Where Does Warm-Up Come From? Adaptive Scheduling for Norm-Constrained Optimizers

Artem Riabinin[†,1,2] **Andrey Veprikov**[†,1,2,3] **Arman Bolatov**[†,4]
Martin Takáč[4] **Aleksandr Beznosikov**[1,2,5]

[1]Basic Research of Artificial Intelligence Laboratory (BRAIn Lab)
[2]Moscow Independent Research Institute of Artificial Intelligence (MIRAI)
[3]SB AI Lab
[4]Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
[5]Innopolis University

[†]Equal contribution

Abstract

Learning rate warm-up is used in almost every large training run, yet its length is still chosen by trial and error, and each trial is a full training run. We study where warm-up comes from for norm-constrained optimizers such as Muon, Lion and normalized SGD, whose update is a linear minimization oracle over a norm ball. We introduce a generalized smoothness assumption under which local curvature decreases with the suboptimality gap, and we measure this behavior along optimization trajectories, including trajectories trained without any warm-up. Under this assumption, the learning rate that our convergence analysis selects first increases and then decreases, so warm-up followed by decay is a consequence of the proof rather than a design choice. The theory fixes the functional form of the schedule but not its constants. We calibrate them from quantities a practitioner already specifies, namely the peak learning rate, the initial learning rate and the preferred shape of the ramp, which leaves the warm-up length to be determined by the training trajectory itself. On LLaMA pretraining from 124M to 720M parameters with Muon, Lion and normalized SGD, and on Swin-B trained on ImageNet-1K, the adaptive warm-up matches or improves on the best length found by a grid search, without running that search. Our source code is available at `https://github.com/brain-lab-research/adaptive-warmup`.

1 Introduction

In this paper, we consider the problem of training large language models (LLMs), which can be formulated as the following optimization problem:

$$f^\star := \min_{x \in \mathcal{X}} f(x), \tag{1}$$

where $f : \mathcal{X} \to \mathbb{R}$ denotes the loss of the model x with a parameter space $\mathcal{X} := \{(W_1, \ldots, W_L) \mid W_i \in \mathbb{R}^{m_i \times n_i}\}$ representing the collection of the model's L layers.

A prominent recent approach is to solve (1) with norm-constrained optimizers, where the update direction is given by a Linear Minimization Oracle (LMO) over a unit ball. This framework has emerged as a powerful family of methods for training deep networks, with recent successes including Kimi K2 [Team et al., 2025], Moonlight [Liu et al., 2025a], and DeepSeek-V4 [DeepSeek-AI, 2026].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

It unifies several modern optimizers [Veprikov et al., 2026], including normSGD [Hazan et al., 2015], signSGD [Bernstein et al., 2018], Lion [Chen et al., 2023], and Muon [Jordan et al., 2024]. Specifically, the LMO-based update rule is:

$$x^{t+1} = x^t + \eta^t \,\mathrm{LMO}(g^t), \;\; \mathrm{LMO}(g^t) := \arg\min_{q \in \mathcal{X}: \|q\|=1} \langle g^t, q \rangle, \tag{2}$$

where t is the optimization step, $\eta^t > 0$ is the learning rate, g^t is a gradient approximation (e.g., momentum), and $\|\cdot\|$ refers to an arbitrary, possibly non-Euclidean norm. This formulation arises naturally from minimizing a quadratic approximation of the loss function around the point x^t:

$$f(x^t + \Delta x^t) \approx f(x^t) + \langle \nabla f(x^t), \Delta x^t \rangle + \frac{\lambda}{2} \|\Delta x^t\|^2. \tag{3}$$

The update $\Delta x^t = x^{t+1} - x^t$ from (2) is the $\arg\min$ of (3) with respect to $\Delta x^t \in \mathcal{X}$ up to multiplication factors. Different choices of norms $\|\cdot\|$ yield different optimizers: the Euclidean norm recovers normSGD, the ℓ_∞ norm gives signSGD, and the spectral norm leads to Muon. For a detailed derivation of the resulting updates see Appendix E.

