1) each T_i ranges over integers in $[k_{\min}, k_{\max}]$; (A2) each E_i is strictly decreasing on this range, as more NS iterations cannot increase the post-NS error under the Chebyshev-based bound that PE-composed polynomials inherit (Amsel et al., 2026; Grishina et al., 2026); and (A4) the budget satisfies $|\mathcal{T}| \cdot k_{\min} \leq B \leq |\mathcal{T}| \cdot k_{\max}$ under the (possibly relaxed) range. The remaining assumption is the only non-trivial one: (A3) each E_i is *discretely convex*, meaning the

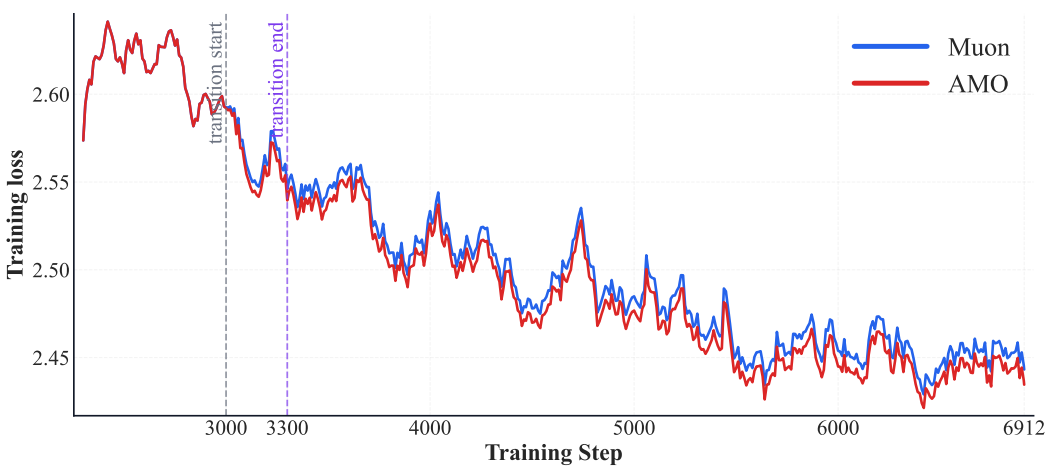


Figure 6: Training loss curves of AMO and Muon in pre-training Qwen3-1.7B.

Table 3: Full configurations of AMO for pre-training Llama3.1-760M/1.4B and Qwen3-0.6B/1.7B models.

Model	Observation Horizon & Interval & Sample Size	Shrinkage	Transition Duration
Qwen3-0.6B	(1200, 150, 8)	0.7	300
Qwen3-1.7B	(3000, 375, 8)	0.7	300
Llama-3.1-760M	(1200, 150, 8)	0.7	300
Llama-3.1-1.4B	(3000, 375, 8)	0.7	300

marginal gain $\Delta E_i(k) := E_i(k) - E_i(k+1)$ is non-increasing in k, or equivalently $E_i(k+1) + E_i(k-1) \geq 2E_i(k)$. Intuitively, the benefit of adding one more NS iteration diminishes as the step count grows.

Under these assumptions, the allocation problem is a separable minimization of discretely convex functions under an integer box constraint with one linear equality, a classical instance of separable discrete convex optimization, and BUDGETALLOCATE solves it to global optimality:

Theorem C.1 (Optimality of BUDGETALLOCATE)**.** *Under (A1)–(A4), the allocation* $\{T_i^\star\}$ *returned by Alg. 2 satisfies*

$$\{T_i^\star\} \;\in\; \arg\min_{\{T_i\}} \sum_{i=1}^{|\mathcal{T}|} E_i(T_i), \quad \textit{subject to} \quad \sum_i T_i = B,\;\; T_i \in [k_{\min}, k_{\max}].$$

Proof of Theorem C.1. We first establish a necessary and sufficient optimality condition, then show that BUDGETALLOCATE terminates at a point satisfying it.

Optimality condition. We claim that a feasible allocation $\{T_i\}$ minimizes $\sum_i E_i(T_i)$ subject to $\sum_i T_i = B$ and $T_i \in [k_{\min}, k_{\max}]$ *if and only if*

$$\Delta E_i(T_i) \;\leq\; \Delta E_j(T_j - 1) \qquad \text{for all } (i,j) \text{ with } T_i < k_{\max} \text{ and } T_j > k_{\min}. \tag{$\star$}$$

The "only if" direction is immediate. If $(\star)$ fails at some pair (i,j), then $\Delta E_i(T_i) > \Delta E_j(T_j - 1)$, and the transfer $(T_i, T_j) \to (T_i + 1, T_j - 1)$ is feasible (since $T_i < k_{\max}$ and $T_j > k_{\min}$) and changes the objective by

$$\big[E_i(T_i+1) - E_i(T_i)\big] + \big[E_j(T_j - 1) - E_j(T_j)\big] = -\Delta E_i(T_i) + \Delta E_j(T_j - 1) < 0,$$

strictly decreasing the objective and contradicting optimality.

The "if" direction uses A3. Suppose $(\star)$ holds at $\{T_i\}$ but there is a strictly better feasible $\{T_i'\}$, so $\sum_i E_i(T_i') < \sum_i E_i(T_i)$. Let $I^+ = \{i : T_i' > T_i\}$ and $I^- = \{i : T_i' < T_i\}$. Since both $\sum_i T_i = B$

and $\sum_i T'_i = B$, we have $\sum_{i\in I^+}(T'_i - T_i) = \sum_{i\in I^-}(T_i - T'_i) =: L$. We can therefore construct a sequence of L single-unit transfers that transforms $\{T_i\}$ into $\{T'_i\}$: at each step, pick any $i \in I^+$ with the current T_i below its target T'_i and any $j \in I^-$ with the current T_j above its target, and perform $(T_i, T_j) \to (T_i + 1, T_j - 1)$. Every intermediate allocation in this sequence remains in $[k_{\min}, k_{\max}]^{|\mathcal{T}|}$ and has total B, so each transfer is feasible.

We now track the objective change along this path. Consider the ℓ-th transfer, applied to a current allocation $\{T_i^{(\ell)}\}$ with the chosen pair (i_ℓ, j_ℓ). The transfer changes the objective by $-\Delta E_{i_\ell}(T_{i_\ell}^{(\ell)}) + \Delta E_{j_\ell}(T_{j_\ell}^{(\ell)} - 1)$. At the *starting* allocation $\{T_i\}$, condition $(\star)$ gives $\Delta E_{i_\ell}(T_{i_\ell}) \le \Delta E_{j_\ell}(T_{j_\ell} - 1)$. Along the path, T_{i_ℓ} only increases (since $i_\ell \in I^+$) and T_{j_ℓ} only decreases (since $j_\ell \in I^-$), so $T_{i_\ell}^{(\ell)} \ge T_{i_\ell}$ and $T_{j_\ell}^{(\ell)} - 1 \le T_{j_\ell} - 1$. By A3 (marginal gain non-increasing in k),

$$\Delta E_{i_\ell}(T_{i_\ell}^{(\ell)}) \le \Delta E_{i_\ell}(T_{i_\ell}), \qquad \Delta E_{j_\ell}(T_{j_\ell}^{(\ell)} - 1) \ge \Delta E_{j_\ell}(T_{j_\ell} - 1),$$

which combined with $(\star)$ yields

$$\Delta E_{i_\ell}(T_{i_\ell}^{(\ell)}) \le \Delta E_{i_\ell}(T_{i_\ell}) \le \Delta E_{j_\ell}(T_{j_\ell} - 1) \le \Delta E_{j_\ell}(T_{j_\ell}^{(\ell)} - 1).$$

Hence the objective change at the ℓ-th transfer is non-negative:

$$-\Delta E_{i_\ell}(T_{i_\ell}^{(\ell)}) + \Delta E_{j_\ell}(T_{j_\ell}^{(\ell)} - 1) \ge 0.$$

Summing over $\ell = 1, \dots, L$, the total objective change from $\{T_i\}$ to $\{T'_i\}$ is non-negative, i.e. $\sum_i E_i(T'_i) \ge \sum_i E_i(T_i)$, contradicting the assumption $\sum_i E_i(T'_i) < \sum_i E_i(T_i)$. This proves the "if" direction, so $(\star)$ characterizes global optimality.

BudgetAllocate terminates at a point satisfying $(\star)$. It remains to show that the output of Alg. 2 satisfies $(\star)$.

First, the add and remove passes terminate and ensure $\sum_i T_i = B$ at their end. If initialization already satisfies $\sum_i T_i = B$, both while-loops exit immediately. Otherwise exactly one is entered; the add pass (the remove case is symmetric) executes at most $B - \sum_i T_i$ iterations, each increasing $\sum_i T_i$ by 1, after which $\sum_i T_i = B$ and the loop exits. Because each step picks $\arg\max_i [E_i(T_i) - E_i(T_i + 1)]$ among types with $T_i < k_{\max}$, these passes respect the box constraint.

The transfer pass then strictly decreases $\sum_i E_i(T_i)$ at every iteration, by the loop condition $\Delta E_i(T_i) > \Delta E_j(T_j - 1)$ combined with the identity $-\Delta E_i(T_i) + \Delta E_j(T_j - 1) < 0$ derived above. Since each E_i takes finitely many values on $\{k_{\min}, \dots, k_{\max}\}$, the objective $\sum_i E_i(T_i)$ also takes finitely many values, so strict decrease forces termination in finitely many iterations. At termination, no pair (i, j) with $T_i < k_{\max}$ and $T_j > k_{\min}$ satisfies $\Delta E_i(T_i) > \Delta E_j(T_j - 1)$, which is exactly $(\star)$.

Combining: at the output of BUDGETALLOCATE, $\sum_i T_i = B$, $T_i \in [k_{\min}, k_{\max}]$, and $(\star)$ holds, so the output is a global minimizer of $\sum_i E_i(T_i)$ under the stated constraints. □

D PILOT ANALYSIS

In these two pilot experiments, we select a total of eight models for observation. Full details of model architecture, datasets, and training hyperparameters are provided in App. F.1.

D.1 DATA CURRICULUM

Setup. We conduct a factorial study over two base models, **Llama3.1-760M** and **Qwen3-0.6B**, two optimizers (Muon and AdamW), four learning-rate schedulers, and three data orders. All runs use the same `FineWeb-Edu` dataset, packed training sequences of length 2048, bf16 mixed precision, gradient clipping at 1.0, and seed 42. Total training tokens follow 1x Chinchilla optimal ratio of 20.

The data curriculum is implemented as a fixed-pool reordering scheme. The `uniform` setting applies a seeded shuffle to the pool. The `ascend` and `descend` settings sort examples by the *Educational Value* score field in increasing or decreasing order, respectively. Therefore, all curriculum variants see the same data pool and differ only in the **presentation order** of examples.

Table 4: LR scheduler configurations used in data curriculum. WS denotes warmup-stable, WSD denotes warmup-stable-decay, and Cosine denotes warmup and cosine decay. The same scheduler settings are used for both Muon and AdamW within each model.

Model	**Total Steps**	η_{stable}	**WS** (w, s)	**WSD** (w, s, d)	**Cosine** (w, d)	**Constant**
Llama3.1-760M	3622	1.573×10^{-3}	$(181, 3441)$	$(181, 3079, 362)$	$(193, 3866)$	3.0×10^{-4}
Qwen3-0.6B	2860	1.801×10^{-3}	$(143, 2717)$	$(143, 2431, 286)$	$(143, 2860)$	1.8×10^{-3}

For learning-rate control, we compare four scheduler families. We first define the reference stable learning rate by the empirical rule from Zhou et al. (2026)

$$\eta_{\text{stable}} = 10^{-4} \cdot 38.4588 \cdot N^{-0.2219} \cdot D^{-0.3509},$$

where N and D denote model size and reference training tokens in billions. This gives $\eta_{\text{stable}} = 1.573 \times 10^{-3}$ for Llama3.1-760M and $\eta_{\text{stable}} = 1.801 \times 10^{-3}$ for Qwen3-0.6B.

As listed in Tab. 4, we consider four LR schedulers. **WS (Warmup then Stable)** uses linear warmup to η_{stable} followed by a constant plateau. **WSD (Warmup, Stable, then Decay)** uses linear warmup, a stable phase at η_{stable}, and then a linear decay to $0.1\,\eta_{\text{stable}}$. **Cosine** uses linear warmup followed by cosine decay toward $0.1\,\eta_{\text{stable}}$. **Constant** keeps learning rate fixed throughout training.

For Qwen3-0.6B, WS uses 143 warmup steps and 2717 stable steps; WSD uses 143/2431/286 warmup/stable/decay steps; Cosine uses 2860 total steps with 143 warmup steps; and the Constant baseline uses a fixed learning rate of 1.8×10^{-3}, effectively matching η_{stable}.

For Llama3.1-760M, WS uses 181 warmup steps and 3441 stable steps; WSD uses 181/3079/362 warmup/stable/decay steps; Cosine is configured with 3866 total steps and 193 warmup steps, while the actual grid runs for 3622 optimizer steps, so the final learning rate remains slightly above the nominal cosine floor; and the Constant baseline uses a smaller fixed learning rate of 3.0×10^{-4}.

Optimizer hyperparameters are otherwise fixed within each model. AdamW uses $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and weight decay 0.1. Muon uses momentum 0.95, Nesterov momentum, five NS steps, and weight decay 0.1; non-matrix parameters are handled by AdamW.

Results. Full results are provided in Tabs. 5 and 6. Additionally, we pre-train Llama3.1-760M on another training pool, `Slimpajama`, using metric *dsir_books* under Muon, and similar performance trends are observed in Tab. 7.

D.2 Model Geometry

Setup. We track training trajectories of all models with geometry observation on all Muon-updated 2D matrices in each transformer block, namely seven types of matrices: `attn_q`, `attn_k`, `attn_v`, `attn_o`, `mlp_gate`, `mlp_up`, and `mlp_down`.

For a given parameter matrix, let G_t denote the stochastic gradient at step t, and let B_t be the momentum buffer updated as

$$B_t = \mu B_{t-1} + G_t,$$

where μ is the momentum coefficient. Muon applies NS orthogonalization not directly to G_t, but to the update-source matrix

$$M_t = \begin{cases} G_t + \mu B_t, & \text{with Nesterov momentum,} \\ B_t, & \text{otherwise.} \end{cases}$$

We record both the pre-orthogonalization matrix M_t and its post-NS counterpart M_t'. For both M_t and M_t', we compute singular-value-based geometry statistics. In particular, let

$$\sigma_1(M) \geq \sigma_2(M) \geq \cdots$$

denote the singular values of M. We define $\sigma_{\max}(M) = \sigma_1(M)$, and we define the smallest nontrivial singular value as

$$\sigma_{\min}^{(\tau)}(M) = \min\{\sigma_i(M) : \sigma_i(M) > \tau\},$$

Table 5: Data curriculum results of Llama3.1-760M with (a) Muon and (b) AdamW. Within each LR scheduler setting, the pattern `Uniform` > `Descend` > `Ascend` holds consistently. Abbreviations are as follows. AE: ARC-Easy; AC: ARC-Challenge; MM: MMLU; MM-P: MMLU-pro; HS: Hellaswag; OBQA: OpenBookQA; WG: WinoGrande; CSQA: CommonsenseQA; AGI: AGIEval-en.

(a) Muon

LR Scheduler	Order	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Const	Ascend	49.45	22.70	74.00	22.84	8.75	30.68	19.40	63.87	30.62	50.67	19.49	17.06	34.13
	Descend	53.96	21.42	76.40	24.40	7.73	30.10	21.80	64.15	29.67	50.43	19.57	17.11	34.73
	Uniform	56.90	23.21	75.40	23.00	7.26	30.49	20.60	64.58	28.23	51.54	19.66	17.26	**34.84**
WS	Ascend	55.26	22.61	76.50	22.97	5.35	33.61	21.00	66.21	31.10	50.91	18.84	16.90	35.11
	Descend	57.87	25.17	77.60	25.35	5.53	32.65	19.60	66.00	29.76	49.17	19.74	17.19	35.47
	Uniform	59.22	26.88	81.30	23.28	4.55	33.42	21.40	66.38	30.62	51.14	19.33	17.55	**36.26**
WSD	Ascend	55.93	24.57	79.40	24.33	6.84	33.63	21.60	67.30	31.20	51.30	18.51	17.96	36.05
	Descend	60.94	26.70	80.80	24.56	6.67	32.87	24.80	67.30	31.39	52.96	21.87	17.55	37.37
	Uniform	61.49	26.88	81.10	24.76	7.73	33.76	24.60	67.41	31.29	52.46	21.87	17.08	**37.54**
Cosine	Ascend	54.29	21.59	79.50	24.11	7.48	33.19	21.00	66.65	30.72	51.54	19.74	17.13	35.58
	Descend	59.34	25.09	79.90	23.41	6.67	31.96	20.60	66.10	31.00	52.49	20.07	16.41	36.09
	Uniform	59.60	25.68	82.70	25.49	7.42	33.00	22.00	66.27	29.86	51.30	19.25	17.32	**36.66**

(b) AdamW

LR Scheduler	Order	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Const	Ascend	48.32	20.73	69.90	23.02	6.77	28.90	16.00	61.92	28.33	52.33	19.57	17.60	32.78
	Descend	49.03	20.65	71.50	23.88	4.98	28.96	19.00	61.92	27.94	51.54	19.41	17.26	33.01
	Uniform	50.51	21.26	72.30	23.07	5.28	28.72	19.20	61.26	27.66	51.20	19.57	17.87	**33.16**
WS	Ascend	54.46	23.29	76.70	23.68	6.93	32.29	21.00	66.43	29.67	51.78	19.82	17.52	35.30
	Descend	57.95	23.46	80.30	25.16	6.13	32.29	19.60	66.10	29.86	52.25	20.07	17.26	35.87
	Uniform	58.25	26.02	79.80	23.69	8.28	32.32	21.60	66.16	29.86	52.09	19.74	16.95	**36.23**
WSD	Ascend	54.67	24.15	77.30	22.97	7.97	32.94	20.40	66.54	32.44	52.09	19.57	16.90	35.66
	Descend	60.31	24.66	80.60	26.58	6.18	32.82	20.60	66.27	30.05	52.33	20.15	17.03	36.47
	Uniform	59.39	25.68	80.80	24.63	6.46	32.78	24.40	66.92	29.19	51.78	19.57	17.65	**36.60**
Cosine	Ascend	53.24	23.29	77.60	25.30	7.72	32.36	19.80	65.83	29.57	51.46	19.90	17.21	35.27
	Descend	58.84	25.00	81.80	24.74	7.87	30.68	19.00	65.29	30.14	51.20	19.82	16.64	35.92
	Uniform	59.22	25.77	79.40	24.19	5.38	32.59	22.00	66.21	29.47	51.38	20.64	17.52	**36.15**

where τ is a very small numerical threshold used to discard singular values that are effectively zero up to numerical precision. In other words, $\sigma_{\min}^{(\tau)}$ is the smallest singular value that is still treated as numerically meaningful, rather than a near-zero mode caused by rank deficiency or floating-point noise. We then define the condition number as

$$\kappa(M) = \frac{\sigma_{\max}(M)}{\sigma_{\min}^{(\tau)}(M)}.$$

In addition, we record the spectral entropy and the effective rank, where the latter is defined as the minimum number of singular directions needed to explain 99% of the total spectral energy.

To measure how well NS orthogonalizes the update, we additionally compute the NS error of M_t' relative to the identity on its smaller dimension:

$$\epsilon_{\mathrm{NS}} = \begin{cases} \|M_t' M_t'^{\top} - I\|_F, & m \le n, \\ \|M_t'^{\top} M_t' - I\|_F, & m > n, \end{cases}$$

where $M_t' \in \mathbb{R}^{m \times n}$. Then we also report the shape-normalized residual

$$\tilde{\epsilon}_{\mathrm{NS}} = \epsilon_{\mathrm{NS}} / \sqrt{\min(m, n)},$$

to make residual magnitudes comparable across matrices of different sizes.

Table 6: Data curriculum results of Qwen3-0.6B with (a) Muon and (b) AdamW. Within each LR scheduler setting, the pattern `Uniform` > `Descend` > `Ascend` holds consistently. Abbreviations follow Table 5.

(a) Muon

LR Scheduler	Order	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Const	Ascend	56.57	23.12	81.10	23.58	6.92	34.52	21.60	68.01	31.20	53.35	20.39	17.29	36.47
	Descend	60.73	26.02	83.50	25.60	6.48	34.00	21.80	65.12	30.81	51.93	19.50	16.46	36.83
	Uniform	62.00	26.71	82.20	25.17	5.24	34.70	22.80	66.97	30.62	52.09	19.98	17.29	**37.15**
WS	Ascend	55.93	23.89	77.60	23.23	5.88	34.84	23.20	67.68	30.72	53.35	19.57	17.45	36.11
	Descend	61.74	26.54	81.60	23.61	5.31	33.38	22.00	66.05	31.20	52.01	19.74	17.58	36.73
	Uniform	59.09	27.39	82.60	25.24	5.47	34.48	23.20	66.54	32.15	52.72	20.07	17.11	**37.17**
WSD	Ascend	56.36	24.32	80.90	23.52	5.98	35.11	23.20	68.44	30.91	53.20	20.23	17.60	36.65
	Descend	61.78	26.62	83.10	24.63	5.82	34.47	22.40	66.49	30.53	51.93	19.57	17.55	37.07
	Uniform	61.74	26.45	82.00	24.73	5.22	34.97	23.60	67.90	31.96	51.22	20.31	17.55	**37.30**
Cosine	Ascend	54.38	22.61	79.60	24.31	6.56	34.37	20.60	67.36	30.72	51.54	19.16	17.68	35.74
	Descend	61.83	25.94	83.00	25.00	6.48	33.44	22.40	66.00	30.81	49.49	20.97	17.37	36.89
	Uniform	62.42	26.45	81.50	23.59	6.77	34.51	21.40	67.74	30.62	51.70	19.90	17.50	**37.01**

(b) AdamW

LR Scheduler	Order	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Const	Ascend	50.42	21.42	71.70	22.93	6.60	30.01	17.40	64.58	28.42	50.36	19.57	17.11	33.38
	Descend	53.91	22.70	76.20	22.90	6.07	29.90	18.60	63.00	29.09	49.01	19.66	16.87	33.99
	Uniform	52.74	22.44	75.10	24.18	5.41	29.88	20.00	63.92	30.43	49.80	19.66	17.42	**34.25**
WS	Ascend	50.51	21.42	76.10	23.04	6.77	31.76	19.60	64.80	28.80	49.96	19.82	17.52	34.18
	Descend	55.85	23.04	78.90	23.49	4.65	31.74	20.60	65.02	30.14	49.64	19.49	17.37	34.99
	Uniform	57.91	24.40	80.40	24.63	7.14	31.77	20.00	65.77	29.95	52.48	19.74	17.47	**35.97**
WSD	Ascend	52.44	23.72	75.90	23.07	7.00	32.43	20.20	66.10	29.38	51.78	19.57	17.94	34.96
	Descend	57.83	24.83	80.80	23.66	5.69	31.78	21.80	64.91	29.67	49.57	19.57	17.42	35.63
	Uniform	57.83	24.91	80.40	25.94	6.80	32.45	21.80	66.49	30.14	49.72	20.31	16.95	**36.15**
Cosine	Ascend	52.40	19.97	75.70	23.11	8.30	31.03	20.00	65.89	30.33	50.43	19.82	17.50	34.54
	Descend	56.82	22.78	78.60	24.21	6.62	30.55	21.20	63.98	28.90	51.46	19.41	17.26	35.15
	Uniform	55.98	23.98	79.30	23.74	7.82	31.31	19.60	64.31	30.91	50.20	19.00	17.45	**35.30**

Table 7: Data curriculum results of Llama3.1-760M. The pattern `Uniform` > `Descend` > `Ascend` holds consistently except under Cosine Scheduler.

LR Scheduler	Order	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Cosine	Ascend	43.01	21.25	73.90	22.97	7.84	29.60	16.60	60.88	30.53	51.62	19.57	17.11	32.91
	Descend	49.12	19.71	74.30	23.16	7.06	30.05	16.40	60.83	28.71	50.99	20.07	16.93	**33.11**
	Uniform	48.06	20.22	75.80	23.90	6.37	29.65	16.80	60.28	29.00	49.96	19.57	17.24	33.07
WS	Ascend	48.57	19.71	76.40	24.08	6.30	30.24	16.40	60.77	29.86	49.64	19.82	17.58	33.28
	Descend	47.10	20.82	72.80	24.36	6.42	30.32	17.80	61.10	29.86	51.62	19.98	17.29	33.29
	Uniform	48.40	22.27	77.20	23.02	7.43	30.35	18.80	61.59	31.10	49.96	20.64	16.85	**33.97**
WSD	Ascend	49.58	21.76	74.50	23.17	7.01	30.63	16.80	61.59	30.62	50.04	19.74	17.11	33.55
	Descend	50.67	21.42	76.00	24.85	5.09	30.28	17.20	61.21	29.57	50.36	19.82	17.58	33.67
	Uniform	51.26	23.38	78.50	24.04	6.77	30.26	18.80	62.79	29.57	50.43	20.88	17.24	**34.49**
Const	Ascend	45.83	18.69	71.30	23.40	7.66	28.66	15.80	59.79	27.37	50.83	19.66	16.85	32.15
	Descend	44.82	19.71	71.00	22.95	7.75	28.36	17.40	60.07	29.47	50.75	19.57	17.45	32.44
	Uniform	45.03	19.45	72.80	23.22	8.07	28.62	15.80	60.50	28.80	51.22	19.57	17.11	**32.52**

Throughout this analysis, we focus on $\kappa(M_t)$ and $\tilde{\epsilon}_{\mathrm{NS}}$, since these two quantities most directly reflect the conditioning of the Muon input and the realized quality of NS orthogonalization. All geometry runs use exact SVD metrics, effective-rank energy 0.99, and singular-value threshold $\tau = 10^{-10}$.