The success of LMO-based methods depends not only on the appropriate choice of the norm in (2), but also on the proper selection of the learning rate η^t [Goyal et al., 2017]. In practice, empirically designed schedules are commonly used, such as linear warm-up followed by cosine decay [Loshchilov and Hutter, 2017]. In this work, we focus on the warm-up phase: starting with small learning rates and gradually increasing them before decay. Warm-up is nearly ubiquitous in practice [Goyal et al., 2017, Vaswani et al., 2017, Loshchilov and Hutter, 2017], where it keeps the first steps of training stable and allows a larger peak learning rate afterwards. Its length, however, has no accepted rule. It is copied from a previous setup or found by a sweep in which every point is a full training run, and the best value does not transfer: for Muon in our experiments it ranges from 10 to 3000 steps (Section 6). At the same time, the theoretical reason for warm-up is not fully understood. Therefore, in this paper we address the following research questions:

(i) Can learning rate warm-up be theoretically justified for LMO-based optimizers, rather than being treated as a purely empirical heuristic?

(ii) Can the warm-up duration be determined adaptively during training, eliminating the need for manual tuning?

Guided by these research questions, we make the following contributions:

• We introduce a generalized smoothness assumption where local curvature decreases with the suboptimality gap, and we measure this behavior along optimization trajectories, including trajectories trained with no warm-up and a deterministic non-LLM control.

• We provide a convergence analysis for LMO-based optimizers under this assumption, in which warm-up followed by decay emerges from the proof rather than being imposed heuristically.

• Based on the theory, we develop a practical learning rate scheduler whose constants are calibrated from standard hyperparameters, so that the warm-up length is determined by the training trajectory instead of a search.

• We evaluate the scheduler on LLaMA pretraining from 124M to 720M parameters with Muon, Lion, normalized SGD and Scion, and on Swin-B/ImageNet. In each main configuration, a single adaptive run matches or improves on the best warm-up length of a grid search, and at 583M and 720M it is clearly better than the common 1000-step default.

2 Related work

Norm-constrained optimizers. Norm-constrained optimizers have recently attracted significant attention in deep learning. One of the most prominent examples of such optimizers is Muon [Jordan et al., 2024], which demonstrates strong performance for training deep neural networks [Liu et al., 2025a]. Numerous studies have developed practical variants of Muon and related LMO-based algorithms for large-scale models, analyzing their empirical behavior under spectral or orthogonality constraints [Pethick et al., 2025, Riabinin et al., 2025, Amsel et al., 2025, Chen et al., 2026, Huang et al., 2026, Kovalev, 2025, He et al., 2025, Zhang et al., 2026, Bolatov et al., 2026, Feoktistov et al., 2026].

Adaptive and parameter-free optimizers. Adaptive methods such as Adam [Kingma and Ba, 2015] and AdamW [Loshchilov and Hutter, 2019], distance-estimating methods such as D-Adaptation [Defazio and Mishchenko, 2023] and Prodigy [Mishchenko and Defazio, 2024], and Schedule-Free methods [Defazio et al., 2024] reduce or remove learning rate tuning, but all of them are still used with a heuristically chosen warm-up phase.

Learning rate warm-up. The learning rate warm-up is a widely used heuristic to train deep neural networks, dating back at least to He et al. [2016], which used a small constant learning rate during the initial training phase. The linear warm-up strategy, introduced by Goyal et al. [2017], has since become standard for training ResNets [He et al., 2016] and transformers [Vaswani et al., 2017]. Empirical studies have shown that warm-up enhances training stability to allow larger learning rates, reduces gradient variance, and improves model performance [Gotmare et al., 2019, Liu et al., 2020, Kosson et al., 2024a]. Recent work studies warm-up more directly in modern language-model training: Kosson et al. [2024b] analyze when warm-up can be reduced in GPT training with AdamW and Lion, while Gupta et al. [2023] study re-warmup in continual pretraining under distribution shift. From a geometric perspective, Gilmer et al. [2022] and Kalra and Barkeshli [2024] observed that warm-up induces a sharpness reduction phase where the largest Hessian eigenvalue decreases, enabling larger learning rates in subsequent training. Liu et al. [2020] explain warm-up for Adam through the variance of the adaptive learning rate early in training and propose RAdam, which rectifies this variance instead of warming up. The two accounts are related: the second-moment estimate averages squared gradients over about $1/(1-\beta_2)$ steps while the iterate moves, and the resulting staleness is controlled by the gradient change that Assumption 3.2 bounds, so high early curvature inflates the variance that RAdam rectifies. The variance mechanism, however, needs a per-coordinate second moment, which Muon, Lion and normSGD do not have, and yet all three need warm-up. We include RAdam as a warm-up-free baseline in Section 6.1.