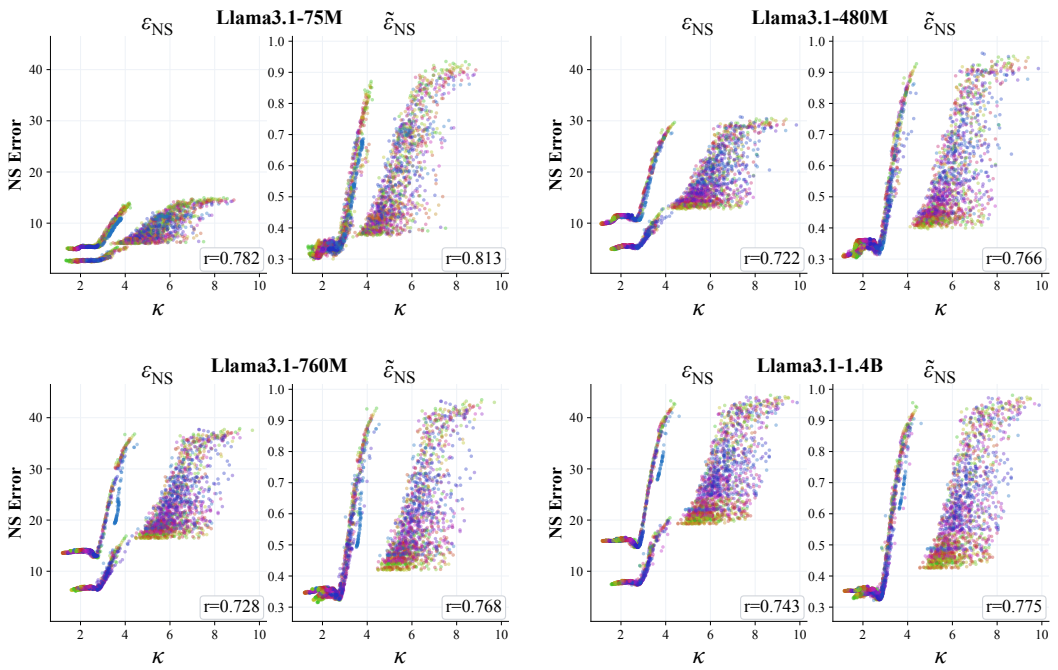


Figure 7: Condition number vs NS residual results for Llama3.1-family models. Colors indicate layer depth, so each point reflects the geometry of one parameter matrix at one training step, grouped visually by the layer it belongs to. κ is displayed in $\log_{10}$ scale. Across all scales, as indicated from Pearson Correlation coefficients (from 0.722 to 0.813), worse conditioning of the Muon update source is consistently associated with larger orthogonalization error, indicating a stable coupling between matrix geometry and NS convergence quality.

Geometry tracking is logged every 10 training steps for most models, and every 5 steps for the two smallest models (Llama3.1-75M and Qwen3-70M).

Llama3.1 Results. Figs. 7, 8, 9 and 10 summarize the geometry observations for the four Llama3.1 scales. As shown in Fig. 7, $\kappa(M_t)$ is positively correlated with the NS residual across all scales, indicating that poorly conditioned Muon update sources are *consistently harder* to orthogonalize. Fig. 8 further shows persistent operator-level heterogeneity: `mlp_up`, `mlp_gate`, and `attn_o` are frequently among the most geometry-sensitive matrix types, although no single operator dominates uniformly at every scale. From the layerwise snapshot profiles in Fig. 9, we observe that the depth profile of conditioning changes substantially over training, with geometry hotspots *shifting* across layers rather than remaining fixed. Finally, Fig. 10 shows that shallow and deep layers can *exchange* their relative ordering early in training, indicating that the geometry hierarchy over depth is itself dynamic. Taken together, these results suggest that a fixed, depth-independent NS budget is unlikely to be uniformly optimal throughout optimization.

Qwen3 Results. Figs. 11, 12, 13, and 14 show the corresponding geometry observations for the four Qwen3 scales. As in Llama3.1, Fig. 11 shows a clear *positive correlation* between $\kappa(M_t)$ and the NS residual across all scales, confirming that worse conditioning of the Muon update source is consistently associated with poorer orthogonalization quality. Compared with Llama3.1, however, Qwen3 exhibits stronger scale-to-scale heterogeneity. As shown in Fig. 12, the operator-type trajectories are often more concentrated, with `attn_v` more frequently emerging as a pronounced geometry-sensitive operator, especially through sharp early-training peaks. The layerwise snapshot profiles in Fig. 13 further show that geometry hotspots are *more localized* and can evolve more sharply over depth and training stage. Finally, Fig. 14 shows that shallow and deep layers again *exchange* their relative ordering, but in several Qwen3 models these crossings occur later or more abruptly than in Llama3.1. Taken together, these results reinforce the same overall picture as in Llama3.1 while also suggesting

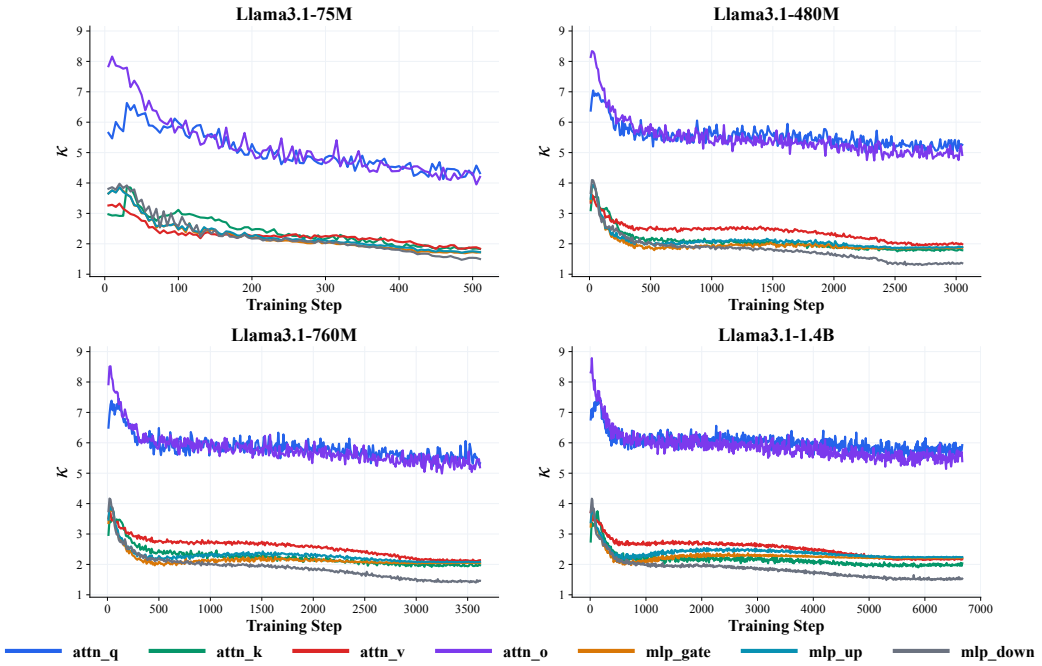


Figure 8: Evolution of median condition number by operator type across training for Llama3.1-family models. κ is displayed in $\log_{10}$ scale. The trajectories reveal strong operator-level heterogeneity: geometry difficulty is not uniformly distributed across matrix types, a small subset of operators often dominates the hardest regimes while `mlp_down` often shows easiest regimes.

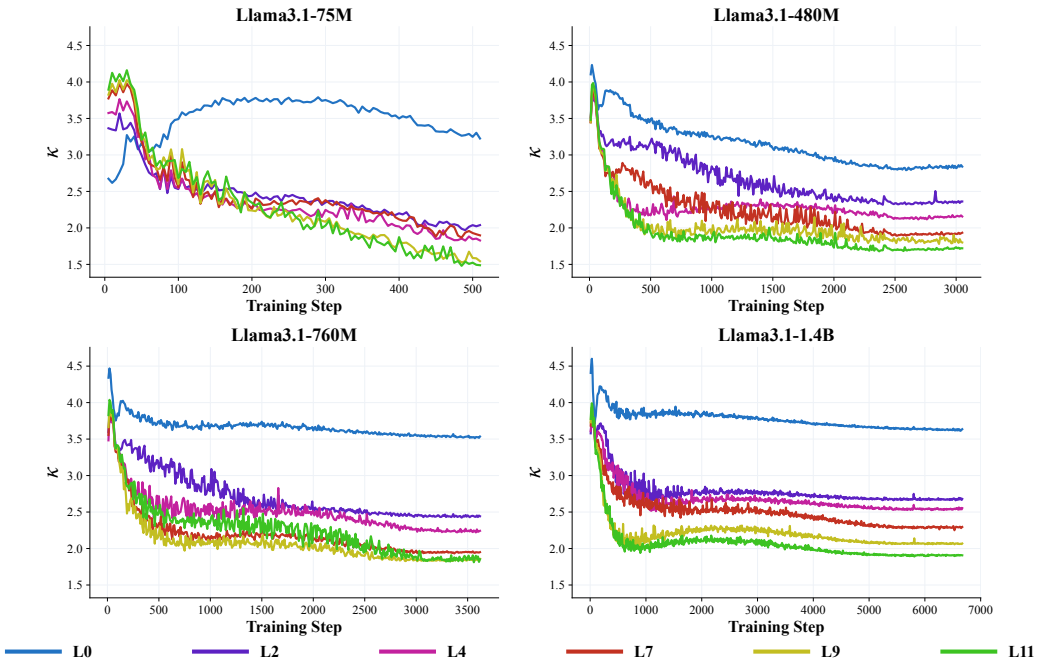


Figure 9: Layer-wise profiles of median condition number at representative training snapshots for Llama3.1-family models. κ is displayed in $\log_{10}$ scale. The depth dependence of conditioning changes substantially over training, showing that geometry hotspots are dynamically reorganized rather than fixed to a static set of layers.

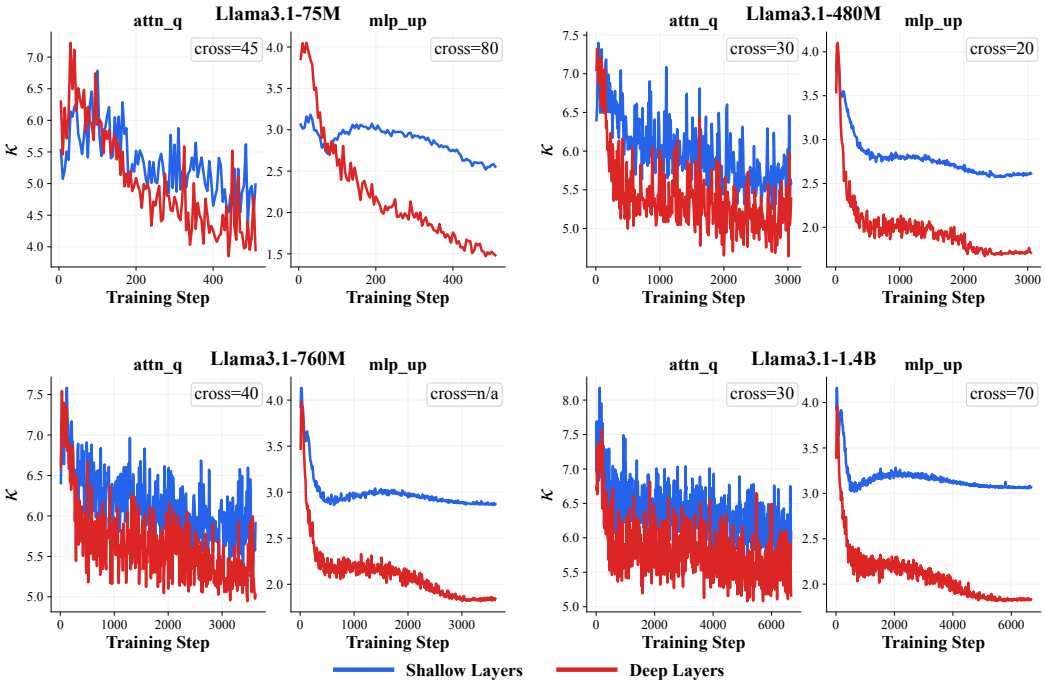


Figure 10: Shallow vs deep layer training trajectories of median condition number for two representative operator types, `attn_q` and `mlp_up`, across Llama3.1-family models. For each model, the shallow trajectory is computed from the **first three** layers, while the deep trajectory is computed from the **last three** layers. `cross` marks the first logged training step at which the shallow and deep trajectories intersect or reverse their relative ordering and `cross=n/a` indicates that no such crossing is observed over the logged training window. Across model scales, shallow and deep layers can exchange their relative ordering during training, indicating that the geometry hierarchy over depth is dynamic rather than fixed.

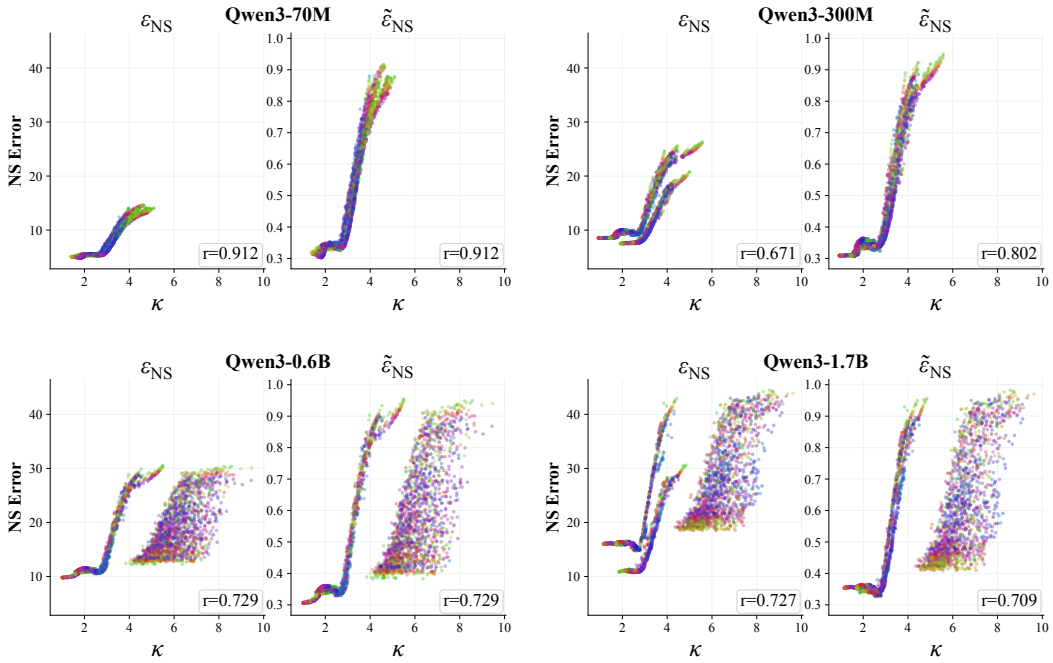


Figure 11: Condition number vs NS residual results for Qwen3-family models. Colors indicate layer depth, so each point reflects the geometry of one parameter matrix at one training step, grouped visually by the layer it belongs to. κ is displayed in $\log_{10}$ scale. Across all scales, as indicated from Pearson Correlation coefficients (from 0.671 to 0.912), worse conditioning of the Muon update source is consistently associated with larger orthogonalization error, indicating a stable coupling between matrix geometry and NS convergence quality.

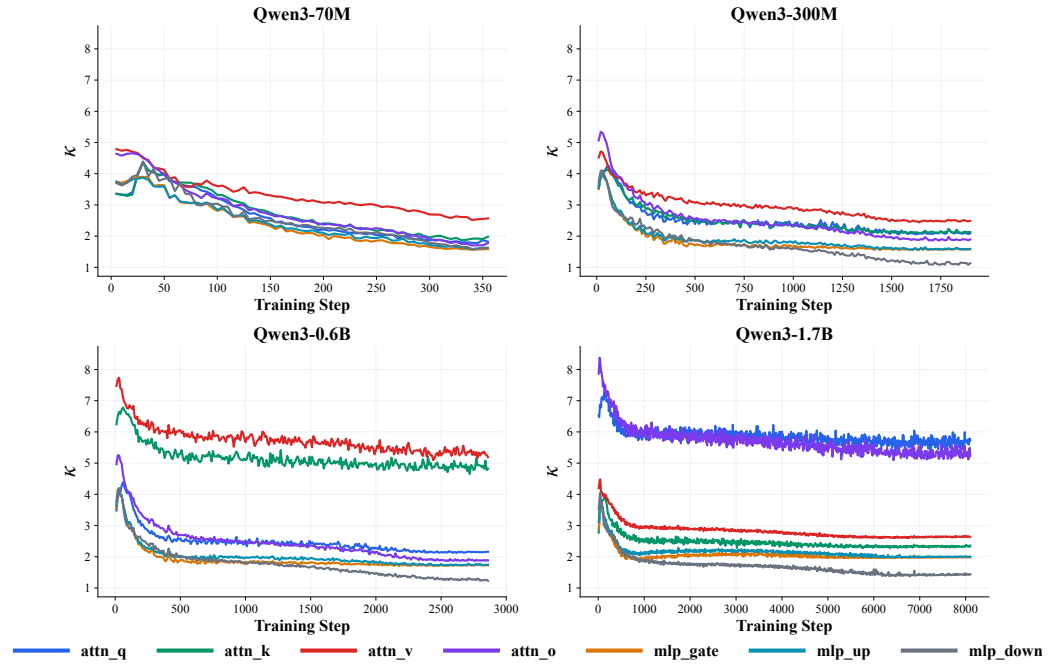


Figure 12: Evolution of median condition number by operator type across training for Qwen3-family models. κ is displayed in $\log_{10}$ scale. The trajectories reveal strong operator-level heterogeneity: geometry difficulty is not uniformly distributed across matrix types, a small subset of operators often dominates the hardest regimes while `mlp_down` often shows easiest regimes.

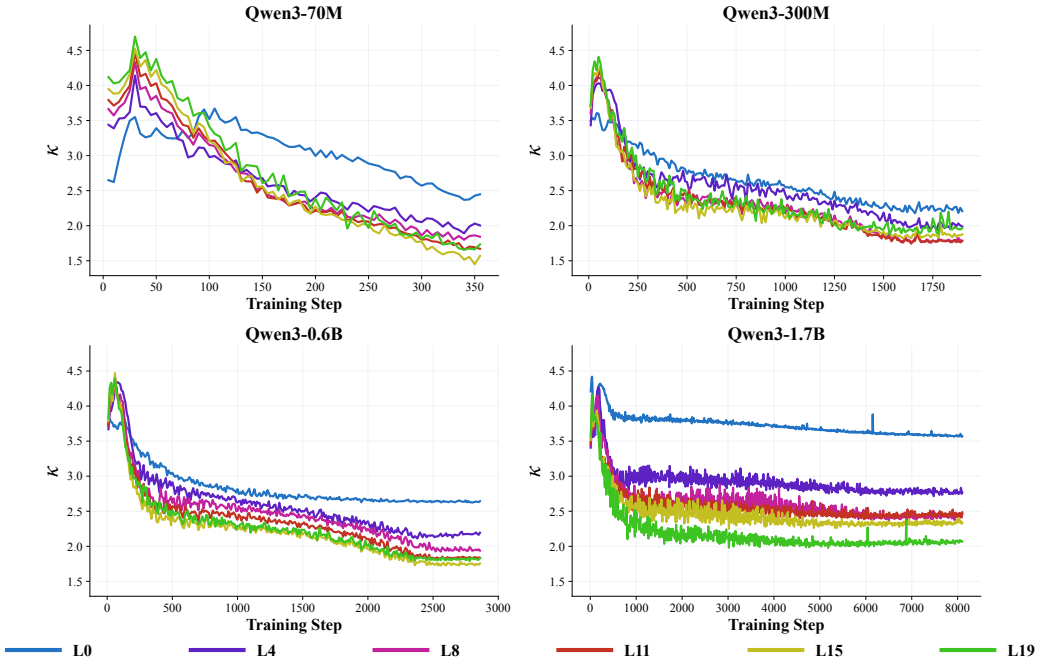


Figure 13: Layer-wise profiles of median condition number at representative training snapshots for Qwen3-family models. κ is displayed in $\log_{10}$ scale. The depth dependence of conditioning changes substantially over training, showing that geometry hotspots are dynamically reorganized rather than fixed to a static set of layers.

Table 8: Comparison of persistent optimizer-state memory and per-step update time complexity across optimizers. $P = P_\mu + P_{\text{adam}}$ denotes total trainable parameters, where P_μ are 2D matrices updated by the Muon-family rule and P_{adam} are the rest handled by AdamW. $C_{t,i} = \Theta\big(\min(m_{t,i}, n_{t,i})^2 \max(m_{t,i}, n_{t,i})\big)$ is the cost of one NS iteration on the i-th matrix of layer type t. k_t is the NS step count assigned to type t, and $B = \sum_t k_t^\star$ is AMO's global step budget.

Optimizer	Memory Cost	Per-step Time Cost	NS Budget
AdamW	$2P$	$\Theta(P)$	–
AdaMuon	$2P$	$\Theta\Big(P_{\text{adam}} + P_\mu + \sum_t k_t \sum_{i=1}^{N_t} C_{t,i}\Big)$	Uniform, $k_t \equiv 5$
Muon/Muon-PE	$P_\mu + 2P_{\text{adam}}$	$\Theta\Big(P_{\text{adam}} + \sum_t k_t \sum_{i=1}^{N_t} C_{t,i}\Big)$	Uniform, $k_t \equiv 5$
AMO (ours)	$P_\mu + 2P_{\text{adam}} + \mathcal{O}(\lvert\mathcal{T}\rvert)$	$\Theta\Big(P_{\text{adam}} + \sum_t k_t^\star \sum_{i=1}^{N_t} C_{t,i}\Big)$	Adaptive, $\sum_t k_t^\star = B$

that geometry dynamics remain family-dependent, with Qwen3 showing stronger operator- and scale-specific concentration.

E Cost Analysis

We provide a comprehensive cost analysis from both theoretical and practical perspectives.

E.1 Theoretical Complexity

We analyze algorithm-level cost in terms of **persistent optimizer-state memory complexity** and **per-step update time complexity**, intentionally separating these quantities from backend-dependent effects such as fused kernels or implementation-specific temporary allocations. A summarization of theoretical cost is provided in Tab. 8.

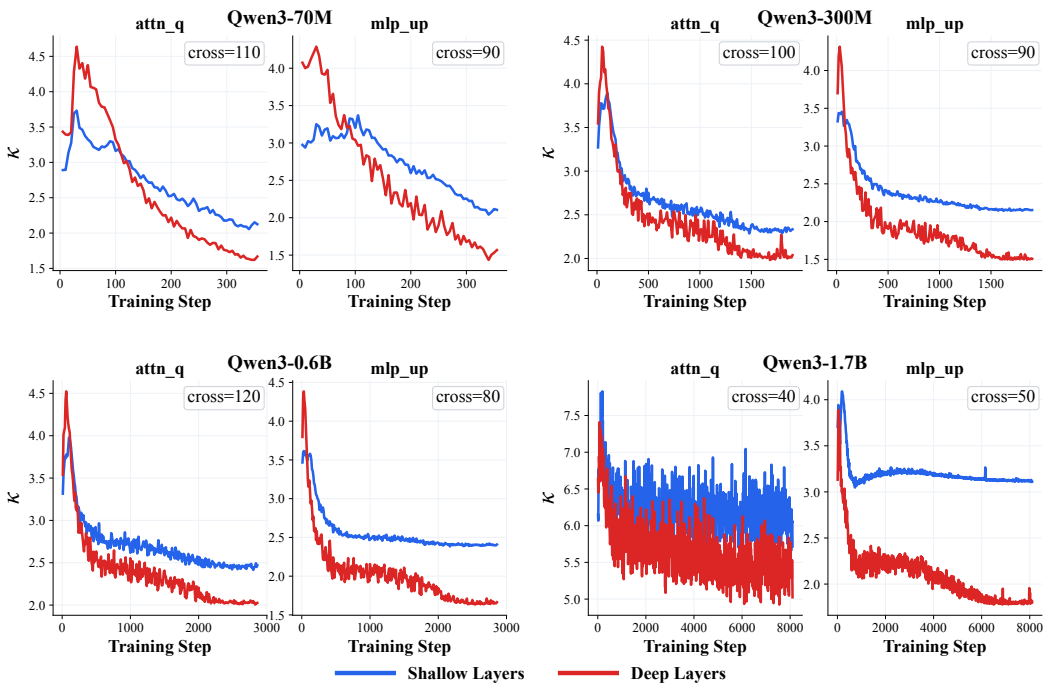


Figure 14: Shallow vs deep layer training trajectories of median condition number for two representative operator types, `attn_q` and `mlp_up`, across Qwen3-family models. For each model, the shallow trajectory is computed from the **first three** layers, while the deep trajectory is computed from the **last three** layers. `cross` marks the first logged training step at which the shallow and deep trajectories intersect or reverse their relative ordering and `cross=n/a` indicates that no such crossing is observed over the logged training window. Across model scales, shallow and deep layers can exchange their relative ordering during training, indicating that the geometry hierarchy over depth is dynamic rather than fixed.

Let the total number of trainable parameters be

$$P = P_\mu + P_{\text{adam}},$$

where P_μ denotes the parameters updated by the Muon-family rule (i.e., the 2D weight matrices selected for orthogonalized updates), and P_{adam} denotes the remaining parameters handled by AdamW (e.g., embeddings, output heads, and normalization parameters). We further partition P_μ into layer types $t \in \mathcal{T}$. For a given layer type t, let there be N_t matrices, and let the i-th matrix in this type have shape $m_{t,i} \times n_{t,i}$. One NS iteration on this matrix has dominant cost

$$C_{t,i} = \Theta\big(\min(m_{t,i}, n_{t,i})^2 \max(m_{t,i}, n_{t,i})\big).$$

Therefore, the true cost of a Muon-family update is a weighted sum over all matrices,

$$\sum_{t\in\mathcal{T}} k_t \sum_{i=1}^{N_t} C_{t,i},$$

where k_t is the NS step count assigned to layer type t. This notation is important because, although the optimizer shares a single step count within each layer type, matrices inside the same type need not have identical shapes in general.

From the **memory** perspective, standard AdamW maintains two momentum states for all parameters, giving

$$M_{\text{AdamW}} = 2P.$$

By contrast, Muon, Muon-PE, and our AMO all maintain only a single full-size momentum state on P_μ, while retaining two AdamW momentums on P_{adam}, hence

$$M_{\text{Muon/Muon-PE/AMO}} = P_\mu + 2P_{\text{adam}}.$$

Here, Muon-PE only replaces the NS coefficients, and therefore introduces no additional optimizer state. Likewise, AMO only stores a small amount of control metadata (e.g., observation history, target steps, and per-type override coefficients), whose size is $O(|\mathcal{T}|)$ and is negligible relative to parameter-sized states. AdaMuon additionally maintains a full-size second-momentum buffer, so

$$M_{\text{AdaMuon}} = 2P_\mu + 2P_{\text{adam}} = 2P,$$

which places it in the same memory class as AdamW.

Thus, at the level of persistent algorithmic memory, the methods fall into three groups: AdamW and AdaMuon at $2P$ and Muon/Muon-PE/AMO at $P_\mu + 2P_{\text{adam}}$.