Generalized smoothness and warm-up theory. The standard L-smoothness assumption $\|\nabla f(x) - \nabla f(y)\| \leq L\|x-y\|$ is insufficient to explain the necessity of warm-up, as it implies a uniform bound on curvature throughout the training landscape. Similarly, the (L_0, L_1)-smoothness condition introduced by Zhang et al. [2020], which bounds the Hessian norm by $L_0 + L_1\|\nabla f(x)\|$, has not, to the best of our knowledge, yielded theoretical justifications for warm-up strategies. Recent works address this limitation by linking smoothness to the suboptimality gap, bounding the Hessian by $K_0 + K_1(f(x) - f^\star)$ [Alimisis et al., 2025] or $K_0 + K_\rho(f(x) - f^\star)^\rho$ [Liu et al., 2025b], where $f^\star$ is a target (perfect) loss for the problem (1). However, the theoretical results in these studies are limited to GD and SGD optimizers and derive schedules consisting solely of a warm-up phase without any decay. In contrast, our Assumption 3.2 adopts this suboptimality-dependent framework to naturally derive warm-up and subsequent decay phases specifically for LMO-based methods.

Target loss estimation. As was mentioned in the previous paragraph, all the theoretical frameworks about warmup rely on knowledge of the target value $f^\star$. It commonly appears in adaptive step-size methods, including Polyak-type step sizes Polyak [1963], Orabona and D'Orazio [2025], its stochastic variant Loizou et al. [2021] and Polyak step method for mirror descent You et al. [2022]. It can also be estimated online, from running lower bounds [Hazan and Kakade, 2019] or evolving target levels [You and Li, 2022]. In this work, we do not employ these mechanisms and instead fix an estimate of $f^\star$ for each setup, taken from the expected final loss of the model at its token budget. In Section 6.2, we measure how much the result depends on this estimate at 124M, 583M and 720M parameters and on Swin-B, and compare it with the dependence on the warm-up length that $f^\star$ replaces.

3 Theoretical analysis

3.1 Assumptions and practical motivation

We begin our analysis with the set of assumptions required to study LMO-based optimizers for problem (1).

Assumption 3.1. The function $f : \mathcal{X} \to \mathbb{R}$ is star-convex, i.e., the following inequality holds:

$$f(\beta x^\star + (1-\beta)x) \leq \beta f(x^\star) + (1-\beta)f(x),$$

for all $x \in \mathcal{X}$ and $\beta \in [0,1]$, where $x^\star$ is a global minimizer of f, i.e. $f(x^\star) = f^\star$.

Unlike classical convexity, which requires the inequality to hold for every pair of points in $\mathcal{X}$, star-convexity only enforces convexity with respect to the optimum. This reflects a setting in which the loss landscape may be highly nonconvex globally, yet becomes progressively well behaved along trajectories that approach $x^\star$. Such star-shaped conditions are now standard in modern theoretical analyses of deep neural networks [Zhou et al., 2019]. In particular, recent works analyzing LMO-based optimizers show that star-convexity, together with classical Lipschitz smoothness, is sufficient to guarantee convergence with a constant learning rate [Pethick et al., 2025, Kovalev, 2025]. However, since these works rely on the standard smoothness assumption, they fail to explain the emergence of learning rate warm-up. This motivates Assumption 3.2, which we introduce next.

Assumption 3.2. The function $f : \mathcal{X} \to \mathbb{R}$ is (ρ, K_0, K_1, K_ρ)-smooth, i.e., there exist $K_0, K_1, K_\rho \geq 0$ and $\rho > 0$ such that for all $x, y \in \mathcal{X}$ it holds that:

$$\|\nabla f(x) - \nabla f(y)\|_\star \leq \mathcal{K}(x)\|x - y\|,$$

$$\mathcal{K}(x) := K_0 + K_1 \left(f(x) - f^\star\right) + K_\rho \left(f(x) - f^\star\right)^\rho,$$

where $\|\cdot\|_\star$ is the conjugate norm for $\|\cdot\|$ used in the update rule (2).

This condition strengthens the classical smoothness model in a manner that is aligned with the geometry of deep learning. It recovers the (ρ, L_0, L_ρ)-smoothness (see Section G for details). When $K_\rho = 0$ and $\|\cdot\| := \|\cdot\|_2$, Assumption 3.2 reduces to the version of assumption from Alimisis et al. [2025], which corresponds to a bound of the type:

$$\|\nabla f(x) - \nabla f(y)\| \leq \left(K_0 + K_1(f(x) - f^\star)\right)\|x - y\|.$$

However, our findings (both theoretical and empirical) indicate that the regime $K_\rho > 0$ is not merely an analytical convenience but necessary for accurately modeling the behavior of norm-constrained optimizers in deep learning. To demonstrate this, we evaluate the empirical stability ratio

$$\mathcal{K}^t = \frac{\|\nabla f(x^{t+1}) - \nabla f(x^t)\|_\star}{\|x^{t+1} - x^t\|}$$

as a function of $\Delta^t := f(x^t) - f^\star$ along optimization trajectories of several LMO-based methods: Muon, normSGD, and Lion. The experiments are carried out in a large-scale pretraining regime (see Section 6 for details). Figure 1 reports the observed dependence for Lion. Analogous curves for the remaining methods are provided in Appendix D.

In all cases we observe a clear parabolic trend linking $\mathcal{K}^t$ and Δ^t, consistent with a dominant $K_\rho(f(x) - f^\star)^\rho$ term. This behavior persists across optimizers, and cannot be captured by models with $K_\rho = 0$.

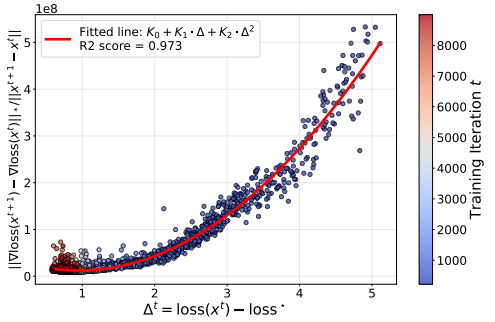


Figure 1: Empirical smoothness ratio $\mathcal{K}^t$ vs. suboptimality gap Δ^t for Lion on large-scale pretraining. The trajectory fits a quadratic dependence, indicating $K_\rho > 0$. Hyperparameter details are in Appendix N.1.

Curvature without warm-up. The trajectories above were trained with a warm-up, so the curvature measured on them could in principle be a side effect of the schedule rather than its cause. We therefore repeat the measurement on trajectories that use no warm-up (Table 2 in Appendix I). On 124M LLaMA/FineWeb with Lion, with no warm-up and no decay, $\mathcal{K}^t$ still decreases together with Δ^t, and the quadratic model explains most of its variance ($R^2 = 0.954 \pm 0.004$ over three seeds). The same holds with a 256-step ramp into a constant learning rate (0.960 ± 0.002), across four learning rates (0.953 ± 0.009), and for deterministic full-batch gradient descent on a CIFAR-10 [Krizhevsky, 2009] CNN (0.946 ± 0.040). The decrease of curvature with the gap is therefore not produced by the warm-up schedule.

Remark 3.3 (On Assumption 3.2). Prior work [Alimisis et al., 2025] fitted $\mathcal{K}^t$ linearly over early training ($\Delta \geq 6$); over the full trajectory the linear model breaks down and the quadratic term is essential. Liu et al. [2025b] diagnosed this limitation; our quadratic model fits the dynamics across the entire trajectory. Our proofs use Assumption 3.2 only with the first argument at an iterate,

$x = x^t$, and y on the segment from x^t to x^{t+1} or on the segment from x^t to $x^\star$. We intend the assumption in this trajectory form. Requiring it for every pair would be too strong, since taking $x = x^\star$ would force $\|\nabla f(y)\|_\star \le K_0\|y - x^\star\|$ for all y. The trajectory form is also the one that Figure 1 and Table 2 measure. A function on the unit ball that satisfies it with $K_0, K_1, K_\rho > 0$, and on which neither the linear nor the superlinear term can be dropped, is given in Appendix J.

The ratio $\mathcal{K}^t$ differs from the top Hessian eigenvalue studied in Edge of Stability [Cohen et al., 2021], and both can coexist (Appendix D.1).

Finally, we require the generated sequence of iterates to remain within a bounded domain.

Assumption 3.4. The iterates x^t generated by (2) are bounded, i.e., there exists $D > 0$ such that for all $t \ge 0$ it holds that $\|x^t - x^\star\| \le D$.

Assumption 3.4 is standard in LMO-type methods [Kovalev, 2025], ensuring that the smoothness constants in Assumption 3.2 remain meaningful along the trajectory. Importantly, this assumption is not required in two common cases: (i) when weight decay is used (Appendix A), (ii) for the Euclidean norm in the step (2), boundedness follows automatically (see Lemma C.1 in Appendix).