From the **time** perspective, AdamW has linear per-step update cost,

$$T_{\text{AdamW}} = \Theta(P).$$

For Muon and Muon-PE, the dominant cost comes from applying k_t NS iterations to every 2D matrices in P_μ, giving

$$T_{\text{Muon/Muon-PE}} = \Theta\left(P_{\text{adam}} + \sum_{t\in\mathcal{T}} k_t \sum_{i=1}^{N_t} C_{t,i}\right).$$

For Muon or Muon-PE, $k_t \equiv 5$ for all t. Muon-PE stays in the same asymptotic class as Muon because its benefit comes from improved fixed coefficients, rather than a change in complexity order. AdaMuon adds extra amplitude normalization and second-moment updates on top of Muon, contributing only a linear term:

$$T_{\text{AdaMuon}} = \Theta\left(P_{\text{adam}} + \sum_{t\in\mathcal{T}} k_t \sum_{i=1}^{N_t} C_{t,i} + P_\mu\right).$$

Because the additional P_μ term is linear, the NS term remains dominant for large models and nontrivial step counts.

The key distinction of our AMO is that it does not use a uniform or purely static k_t. Instead, it allocates per-type step counts under a global budget constraint,

$$\sum_{t\in\mathcal{T}} k_t^\star = B,$$

and its locked-phase steady-state complexity becomes

$$T_{\text{AMO, lock}} = \Theta\left(P_{\text{adam}} + \sum_{t\in\mathcal{T}} k_t^\star \sum_{i=1}^{N_t} C_{t,i}\right).$$

This expression makes the core advantage of AMO explicit: fixing the total step budget B does not fix the true computation, because the aggregate weights $\sum_{i=1}^{N_t} C_{t,i}$ can differ substantially across layer types. Consequently, AMO can reduce the real update cost without reducing the nominal budget by transferring steps from expensive layer types to cheaper ones. In other words, AMO optimizes the weighted objective

$$\sum_{t\in\mathcal{T}} k_t \sum_{i=1}^{N_t} C_{t,i}$$

rather than merely the average NS depth. This is precisely why AMO should be viewed as a distinct complexity class relative to fixed-budget Muon variants: its gain comes from *budget-aware cross-type reallocation*.

E.2 Practical Cost

We next report implementation-level measurements of time and memory cost. For an apples-to-apples comparison, we train AdamW, Muon, AdaMuon, and AMO on **Qwen3-1.7B** for the same 6912 steps under an identical batch configuration. Full results are summarized in Tab. 9.

Table 9: Practical time and memory cost on pre-training **Qwen3-1.7B**, measured over 6912 training steps under an identical batch configuration. *Step Time* is end-to-end wall-clock per step; *Optimizer Update Time* is the time spent in the optimizer step; *Throughput* is tokens processed per second; *Peak memory* is the maximum allocated GPU memory during training. Arrows (↓/↑) indicate whether lower or higher is better. **Best** and second-best values are highlighted.

Optimizer	**Step Time** ↓ (s)	**Optimizer Update Time** ↓ (ms)	**Throughput** ↑ (M token/s)	**Peak Memory** ↓ (GB)
AdamW	**5.96**	**4.48**	**0.82**	63.75
Muon	6.14	195.81	0.80	**61.12**
AdaMuon	6.19	241.65	0.79	69.00
AMO (ours)	6.17	191.47	0.80	**61.12**

As shown in Tab. 9, AdamW attains the best raw step time, optimizer update cost, and throughput, which is expected given that its update rule is a pure element-wise operation. The Muon-family methods are roughly 43–$54\times$ more expensive in the optimizer update itself (from ~ 191 to ~ 242 ms versus AdamW's 4.48 ms), yet the end-to-end step time grows by only $\approx 3.0\%$–3.8%. This gap is small because the forward/backward pass continues to dominate the training loop, consistently accounting for roughly 5.82 s per step, while the Muon-family optimizer update stays within 0.19–0.24 s. In other words, the optimizer is measurably more expensive, but its cost is largely amortized by the much heavier non-optimizer portion of each step, making the overhead negligible in practice.

Within the Muon family, the ordering is also informative. AMO has the lowest optimizer-update cost among the Muon-family methods (191.47 ms), slightly below Muon (195.81 ms) and substantially below AdaMuon (241.65 ms), while avoiding Muon's uniform $k_t \equiv 5$ rule and instead using a budget-aware allocation. In terms of peak memory, Muon and AMO tie at the lowest value (61.12 GB), consistent with their shared persistent-state class $P_\mu + 2P_{\text{adam}}$; AdaMuon peaks higher at 69.00 GB, even above AdamW's 63.75 GB. Since both AdaMuon and AdamW lie in the same $2P$ persistent-state class theoretically, this reversal is best explained by transient allocations along the Muon-style update path rather than by any contradiction with the state-complexity analysis: the peak allocation reported in Tab. 1 is an implementation-level system quantity that includes temporary tensors and backend effects such as fused kernels, not a pure algorithmic-state metric.

In this run the budget ratio is set to 1.0, so AMO does not reduce the nominal total NS step budget; it instead redistributes the budget across layer types and locks to a schedule whose mean remains 5.0 NS steps across the seven layer types. The dominant runtime penalty of AMO does not arise from its locked-phase update rule, but from its low-frequency observation steps: the twenty observation steps average 15.63 s/step, of which 9.47 s is spent in the post-step exact-SVD observation probe, whereas non-observation steps average only about 6.14 s/step. Once the schedule is locked, AMO settles at 6.13 s/step and 0.80M tokens/s, essentially matching Muon's tail-region average of 6.14 s/step and 0.80M tokens/s. Consequently, in the present implementation the theoretical savings from AMO's lower locked-phase weighted orthogonalization cost are largely offset, at the wall-clock level of the full run, by the exact-SVD observation overhead and by the fact that the optimizer update accounts for only a small fraction of total step time.

F Experiments

F.1 Experimental Settings

Models. We consider two families of dense decoder-only Transformer language models: Qwen3 and Llama-3.1 (Yang et al., 2025; AI@Meta, 2024). Specifically, we use four Qwen3 variants (**70M**, **300M**, **0.6B**, and **1.7B**) and four Llama-3.1 variants (**75M**, **480M**, **760M**, and **1.4B**). Following official recommendations for model initialization, after instantiation from configuration, all linear and embedding weights are initialized from $\mathcal{N}(0, 0.02^2)$, linear biases are set to zero, and normalization parameters keep their default values. Tab. 10 summarizes their main architectural hyperparameters.

All models adopt grouped-query attention (Ainslie et al., 2023), RMSNorm (Zhang & Sennrich, 2019), and SwiGLU-style feed-forward blocks (Shazeer, 2020), but the two families differ systematically

Table 10: Architectural configurations of eight models used in our experiments. All models are dense decoder-only Transformers. L denotes the number of Transformer layers, d the hidden size, d_{ff} the intermediate size of the feed-forward block, and d_q/d the ratio between the query projection output dimension and the hidden size. We use a training context length of 2048 tokens for all models.

Model	Total Params	L	d	d_{ff}	Heads / KV Heads	d_q/d	Context Length
Qwen3-70M	70,368,512	20	256	1024	8 / 4	4.0	2048
Qwen3-300M	321,195,776	20	768	3072	12 / 4	2.0	2048
Qwen3-0.6B	596,049,920	28	1024	3072	16 / 8	2.0	2048
Qwen3-1.7B	1,720,574,976	28	2048	6144	16 / 8	1.0	2048
Llama-3.1-75M	74,717,440	12	256	768	8 / 2	1.0	2048
Llama-3.1-480M	480,805,888	16	1024	3584	8 / 2	1.0	2048
Llama-3.1-760M	762,091,008	12	1536	5376	12 / 3	1.0	2048
Llama-3.1-1.4B	1,397,819,392	16	2048	7168	16 / 4	1.0	2048

in several aspects. Qwen3 models use a vocabulary size of 151,936, rotary position embeddings with $\theta = 10^6$, and tied input/output embeddings in all scales considered here. In contrast, Llama-3.1 models use a vocabulary size of 128,256, the Llama-3 rotary scaling scheme with $\theta = 5 \times 10^5$, and untied output embeddings. Across comparable scales, the Qwen3 variants are generally deeper, whereas the Llama-3.1 variants are typically wider. Moreover, the smaller Qwen3 models employ expanded query projections with $d_q > d$, which leads to a noticeably different compute profile even at similar parameter scales.

Training Datasets. All pre-training runs use web documents sampled from the training set of `Fineweb-edu` (Penedo et al., 2024). We do not use a separate validation split during pre-training. During the training process, training data are streamed from disk and shuffled with a buffer size of 10,000 and a random seed of 42.

Tokenization is model-family-specific: Qwen3 models use the Qwen tokenizer, while Llama-3.1 models use the Llama-3.1 tokenizer. In both cases, text is tokenized without prepended BOS tokens, terminated with EOS, and packed into fixed-length sequences of 2048 tokens. Residual fragments shorter than the target sequence length are discarded. Because the two model families use different tokenizers, token counts are tokenizer-dependent; accordingly, we distinguish between the raw corpus identifier and the effective number of training tokens consumed by each run.

Pre-training. All models are trained from scratch using bfloat16 mixed precision and Flash-Attention-2 (Dao, 2024) on 6× to 32× NVIDIA H100 (80GB) GPUs. We use a fixed training sequence length of 2048 tokens for all runs, and apply gradient clipping with a maximum norm of 1.0. Unless otherwise specified, all runs use random seed 42 and do not employ validation during pre-training. Tab. 11 summarizes the pre-training hyperparameters for each run, including the effective global batch size, the per-device micro-batch size, the number of gradient accumulation steps, the total number of optimization steps, and the corresponding token budget.

Moreover, since Qwen3 and Llama-3.1 use different tokenizers, token budgets should be interpreted as tokenizer-specific token counts rather than a tokenizer-independent measure of raw corpus size. Across the runs considered here, the token budget ranges from roughly 1B to 34B tokens, increasing with model scale, following 1x Chinchilla optimal ratio (Hoffmann et al., 2022).

Optimizers. We consider two kinds of optimizers: AdamW and Muon-style variants. For both optimizers, the base learning rate is set according to the empirical scaling rule from (Zhou et al., 2026):

$$\eta_{\text{stable}}(N, D) = 10^{-4} \cdot 38.4588 \cdot N^{-0.2219} \cdot D^{-0.3509},$$

where N and D denote the model size and training token budget in billions, respectively.

For AdamW, we use $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, and weight decay 0.1. AdamW runs use a warmup-stable-decay (WSD) schedule: the learning rate is linearly warmed up to η_{stable}, kept constant during the stable phase, and then linearly decayed to $0.1\,\eta_{\text{stable}}$.

Table 11: Pre-training hyperparameters of all experiments. *Global Batch* denotes the global batch size, while *Micro Batch* denotes the per-device micro-batch size. *Grad Accum* denotes the number of gradient accumulation steps. *Token budget* follows 1x Chinchilla optimal ratio of 20 (Hoffmann et al., 2022) and is reported in billions (10^9) of packed training tokens.

Model	#GPUs	Global Batch	Micro Batch	Grad Accum	Total Steps	Token Budget
AdamW runs						
Qwen3-70M	8	2048	8	32	335	1.41
Qwen3-300M	16	2048	8	16	1430	5.98
Qwen3-0.6B	16	2048	8	16	2860	12.00
Qwen3-1.7B	32	2400	5	15	6912	33.97
Llama-3.1-75M	8	2048	8	32	360	1.51
Llama-3.1-480M	16	1536	12	8	3051	9.60
Llama-3.1-760M	16	2048	4	32	3622	15.20
Llama-3.1-1.4B	32	2048	8	8	6670	27.98
Muon runs						
Qwen3-70M	6	1728	12	24	398	1.41
Qwen3-300M	16	2048	8	16	1430	5.98
Qwen3-0.6B	16	2048	8	16	2860	12.00
Qwen3-1.7B	32	2400	5	15	6912	33.97
Llama-3.1-75M	6	1440	10	24	512	1.51
Llama-3.1-480M	16	1536	12	8	3051	9.60
Llama-3.1-760M	16	1920	12	10	3866	15.20
Llama-3.1-1.4B	32	2048	8	8	6670	27.98

Table 12: Muon/AdamW parameter split configurations in Muon runs. *Muon Ratio* is computed as the number of Muon-updated Parameters divided by total trainable parameters.

Model	Muon Params (M)	AdamW Params (M)	Muon Ratio (%)
Llama3.1-75M	9.04	65.67	12.10
Llama3.1-480M	218.10	262.70	45.36
Llama3.1-760M	368.05	394.04	48.29
Llama3.1-1.4B	872.42	525.40	62.41
Qwen3-70M	31.46	38.91	44.70
Qwen3-300M	204.47	116.72	63.66
Qwen3-0.6B	440.40	155.65	73.89
Qwen3-1.7B	1409.29	311.29	81.91

For all Muon-style variants, Muon is applied to 2D matrices in the Transformer blocks, while embedding layers, output heads (e.g., `lm_head` or `output`), and all remaining parameters such as biases and normalization weights are optimized with AdamW.

Across all Muon runs, we use momentum coefficient of 0.95, Nesterov momentum, weight decay 0.1, and five-step NS iterations with default KJ coefficients (Jordan et al., 2024). Concretely, Muon first forms a momentum-smoothed update, orthogonalizes it with NS iteration, and then applies a shape-dependent rescaling step size proportional to $0.2\sqrt{\max(m,n)}$ for an $m \times n$ parameter matrix, following Liu et al. (2025a). Muon runs use a cosine-decay schedule with linear warmup and a final learning-rate floor of $0.1\,\eta_{\text{stable}}$. Detailed configurations of Muon runs are provided in Tab. 12.

The details of other baseline optimizers are as follows:

- **Muon-You**[1]. This variant keeps the same parameter partition and outer hyperparameters as the default Muon setup, but replaces the NS coefficient sequence with the five-step `You` coefficients. No additional input rescaling or preconditioning is applied before the NS iteration.

[1]You's original implementation is provided at this page: https://gist.github.com/YouJiacheng/393c90cbdc23b09d5688815ba382288b

- **Muon-PE** (Amsel et al., 2026). This variant uses the five-step Polar Express coefficient table. In the current configurations, we additionally apply a small normalization safety factor of 1.01 before the NS iteration. All other settings follow the default Muon setup above.
- **Muon-CANS** (Grishina et al., 2026). This variant keeps the same Muon/AdamW parameter split and cosine-decay schedule, but replaces the NS coefficients with the CANS table. As in the PE baseline, we use a mild pre-NS normalization scale of 1.01 in the standard configurations.
- **Muon-Turbo** (Boissin et al., 2025). This variant uses the Turbo coefficient table together with AOL-style preconditioning. The remaining Muon and AdamW settings are unchanged.
- **AdaMuon** (Si et al., 2025). AdaMuon uses the same Muon/AdamW parameter split as above. For Muon-managed matrices, it maintains a momentum buffer with coefficient 0.95, takes the elementwise sign of the momentum-smoothed update, orthogonalizes it with a five-step KJ NS iteration, tracks an elementwise second-moment buffer with the same decay, divides by its square root plus 10^{-8}, and finally rescales the update to Frobenius norm $0.2\sqrt{mn}$ for an $m \times n$ matrix. Non-Muon parameters use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and weight decay 0.1. AdaMuon runs use the same cosine-decay schedule with linear warmup and a final floor of $0.1\,\eta_{\text{stable}}$.
- **ROOT** (He et al., 2025). ROOT replaces the global NS coefficient table with matrix shape-specific coefficients. Specifically, each Muon-managed matrix shape has a single fitted quintic triplet (a, b, c), obtained by minimizing $\text{mean}((g^T(\sigma) - 1)^2)$ for $g(x) = ax + bx^3 + cx^5$ on a mixed geometry-proxy and random-spectrum calibration set. The fitted triplet is repeated across all five NS iterations, with no additional pre-NS safety rescaling or AOL-style preconditioning.
- **PRISM** (Yang et al., 2026). We use the PRISM-5 Muon variant. After the momentum update and a pre-NS normalization scale of 1.01, each Muon-managed matrix is orthogonalized using three degree-five iterations with the fixed upper-bound parameter $\alpha = 29/20$, equivalently $(a, b, c) = (2.95, -3.4, 1.45)$ at every iteration. The parameter partition, AdamW fallback, weight decay, and learning-rate schedule remain identical to the default Muon setup.
- **Drop-Muon** (Gruntkowska et al., 2025). We partition the model in forward order into the input embedding, individual decoder layers, and a final block containing the output normalization and language-model head. At every optimizer step, a cutoff is sampled uniformly over these blocks and only the contiguous suffix from the cutoff to the output is trained; the preceding blocks are frozen before the forward pass, thereby omitting their backward computation and optimizer updates. The cutoff is resampled every step using seed 42, and this procedure is active throughout training.
- **NorMuon** (Li et al., 2025). NorMuon retains the same Muon/AdamW parameter partition. For each Muon-managed matrix, it forms an EMA momentum with $\beta_1 = 0.95$, applies Nesterov look-ahead, and orthogonalizes the resulting update using five KJ NS iterations. It then maintains a row-wise second-moment buffer with $\beta_2 = 0.95$, divides each row by $\sqrt{v + 10^{-10}}$, and rescales the result to restore the Frobenius norm of the orthogonalized update. The final update uses the standard Muon learning-rate scaling $0.2\eta\sqrt{\max(m, n)}$; non-Muon parameters use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and weight decay 0.1.

Evaluation. We evaluate all checkpoints using `lm-evaluation-harness` (Gao et al., 2024) framework on 12 downstream benchmarks: ARC-Easy, ARC-Challenge (Clark et al., 2018), SciQ (Welbl et al., 2017), MMLU (Hendrycks et al., 2021), MMLU-Pro (Wang et al., 2024), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), RACE (Lai et al., 2017), WinoGrande (Sakaguchi et al., 2020), CommonsenseQA (Talmor et al., 2019), and AGIEval-en (Zhong et al., 2024). We use the default task prompts and decoding settings provided by the harness, without additional prompt engineering, chat templates, or manual overrides of the few-shot configuration. Following the harness defaults, RACE uses the `high` subset and WinoGrande uses the `winogrande_xl` variant.

As summarized in Tab. 13, all tasks in our suite are evaluated in zero-shot mode except MMLU-Pro, which uses the task-defined 5-shot setting. In particular, MMLU is evaluated in zero-shot mode

Table 13: Evaluation benchmarks and task settings in our experiments. By default, the framework evaluates on the `test` split when available, and otherwise falls back to the `validation` split.

Benchmark	Framework Task	Evaluation Split	Few-shot	Evaluation Metric
ARC-Easy	`arc_easy`	Test	0	Accuracy
ARC-Challenge	`arc_challenge`	Test	0	Accuracy
SciQ	`sciq`	Test	0	Accuracy
MMLU	`mmlu`	Test	0	Accuracy
MMLU-Pro	`mmlu_pro`	Test	5	Exact Match
HellaSwag	`hellaswag`	Validation	0	Accuracy
OpenBookQA	`openbookqa`	Test	0	Accuracy
PIQA	`piqa`	Validation	0	Accuracy
RACE (high)	`race`	Test	0	Accuracy
WinoGrande (XL)	`winogrande`	Validation	0	Accuracy
CommonsenseQA	`commonsense_qa`	Validation	0	Accuracy
AGIEval-en	`agieval_en`	Test	0	Accuracy

Table 14: Full results of multi-seed experiments on pre-training Llama3.1-1.4B. Methods within each seed use the same model initialization and data order.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Seed=43													
Muon	66.12	30.20	84.10	25.13	6.47	37.64	24.00	68.99	32.54	52.17	20.07	16.54	38.66
Best Static	66.41	29.95	85.00	24.97	6.18	37.69	24.40	69.80	32.73	52.88	20.23	16.36	<u>38.88</u>
AMO (ours)	65.87	31.74	85.00	25.15	6.52	37.54	25.00	70.57	32.54	54.14	20.72	17.08	**39.32**
Seed=123													
Muon	64.86	30.38	85.40	24.94	6.48	37.14	24.00	69.70	32.54	54.06	19.74	17.89	<u>38.93</u>
Best Static	65.74	30.46	84.60	25.03	6.04	37.61	23.00	70.46	31.67	55.25	19.49	16.64	38.83
AMO (ours)	65.32	32.68	85.00	26.65	6.87	37.30	25.60	70.55	31.77	54.78	20.48	17.39	**39.53**
Seed=2026													
Muon	65.53	30.72	85.20	25.19	5.28	37.68	26.00	70.95	33.01	53.91	20.80	17.34	<u>39.30</u>
Best Static	65.66	30.29	84.00	25.07	6.87	37.80	24.20	68.82	32.82	55.80	19.25	16.64	38.94
AMO (ours)	66.46	31.20	85.40	24.96	7.30	37.14	26.00	70.29	32.92	54.46	21.29	17.19	**39.55**

in our setup. Most benchmarks are standard multiple-choice evaluations. For ARC-Easy, ARC-Challenge, SciQ, HellaSwag, OpenBookQA, and PIQA, the framework reports accuracy scores. MMLU is reported as size-weighted accuracy aggregated over four subject groups. MMLU-Pro is evaluated as a 14-domain aggregate with greedy generation and exact-match scoring after answer extraction. AGIEval-en is evaluated through aggregating tasks over 10 English subtasks. This suite is predominantly multiple choice but includes one generative math subtask, and the final score is reported as size-weighted accuracy.

F.2 Multi-seed Experiments

To evaluate the robustness of the observed improvements to random seeds, we repeat the main comparison on Llama3.1-1.4B using three random seeds. For each seed, Muon, the best static schedule, and AMO use the same model initialization and data order, enabling paired comparisons that reduce seed-level nuisance variation. The best static schedule corresponds to the `Floor-1e-2`.

As shown in Tab. 14, AMO outperforms Muon under all three seeds. Across paired runs, AMO improves over Muon by +0.50 $\pm$ 0.22 points on average and over the best static schedule by +0.58 $\pm$ 0.13 points.

F.3 Large-scale Experiments

We further evaluate whether the gains of AMO persist at a larger model scale by pre-training Llama3.1-4B for 100B tokens.

Table 15: Full results of large-scale experiments on pre-training Llama3.1-4B for 100B tokens.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Llama3.1-4B													
Muon	72.60	38.74	90.10	25.43	8.80	45.24	29.00	73.23	34.26	56.27	19.49	17.32	42.54
AMO (ours)	73.65	38.48	90.80	26.19	9.07	45.32	29.20	74.16	36.08	56.59	22.44	17.50	**43.29**

Table 16: Full results of main experiments on pre-training Llama3.1-760M/1.4B.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Llama3.1-760M													
AdamW	59.89	27.22	79.90	23.93	7.36	33.53	23.20	67.03	31.10	51.30	19.33	16.46	36.69
Muon-KJ	59.68	25.00	79.90	24.78	6.91	33.64	24.40	65.89	31.48	49.88	20.48	17.83	36.66
Muon-You	60.27	26.37	80.50	23.47	6.46	33.41	23.00	67.19	29.67	51.78	19.82	17.03	36.58
Muon-PE	59.89	27.39	81.80	23.79	8.15	33.88	22.80	67.68	30.62	51.38	19.66	17.65	<u>37.06</u>
Muon-CANS	59.51	25.34	81.40	26.30	6.44	33.23	22.80	66.81	29.76	50.83	20.31	17.76	36.71
Muon-Turbo	59.81	25.94	79.80	23.47	6.70	33.39	22.40	65.67	30.81	51.22	20.39	17.03	36.39
AdaMuon	58.29	25.34	80.60	24.32	7.58	32.77	23.00	67.30	31.10	51.30	19.25	17.91	36.56
ROOT	59.85	26.19	80.80	25.41	7.52	33.54	19.60	67.36	30.62	51.54	20.88	17.26	36.71
AMO (ours)	59.43	26.79	81.50	26.24	6.76	33.79	22.80	67.70	31.43	51.91	21.13	17.60	**37.26**
Llama3.1-1.4B													
AdamW	65.84	28.67	84.20	24.95	6.13	37.94	22.60	70.23	33.00	53.43	20.23	16.74	38.66
Muon-KJ	65.32	28.41	85.20	25.47	6.94	37.32	25.00	69.36	32.15	53.67	19.99	16.87	38.81
Muon-You	64.81	30.80	84.30	24.86	7.47	37.55	25.00	69.70	32.20	53.28	19.57	17.34	38.91
Muon-PE	66.41	30.03	84.80	23.81	7.53	37.54	24.00	69.15	32.34	53.67	20.64	17.24	<u>38.93</u>
Muon-CANS	65.57	31.74	82.20	25.00	7.75	37.83	24.20	69.75	32.63	52.57	20.64	17.00	38.91
Muon-Turbo	64.77	30.03	84.90	25.33	7.86	37.12	24.00	70.13	32.44	52.72	20.23	17.29	38.90
AdaMuon	63.17	29.52	83.20	25.23	6.50	36.19	26.00	68.61	31.77	52.25	21.29	17.50	38.44
NorMuon	65.28	29.44	84.70	24.04	7.09	37.86	23.40	69.48	31.48	54.93	19.98	16.82	38.71
Drop-Muon	58.71	26.02	79.70	26.21	8.04	32.05	21.60	66.54	30.33	50.43	20.88	17.26	36.48
PRISM	61.78	27.47	81.50	25.02	6.09	34.60	23.60	67.63	30.91	50.43	19.98	17.52	37.21
ROOT	65.74	29.62	83.90	25.91	7.50	37.38	24.00	69.96	31.39	53.12	20.23	16.98	38.81
AMO (ours)	66.46	31.20	85.40	25.60	7.99	37.46	25.60	70.50	33.00	53.43	21.29	18.30	**39.69**

As shown in Tab. 15, AMO improves the average downstream score from 42.54 to 43.29, yielding a gain of 0.75 percentage points over Muon. This result extends the observed advantage of AMO beyond the 2B regime and shows that its performance improvement persists at the 4B scale.

F.4 Main Experiments

Full results of pre-training Llama3.1-760M/1.4B and Qwen3-0.6B/1.7B using all optimizers in this paper are provided in Tabs. 16 and 17.

F.5 Prolonged Pre-training Experiments

Full results of prolonged pre-training are provided in Tab. 18.

F.6 Continual Pre-training Experiments

Full results of continual pre-training are provided in Tab. 19.

F.7 Ablation Experiments

Full results of ablation experiments on *Observation horizon and interval*, *Observation sample size*, *Shrinkage*, and *Transition duration* of AMO are provided in Tab. 20.