3.2 Deterministic case without weight decay

The next theorem provides, to the best of our knowledge, the first convergence analysis under Assumption 3.2 to theoretically predict a learning rate schedule consisting of a warm-up phase followed by decay.

Theorem 3.5. *Suppose Assumptions 3.1, 3.2, and 3.4 hold and the iterates x^t are generated by* (2) *with $g^t = \nabla f(x^t)$.*

If we use the learning rate scheduler (with warm-up followed by decay for $\rho > 1$)

$$\eta^t = \frac{\Delta^t}{D \cdot \mathcal{K}(x^t)} = \frac{\Delta^t}{D \cdot (K_0 + K_1\Delta^t + K_\rho(\Delta^t)^\rho)},$$

where $\Delta^t := f(x^t) - f^\star$, then $\Delta^{t+1} \le \Delta^t$ and $\mathcal{K}(x^{t+1}) \le \mathcal{K}(x^t)$ for all t, and

$$\Delta^T \le \frac{2D^2 \sum_{t=0}^{T-1} \mathcal{K}(x^t)}{T^2} = \mathcal{O}\left(\frac{1}{T}\right).$$

Discussion of Theorem 3.5.

• **Explanation of warm-up behavior.** Since by Theorem 3.5 Δ^t decreases monotonically, the learning rate η^t exhibits a warm-up phase followed by a decay for $\rho > 1$. The transition occurs at the point Δ' obtained by maximizing $\eta(\Delta^t)$:

$$\Delta' := \left(\frac{K_0}{K_\rho(\rho - 1)}\right)^{1/\rho}.$$

Consequently, η^t increases while $\Delta^t > \Delta'$ and decreases thereafter. The superlinear term is necessary for this shape, since $\Delta' \to \infty$ as $K_\rho \to 0$ and only the warm-up phase remains. It is not sufficient on its own. Under the same term, the step size of Liu et al. [2025b] for gradient descent grows as the gap closes and then saturates at a constant, and decay has to be added separately. The decay here comes from the normalized update: the LMO direction has unit length, so the step size equals the step length, and the step that minimizes the bound is the ratio $\Delta/\mathcal{K}(\Delta)$, which has an interior maximum. If we assume sublinear convergence of the objective function, specifically $\Delta^t \sim 1/t$, the asymptotic behavior of the learning rate is illustrated in Figure 7 (Appendix B).

• **Comparison with decay-only schedule.** Since $\mathcal{K}(x^t) \le \mathcal{K}(x^0)$, the bound first gives $\Delta^t \le 2D^2\mathcal{K}(x^0)/t$. Substituting back, $\sum_{t<T} \mathcal{K}(x^t) = K_0 T + \mathcal{O}(\log T)$ for $\rho > 1$, so $\Delta^T \le 2D^2K_0/T + \mathcal{O}(\log T/T^2)$. The leading constant depends only on K_0, the curvature near the optimum. Replacing $\mathcal{K}(x^t)$ with its initial value $\mathcal{K}^0 := K_0 + K_1\Delta^0 + K_\rho(\Delta^0)^\rho$ gives $\Delta^T \le 2D^2\mathcal{K}^0/T$, the same asymptotic rate $\mathcal{O}(1/T)$ but with $\mathcal{K}^0$ in place of the trajectory average of $\mathcal{K}(x^t)$, which is smaller because $\mathcal{K}$ decreases with the gap. The adaptive schedule's advantage is in the transient phase (small t): the decreasing higher-order terms $K_1/t + K_\rho/t^\rho$ allow larger learning rates early in

training. On the function of Appendix J, where the constants are known in closed form, the schedule of Theorem 3.5 reaches a gap of 10^{-2} in 83 iterations, against 830 for a constant step tuned over 61 runs.

• **Extension to weight decay.** The same warm-up-plus-decay schedule carries over to the update rule $x^{t+1} = (1 - \lambda\eta^t)x^t + \eta^t \operatorname{LMO}(g^t)$ with weight decay coefficient $\lambda > 0$, yielding an $\mathcal{O}(1/T)$ rate without Assumption 3.4 (Appendix A).

4 Stochastic extensions

Let us now consider the following stochastic optimization problem:

$$\min_{x \in \mathcal{X}} \{f(x) := \mathbb{E}_{\xi \sim \mathcal{D}}[f_\xi(x)]\}, \tag{4}$$

where $f_\xi : \mathcal{X} \to \mathbb{R}$ represents the loss of the model parameterized by x, associated with a training data point ξ sampled from the probability distribution $\mathcal{D}$.

We consider a variant of the update rule (2) for minimizing (4), by setting $g^t = \nabla f_{\xi^t}(x^t)$ and choosing the Euclidean norm $\|\cdot\| := \|\cdot\|_2$ in the LMO step:

$$x^{t+1} = x^t - \eta^t \frac{\nabla f_{\xi^t}(x^t)}{\|\nabla f_{\xi^t}(x^t)\|}. \tag{5}$$

Following Alimisis et al. [2025], to study convergence in the stochastic setting we require an interpolation condition, which is typically satisfied for over-parameterized networks [Ma et al., 2018].

Assumption 4.1. The optimization problem (4) satisfies the interpolation regime, i.e., for the global minimizer $x^\star$ of f, the following holds almost surely for $\xi \sim \mathcal{D}$: $f_\xi(x^\star) = f_\xi^\star$, where $f_\xi^\star := \inf_{x \in \mathcal{X}} f_\xi(x) > -\infty$. Moreover, almost surely each f_ξ is star-convex with respect to this common point $x^\star$, i.e., Assumption 3.1 holds for f_ξ with $x^\star$ and $f_\xi^\star$ in place of the minimizer and the optimal value.

For the stochastic setting, we can use a weaker variant of Assumption 3.2 that only requires smoothness relative to the optimum $x^\star$.

Assumption 4.2. The function $f_\xi : \mathcal{X} \to \mathbb{R}$ is (ρ, K_0, K_1, K_ρ)-smooth at the global minimizer $x^\star$ of f, i.e., there exist $K_0, K_1, K_\rho \geq 0$, $\rho > 0$ such that for all $x \in \mathcal{X}$ it holds that:

$$\|\nabla f_\xi(x)\|_\star \leq \mathcal{K}_\xi(x)\|x - x^\star\|,$$

$$\mathcal{K}_\xi(x) := K_0 + K_1\left(f_\xi(x) - f_\xi^\star\right) + K_\rho\left(f_\xi(x) - f_\xi^\star\right)^\rho.$$

Under the established assumptions, we characterize the convergence of (5) in Theorem 4.3.

Theorem 4.3. *Suppose Assumptions 4.1 and 4.2 hold with $\|\cdot\| := \|\cdot\|_2$ for each f_ξ, $\xi \sim \mathcal{D}$. Consider the iterates x^t generated by* (5) *with learning rate scheduler*

$$\eta^t = \frac{\Delta_\xi^t}{D \cdot \mathcal{K}_\xi(x^t)},$$

where $D := \|x^0 - x^\star\|$, $\Delta_\xi^t := f_{\xi^t}(x^t) - f_{\xi^t}^\star$, and $\mathcal{K}_\xi(x^t) := K_0 + K_1\Delta_\xi^t + K_\rho(\Delta_\xi^t)^\rho$. Then

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\Delta_\xi^t] \leq \frac{D^2}{T}\sqrt{\mathbb{E}\left[\sum_{t=0}^{T-1}(\mathcal{K}_\xi(x^t))^2\right]}.$$

If, in addition, there exists $M > 0$ such that $\Delta_\xi^t \leq M$ almost surely for all t and $\xi \sim \mathcal{D}$, then

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}[\Delta_\xi^t] \leq \frac{D^2\bar{\mathcal{K}}}{\sqrt{T}} = \mathcal{O}\left(\frac{1}{\sqrt{T}}\right),$$

where $\bar{\mathcal{K}} := K_0 + K_1 M + K_\rho M^\rho$.

Theorem 4.3 holds for the Euclidean norm only, since its proof uses the inner-product expansion of $\|x^t - x^\star\|_2^2$, so it does not cover stochastic Muon or Lion. Existing stochastic guarantees for non-Euclidean LMO methods use constant or monotonically decaying step sizes [Pethick et al., 2025, Riabinin et al., 2025].