Table 17: Full results of main experiments on pre-training Qwen3-0.6B/1.7B.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Qwen3-0.6B													
AdamW	58.96	24.49	82.00	23.51	6.71	33.41	21.00	66.27	30.81	50.12	19.90	17.99	36.26
Muon-KJ	62.16	26.79	81.90	23.80	7.30	34.75	23.20	67.90	31.67	51.93	19.00	16.92	37.28
Muon-You	60.77	27.22	82.20	24.45	8.22	34.60	22.80	67.03	31.39	53.04	19.00	17.65	37.36
Muon-PE	61.53	26.79	83.50	24.01	6.09	34.75	22.20	67.85	31.96	52.17	20.48	16.64	37.33
Muon-CANS	62.33	26.19	80.30	24.09	7.95	35.04	22.20	66.92	32.06	52.72	20.39	17.00	37.27
Muon-Turbo	61.49	25.51	82.70	23.45	6.40	35.05	24.80	68.66	30.72	52.09	19.82	17.71	<u>37.37</u>
AdaMuon	58.16	24.91	80.80	24.02	6.10	33.33	22.20	66.92	31.96	52.49	20.48	17.39	36.56
ROOT	60.01	25.60	81.00	25.43	6.26	33.99	20.80	66.97	29.76	50.28	21.70	17.68	36.62
AMO (ours)	63.17	27.19	82.60	24.89	7.80	35.04	21.60	68.66	31.53	53.09	20.23	17.39	**37.77**
Qwen3-1.7B													
AdamW	68.81	30.34	87.20	24.70	6.60	40.56	25.00	71.76	33.30	55.64	18.67	16.46	39.92
Muon-KJ	68.22	31.74	85.70	24.80	5.54	40.71	26.40	72.09	33.78	55.00	20.72	16.45	40.10
Muon-You	67.59	30.20	86.90	24.35	6.87	40.47	25.40	72.52	34.07	56.12	19.25	17.89	40.14
Muon-PE	68.98	31.66	85.30	25.13	7.54	40.96	26.40	71.82	32.44	54.78	20.88	16.93	<u>40.24</u>
Muon-CANS	69.32	32.08	86.50	24.65	6.18	41.04	25.00	70.84	33.78	54.22	17.28	16.46	39.78
Muon-Turbo	67.17	31.74	86.20	24.50	6.81	39.93	24.40	71.55	33.01	53.75	20.80	16.51	39.70
AdaMuon	67.42	31.48	85.80	24.78	5.24	39.66	25.60	70.89	33.68	53.04	20.39	16.90	39.57
NorMuon	67.17	30.80	86.00	25.07	7.40	40.63	26.40	71.11	33.21	53.91	22.69	17.94	40.19
Drop-Muon	55.93	22.44	76.40	23.90	6.67	32.81	21.20	65.51	29.86	52.01	20.31	17.47	35.38
PRISM	64.10	27.82	82.60	24.48	6.54	36.63	24.40	69.86	30.53	51.46	20.48	17.99	38.07
ROOT	69.95	31.74	87.70	25.25	7.68	40.53	25.80	71.76	31.77	54.99	18.59	16.25	40.17
AMO (ours)	69.36	32.68	86.00	25.13	7.28	41.07	26.40	71.76	34.00	56.20	21.13	18.00	**40.75**

Table 18: Full results of prolonged pre-training on Qwen3-0.6B under 2×, 5×, and 10× Chinchilla optimal token-to-parameter ratios.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
2×													
Muon-KJ	62.21	27.05	81.30	24.33	5.84	35.81	22.60	68.28	31.20	52.96	18.76	17.58	37.33
Muon-PE	62.25	27.03	82.00	25.00	7.77	35.04	22.40	66.92	32.06	52.72	19.74	17.19	37.51
AMO (ours)	62.37	27.22	82.30	25.23	7.02	35.74	24.00	69.15	31.58	52.17	20.33	17.65	**37.90**
5×													
Muon-KJ	65.91	28.75	85.50	25.54	6.58	37.29	24.20	68.88	32.73	55.01	19.66	17.32	38.95
Muon-PE	66.00	27.30	85.00	25.00	7.77	37.54	24.60	69.00	32.34	55.67	20.64	17.65	39.04
AMO (ours)	65.76	28.33	85.70	25.02	7.77	37.61	25.60	69.21	32.67	55.23	20.37	17.98	**39.27**
10×													
Muon-KJ	66.25	33.03	86.20	25.35	6.89	38.33	25.40	69.70	32.73	55.64	18.84	17.73	39.67
Muon-PE	66.98	32.08	86.00	25.13	7.54	37.83	26.00	70.23	32.44	54.78	21.29	17.00	39.78
AMO (ours)	66.00	33.78	86.40	25.13	7.72	38.77	25.60	69.48	33.49	55.93	20.64	17.82	**40.06**

F.8 Budget Control Experiments

Setup. To study compute-quality tradeoff, we vary only the global budget ratio on **Qwen3-0.6B**. Specifically, we evaluate five settings with budget ratio $r \in \{0.8, 0.9, 1.0, 1.1, 1.2\}$. Since the baseline uses 5 NS steps for each of 7 layer types, these settings correspond to total step budgets of 28, 32, 35, 39, and 42, respectively. All other components are kept identical across runs, including the observation phase. In particular, all runs share the same model, data pipeline, optimizer hyperparameters, learning-rate schedule, observation procedure, and execution-phase settings.

All runs begin with the same observation phase under a uniform PE baseline, where each of the 7 Muon-managed layer types uses 5 NS steps. The baseline total budget is therefore $7 \times 5 = 35$. Given a budget ratio r, the total execution-phase budget is set to

$$B = \mathrm{round}(35\, r).$$

Table 19: Full results of continual pre-training with additional 36B tokens on a Qwen3-0.6B checkpoint pretrained on 60B tokens.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Step=2000													
Muon-KJ	64.94	28.41	87.00	24.20	6.59	37.05	23.20	68.50	33.11	55.01	20.72	17.60	38.86
Muon-PE	66.51	29.61	86.00	25.00	7.21	38.63	22.00	70.24	33.88	52.56	19.77	16.43	38.99
AMO (ours)	63.72	29.27	86.20	24.79	7.01	36.52	24.20	68.99	33.40	54.62	22.44	18.07	**39.10**
Step=4000													
Muon-KJ	65.32	29.61	86.40	24.63	6.46	37.94	23.60	68.66	32.54	53.43	19.98	17.96	38.88
Muon-PE	65.17	31.23	86.00	24.89	7.80	37.94	24.00	68.00	31.53	54.06	19.77	17.72	39.01
AMO (ours)	65.53	31.23	86.00	24.55	5.72	37.74	23.40	70.02	33.68	53.99	21.05	17.72	**39.22**
Step=6000													
Muon-KJ	67.47	31.40	87.10	26.51	6.99	39.14	24.40	69.59	32.73	54.06	20.15	17.86	39.78
Muon-PE	68.32	32.08	87.00	25.00	6.18	39.00	25.00	70.84	33.00	54.22	19.74	16.46	39.74
AMO (ours)	68.64	31.97	86.80	26.04	6.23	39.05	25.40	69.70	33.78	54.96	20.31	17.60	**40.04**
Step=8000													
Muon-KJ	66.88	31.74	86.80	26.21	7.25	39.07	25.00	70.35	33.59	53.67	21.62	17.71	39.99
Muon-PE	68.30	31.00	87.00	24.63	5.54	39.11	26.00	72.09	33.78	55.00	20.72	17.74	40.08
AMO (ours)	68.39	30.29	86.60	25.06	7.77	39.18	26.40	70.38	33.88	54.96	21.64	17.19	**40.15**
Step=8580													
Muon-KJ	66.50	31.57	86.60	26.20	7.77	39.12	25.00	70.08	33.40	54.22	21.70	17.94	40.01
Muon-PE	66.00	32.00	88.00	25.60	7.99	39.10	25.60	70.50	34.00	53.43	21.29	18.30	40.15
AMO (ours)	68.56	31.74	88.20	25.00	7.99	39.10	26.00	70.18	34.45	54.56	20.23	17.68	**40.31**

Table 20: Full downstream results of Qwen3-0.6B pre-training under four key designs of AMO: *Observation horizon and interval*, *Observation sample size*, *Shrinkage*, and *Transition duration*.

Setting	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Observation horizon and interval													
(800, 100)	62.09	27.11	82.70	25.21	7.51	34.17	21.80	68.00	31.91	52.80	20.41	18.43	37.68
(1200, 150)	63.17	27.19	82.60	24.89	7.80	35.04	21.60	68.66	31.53	53.09	20.23	17.39	**37.77**
(1600, 200)	63.72	26.89	82.00	25.67	6.27	33.98	21.60	68.01	31.48	53.51	20.48	18.00	37.63
Observation sample size													
4	62.34	26.91	82.60	25.40	7.89	34.15	21.40	68.01	31.81	53.28	20.82	17.24	37.65
8	63.17	27.19	82.60	24.89	7.80	35.04	21.60	68.66	31.53	53.09	20.23	17.39	37.77
16	63.15	27.09	82.80	25.00	7.45	35.00	21.80	68.46	31.86	53.80	20.23	17.37	**37.83**
Shrinkage													
0	62.43	24.74	82.10	24.53	6.13	33.91	23.60	66.97	31.39	53.20	22.03	17.19	37.35
0.5	63.00	25.17	80.30	23.25	7.29	35.95	24.60	67.03	30.91	54.54	22.36	18.02	37.70
0.7	63.17	27.19	82.60	24.89	7.80	35.04	21.60	68.66	31.53	53.09	20.23	17.39	**37.77**
1	63.09	24.74	82.80	26.21	7.86	33.84	24.20	67.14	30.62	52.09	20.73	17.88	37.60
Transition duration													
1	60.98	25.68	81.60	25.24	6.00	33.91	23.00	68.41	31.87	53.20	19.66	17.68	37.27
100	63.86	25.77	81.70	24.09	6.58	34.02	23.60	68.81	31.00	52.17	19.25	17.91	37.40
300	63.17	27.19	82.60	24.89	7.80	35.04	21.60	68.66	31.53	53.09	20.23	17.39	**37.77**
600	62.60	24.49	82.80	25.84	6.77	35.79	24.00	66.70	31.29	52.17	20.88	17.60	37.58

Thus, the five settings above correspond to $B \in \{28, 32, 35, 39, 42\}$. During the observation phase, our AMO keeps the uniform 5-step schedule fixed and only collects spectral statistics. Afterward, it constructs candidate schedules for each layer type within the same step range $[3, 7]$, and solves the final execution-phase allocation under the constraint

$$\sum_t s_t = B,$$

where s_t denotes the NS step count assigned to layer type t. Hence, the only difference across runs is the total orthogonalization budget available when solving the final layer-type-level step allocation.

Table 21: Full results of Llama3.1-760M and Qwen3-0.6B pre-training under budget control settings.

Budget Ratio	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Qwen3-0.6B													
0.8	62.25	26.02	82.40	24.56	7.95	33.89	22.60	67.76	30.72	53.51	21.48	17.19	37.53
0.9	62.68	26.74	82.00	25.00	6.77	33.93	22.40	68.20	31.00	53.70	21.62	18.30	37.70
1.0	63.17	27.19	82.60	24.89	7.80	35.04	21.60	68.66	31.53	53.09	20.23	17.39	37.77
1.1	64.26	27.19	83.00	25.30	7.77	34.04	23.00	67.68	30.05	53.04	20.97	17.89	37.85
1.2	64.39	27.30	83.80	25.82	7.95	34.12	23.60	67.30	30.60	50.67	20.97	18.00	**37.88**
Llama3.1-760M													
0.8	59.93	26.74	81.00	25.00	7.77	33.45	21.80	65.72	30.33	52.64	20.03	17.08	36.79
0.9	59.39	25.85	82.00	26.37	7.45	33.43	22.00	66.00	30.53	52.72	21.00	17.03	36.98
1.0	59.43	26.79	81.50	26.24	6.76	33.79	22.80	67.70	31.43	51.91	21.13	17.60	37.26
1.1	60.18	26.19	82.80	25.43	7.80	33.60	22.00	67.03	31.39	52.51	20.97	17.42	37.28
1.2	60.77	26.79	82.00	25.40	7.02	33.26	22.80	67.80	31.72	53.51	20.13	17.42	**37.39**

Changing r affects only the orthogonalization compute and leaves the rest of training unchanged. For a given layer type, one additional NS step appends one more quintic matrix update on the same tensor shape, so the orthogonalization FLOPs per optimizer step scale approximately linearly with the assigned step count. Relative to the default setting $r = 1.0$ ($B = 35$), $r = 0.8$ and $r = 0.9$ reduce the orthogonalization budget by 20% and 8.6%, while $r = 1.1$ and $r = 1.2$ increase it by 11.4% and 20%, respectively.

Results. Full results are provided in Tab. 21.

F.9 Static Coefficient Experiments

Setup. Inspired by Muon-PE and Muon-CANS (Amsel et al., 2026; Grishina et al., 2026), we construct the following static schedules fully offline and freeze them throughout training. For each Muon-managed layer type (`attn_q`, `attn_k`, `attn_v`, `attn_o`, `mlp_gate`, `mlp_up`, and `mlp_down`), we first estimate an effective PE lower-bound statistic from geometry logs,

$$\ell_{\text{eff}} = \frac{\max(\sigma_{\min}, \tau\sigma_{\max})}{\|M\|_F}.$$

We then aggregate this statistic offline to obtain a fixed ℓ_t for each layer type. The `Typical`, `Conservative`, and `Floor` variants use the 50th, 25th, and 10th percentiles of the aggregated ℓ_{eff}, respectively. Given ℓ_t, we build a PE schedule by optimal composition and choose the smallest number of NS steps $s_t \in \{1, \ldots, 8\}$ whose simulated final approximation error satisfies $1 - l_t^{(s_t)} \leq \varepsilon$, where $\varepsilon \in \{1 \times 10^{-2}, 5 \times 10^{-2}\}$. The corresponding coefficient triplets $\{(a_i, b_i, c_i)\}_{i=1}^{s_t}$ are then fixed for that layer type for the entire run. Additionally, *llama family* or *qwen family* indicates that the offline statistics are pooled only within the corresponding model family; otherwise we use the generic pooled statistics. All other training settings are kept identical to the corresponding main experiments.

Results. Full results are provided in Tabs. 22 and 23.

G Further Analysis

G.1 Prolonged Pre-training

Additional prolonged pre-training results for Llama3.1-1.4B are provided in Tab. 24.

G.2 NS Step Allocation

Setup. In this variant, we keep the optimizer, LR scheduler, coefficient, and training setup unchanged, and only override the number of NS steps according to the layer type for pre-training

Table 22: Full results of Llama3.1-1.4B pre-training under static coefficient settings. *L* denotes the observation data are only from four Llama-family models from pilot experiments.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Muon-KJ	65.32	28.41	85.20	25.47	6.94	37.32	25.00	69.36	32.15	53.67	19.99	16.87	38.81
Muon-PE	66.41	30.03	84.80	23.81	7.53	37.54	24.00	69.15	32.34	53.67	20.64	17.24	**38.93**
Typical 1e-2	66.37	29.78	84.60	24.08	5.59	37.75	23.20	69.04	31.67	51.07	20.31	17.24	38.39
Typical 1e-2 L	66.51	31.91	85.90	25.00	7.21	38.63	24.80	70.24	33.88	52.56	20.31	16.43	39.45
Typical 5e-2	64.06	29.18	82.40	24.12	7.13	37.24	19.80	67.85	31.58	52.64	20.07	17.89	37.83
Typical 5e-2 L	65.07	30.20	85.60	24.65	5.78	38.93	24.20	70.18	34.83	55.56	19.90	17.98	39.41
Cons 1e-2	65.78	29.79	85.00	24.90	6.08	37.84	25.00	69.59	32.63	54.30	19.25	17.24	38.95
Cons 1e-2 L	65.78	31.31	84.80	23.92	6.57	37.51	24.60	70.08	31.96	53.91	21.78	16.48	39.06
Cons 5e-2	64.69	29.69	85.30	25.86	7.20	37.77	25.20	69.31	33.40	53.83	21.38	17.76	39.28
Cons 5e-2 L	66.20	30.38	84.10	25.28	7.39	37.56	26.40	70.39	33.40	53.67	21.25	16.96	39.42
Floor 1e-2	66.58	30.03	84.50	25.99	6.57	37.43	23.00	69.75	32.72	54.06	19.33	16.93	38.91
Floor 1e-2 L	65.53	30.89	85.30	23.94	7.10	38.63	24.80	70.73	33.01	55.33	21.05	17.98	**39.52**
Floor 5e-2	66.33	30.97	84.30	24.66	6.87	39.00	23.60	68.99	33.21	54.62	18.92	17.42	39.07
Floor 5e-2 L	66.98	29.78	85.20	26.00	5.59	38.63	25.40	69.70	33.59	53.99	20.23	17.85	39.41

Table 23: Full results of Qwen3-1.7B under static coefficient settings. *Q* denotes the observation data are only from four Qwen-family models from pilot experiments.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Muon-KJ	68.22	31.74	85.70	24.80	5.54	40.71	26.40	72.09	33.78	55.00	20.72	16.45	40.10
Muon-PE	68.98	31.66	85.30	25.13	7.54	40.96	26.40	71.82	32.44	54.78	20.88	16.93	**40.24**
Typical 1e-2	67.59	32.51	86.30	25.77	6.44	40.73	25.60	70.55	33.11	53.20	19.08	17.39	39.86
Typical 1e-2 Q	69.32	32.68	87.80	24.09	7.04	40.99	27.60	72.52	34.16	56.91	18.10	16.93	**40.68**
Typical 5e-2	68.31	32.51	86.20	24.85	5.36	40.79	24.60	71.82	33.59	56.91	18.35	17.42	40.06
Typical 5e-2 Q	68.98	32.59	86.00	25.00	6.08	40.82	26.80	71.33	32.73	55.96	18.67	16.87	40.15
Cons 1e-2	67.72	32.85	85.60	25.95	6.62	41.12	25.60	70.78	33.21	55.17	21.13	16.98	40.23
Cons 1e-2 Q	69.36	32.94	86.50	24.14	6.74	40.96	26.80	72.74	34.35	55.01	19.25	17.89	40.56
Cons 5e-2	69.32	32.51	86.60	25.02	7.57	40.90	26.60	71.43	33.21	55.81	18.43	17.08	40.37
Cons 5e-2 Q	68.64	31.83	86.20	24.90	7.40	40.98	27.00	71.76	33.59	56.35	19.74	17.61	40.50
Floor 1e-2	67.93	32.68	86.10	25.54	5.86	41.13	27.20	71.76	33.97	56.20	19.74	17.34	40.45
Floor 1e-2 Q	69.82	32.59	86.90	25.23	7.22	41.07	27.40	71.87	32.82	53.00	20.48	17.37	40.38
Floor 5e-2	67.76	31.31	87.10	24.30	6.80	40.94	27.20	70.95	34.26	56.99	19.49	16.61	40.31
Floor 5e-2 Q	68.64	31.83	86.20	24.90	7.40	40.98	27.00	71.76	33.59	56.35	19.74	16.61	40.42

Qwen-300M/0.6B. Based on the observation data, we divide the seven operator types into two coarse groups. The *hard* group contains attention query, output, and value projections, while the *easy* group contains attention key and the three MLP projections. The five static allocation configurations are summarized in Tab. 25.

The allocation sweep exposes a clear sensitivity of Muon to where the NS budget is spent. A1 reduces the step count of the easy group from five to four while keeping the hard group unchanged, testing whether the baseline over-orthogonalizes easier matrices. A2 reallocates more budget toward the hard attention projections by increasing `attn_q/o/v` to six steps while keeping the easy group at four steps. A3 isolates the effect of increasing only the hard group, with all easy layers left at the default five steps. A4 and A5 further probe the coarse hard/easy taxonomy. Instead of uniformly favoring

Table 24: Full results of pre-training Llama3.1-1.4B under 2× Chinchilla optimal ratio.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
2×													
Muon-KJ	65.32	30.41	85.20	25.47	6.94	37.32	24.60	69.36	32.15	53.67	19.99	17.40	38.99
AMO (ours)	67.97	32.13	86.40	24.66	6.84	39.10	24.60	70.95	34.00	53.59	21.70	17.72	**39.97**

Table 25: Static operator-wise NS step allocation configurations. KJ is the default baseline with five NS steps for all Muon-managed matrices.

Group	Operator Type	KJ	A1	A2	A3	A4	A5
Hard	attn_q	5	5	6	6	4	4
	attn_o	5	5	6	6	4	4
	attn_v	5	5	6	6	7	6
Easy	attn_k	5	4	4	5	7	6
	mlp_up	5	4	4	5	4	5
	mlp_gate	5	4	4	5	4	5
	mlp_down	5	4	4	5	4	4

Table 26: Full results of NS step allocation experiments on pre-training Qwen3-300M/0.6B.

Optimizer	AE	AC	SciQ	MM	MM-P	HS	OBQA	PIQA	RACE	WG	CSQA	AGI	Avg
Qwen3-300M													
Muon-KJ	57.15	23.55	77.60	23.07	7.16	30.93	19.00	64.74	29.00	51.38	19.99	17.45	35.09
A1	56.06	21.84	79.60	23.15	6.64	31.16	20.20	65.34	29.09	50.04	19.49	17.68	35.02
A2	56.01	22.44	78.90	23.16	7.75	31.08	21.00	65.89	30.05	53.04	20.80	17.00	**35.59**
A3	56.61	22.78	77.60	23.60	7.15	30.89	20.40	66.21	29.47	50.43	19.98	16.80	35.16
A4	56.40	23.21	78.00	24.43	7.64	30.96	21.60	65.18	30.14	49.09	19.98	17.50	35.34
A5	56.90	23.29	77.20	22.99	7.12	31.01	20.40	65.61	30.72	49.88	19.98	17.42	35.21
Qwen3-0.6B													
Muon-KJ	62.16	26.79	81.90	23.80	7.30	34.75	23.20	67.90	31.67	51.93	19.00	16.92	37.28
A1	61.32	27.39	82.80	24.52	8.22	34.99	24.60	66.76	32.15	53.91	20.39	16.46	**37.79**
A2	61.96	26.37	82.20	24.27	8.12	35.09	22.40	67.08	30.24	52.25	19.57	18.20	37.31
A3	61.15	25.60	82.20	24.64	6.89	34.93	21.60	67.57	31.00	52.49	21.87	18.02	37.33
A4	60.82	25.94	80.70	24.15	6.71	34.90	21.60	67.85	30.14	52.17	20.88	17.32	36.93
A5	61.62	27.22	81.80	25.25	7.41	34.78	22.20	67.41	32.06	51.62	18.84	17.81	37.34

attn_q/o/v, they assign more steps to attn_v and attn_k, while reducing or preserving the MLP projections.

Results. Full results are provided in Tab. 26. Although this simple allocation scheme can be effective, its performance gains are not stable across variants, suggesting that static hand-crafted NS step allocation is only a coarse heuristic rather than a robust replacement for adaptive scheduling.

G.3 Post-warmup Stability

The layer-ordering reversal in Fig. 3(**d**) occurs at step 120, and most crossings reported in App. D.2 fall between steps 20 and 120. Because this period overlaps with learning-rate warmup, **F4** could in principle reflect an initialization transient, with the layer ordering settling once warmup ends. To test this, we analyze layer-ordering stability separately during and after warmup.

Setup. We use full-training geometry traces of Llama3.1-760M and Llama3.1-1.4B, recorded every 10 optimizer steps at checkpoints $t_1 < t_2 < \cdots < t_K$. For each operator type, let $\epsilon_\ell^{(t)}$ denote the post-NS orthogonality error of layer ℓ at step t. A layer pair (i, j) *reverses* between adjacent checkpoints t_k and t_{k+1} if

$$\big(\epsilon_i^{(t_k)} - \epsilon_j^{(t_k)}\big)\big(\epsilon_i^{(t_{k+1})} - \epsilon_j^{(t_{k+1})}\big) < 0. \tag{4}$$

We split adjacent intervals into a warmup phase and a post-warmup phase at the end of warmup (step T_{w}), excluding the single interval that crosses this boundary. Within each phase, the reversal rate is the number of reversals divided by the number of opportunities, i.e., the number of layer pairs times the number of adjacent intervals. We further measure ranking stability with two Spearman correlations of the layer ranking induced by $\epsilon_\ell^{(t)}$: ρ_{adj}, the median correlation between adjacent post-warmup checkpoints, which captures short-term stability; and ρ_{fl}, the correlation between the first post-warmup ranking and the final ranking, which captures cumulative drift.

Table 27: Layer-ordering stability during and after warmup. Reversal rates (%) are normalized by all layer-pair × adjacent-checkpoint opportunities within each phase; *Overall* pools all operator types. ρ_{adj}: median Spearman correlation between adjacent post-warmup checkpoints. ρ_{fl}: Spearman correlation between the first post-warmup and final rankings.

Operator	Warmup Rev.	Post Rev.	Last Rev. Step	ρ_{adj}	ρ_{fl}
Llama3.1-760M					
attn_q	18.63	6.13	3610	0.965	0.769
attn_k	18.18	16.98	3620	0.846	0.420
attn_v	11.41	10.74	3620	0.909	−0.259
attn_o	9.80	3.58	3590	0.986	0.678
mlp_gate	15.51	8.61	3620	0.944	0.413
mlp_up	12.30	11.04	3620	0.923	0.399
mlp_down	16.67	14.10	3620	0.916	−0.399
Overall	14.64	10.17	–	–	–
Llama3.1-1.4B					
attn_q	12.50	4.27	6670	0.979	0.853
attn_k	12.73	15.50	6670	0.862	0.641
attn_v	7.99	10.04	6670	0.929	−0.647
attn_o	6.85	2.40	6670	0.991	0.885
mlp_gate	10.26	6.51	6670	0.971	0.538
mlp_up	9.43	7.66	6670	0.965	0.174
mlp_down	10.76	13.10	6670	0.903	0.409
Overall	10.07	8.50	–	–	–

Results. Tab. 27 leads to three observations. *(i) Reversals persist after warmup.* The overall reversal rate decreases after warmup, from 14.64% to 10.17% on Llama3.1-760M and from 10.07% to 8.50% on Llama3.1-1.4B, but remains far from zero. The decrease is most pronounced for attn_q and attn_o, whereas several operator types show little change or even an increase, e.g., attn_k (12.73% → 15.50%) and mlp_down (10.76% → 13.10%) on Llama3.1-1.4B. *(ii) Reversals continue until late in training.* For every operator type, the last observed reversal occurs at steps 3590–3620 on Llama3.1-760M and at step 6670 on Llama3.1-1.4B, near the end of the recorded traces, so the ordering does not settle at any point after warmup. *(iii) Rankings drift gradually rather than reshuffle abruptly.* ρ_{adj} remains high (0.846–0.991), indicating that rankings at consecutive checkpoints are similar. In contrast, ρ_{fl} is substantially lower and even negative for attn_v on both models (-0.259 and -0.647) and mlp_down on Llama3.1-760M (-0.399), meaning that the final layer ranking can be largely inverted relative to the first post-warmup ranking. Since checkpoint-level noise would lower both correlations alike, the gap between them indicates that small, consistent rank changes accumulate into substantial long-horizon drift.

Together, these results show that the reversal in **F4** is not merely an initialization transient. Although layer ordering becomes more stable after warmup, reversals persist at a non-negligible rate throughout training and accumulate into large changes in ranking. The layer ordering observed at any single point in training is therefore an unreliable predictor of the ordering later on, which supports **F4** and our decision not to prescribe NS allocation by layer depth.