5 Practical scheduler with adaptive warmup

In this section, we demonstrate how to apply the optimal learning rate scheduler in practice. Fixing $\rho = 2$ and removing constant scaling factors for clarity, the scheduler derived from Theorems 3.5, A.1 and 4.3 takes the form

$$\eta(\Delta) = \frac{\Delta}{K_0 + K_1\Delta + K_2\Delta^2}. \tag{6}$$

The theory fixes this functional form, and in particular the fact that the learning rate is a function of the current gap Δ, not of the step index. It does not provide the coefficients K_0, K_1, K_2, which are unknown and cannot be measured during training. We determine them through three constraints, matching the number of unknowns. Each constraint uses a quantity that a practitioner already specifies for any scheduler, and none of them is a warm-up length.

(1) Peak learning rate. The function $\eta(\Delta)$ has a unique local maximum at some $\Delta' \in [0; \Delta^0 = f(x^0) - f^\star]$, and we enforce

$$\eta(\Delta') = \mathrm{lr}, \tag{7}$$

where lr denotes the peak learning rate. This quantity is a standard user-facing parameter in all PyTorch learning rate schedulers [Paszke et al., 2019].

(2) Warm-up floor. We match the initial step size by requiring

$$\eta(\Delta^0) = \frac{\mathrm{lr}}{\mathrm{div}}, \tag{8}$$

where $\mathrm{div} \geq 1$ is a user-specified divisor. Typical implementations use $\mathrm{div} = 100$, again mirroring standard PyTorch practice [Paszke et al., 2019].

We solve constraints (7) and (8) analytically to express K_0, K_1 and K_2 in terms of Δ' (see Appendix F.2 for closed-form expressions). Now we need to define Δ'.

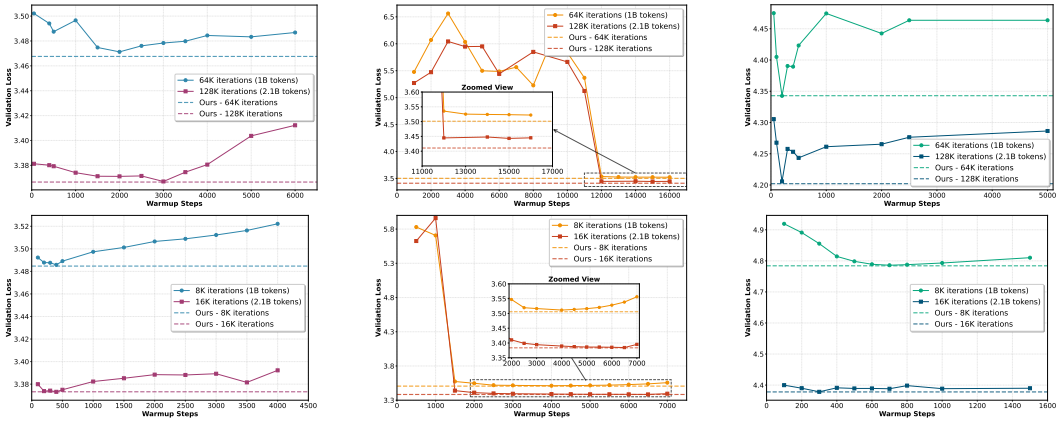
(3) Target-shape matching. The third condition determines Δ' and aligns the profile of (6) with a classical warmup+decay schedule. The target schedule is evaluated once, before the first optimizer step, to solve for Δ', and is then discarded. It sets the learning rate for zero iterations: from the first step on, the warm-up phase uses only the rational function (6) evaluated at the current gap. We first introduce the target learning rate $\eta_{\mathrm{trgt}}(\Delta)$ that we aim to approximate:

$$\eta_{\mathrm{trgt}}(\Delta) = \begin{cases} \frac{\mathrm{lr}}{\mathrm{div}} + \frac{(\mathrm{lr} - \mathrm{lr}/\mathrm{div})(\Delta^0 - \Delta)}{\Delta^0 - \Delta'}, & \Delta \in [\Delta', \Delta^0], \\ \frac{1}{2}\mathrm{lr}\big(1 - \cos(\pi\Delta/\Delta')\big), & \Delta \in [0, \Delta']. \end{cases}$$

The definition of η_{trgt} is provided as the most commonly used choice in practice: linear warm-up and cosine decay Paszke et al. [2019], Loshchilov and Hutter [2017]. The functions on the intervals $[0, \Delta']$ and $[\Delta', \Delta^0]$ may be replaced with any schedulers without affecting the construction.

This condition determines Δ', the gap at which warm-up ends. The target is parameterized by Δ' itself, so no duration in steps is supplied to it. The number of warm-up steps is the number of iterations the trajectory needs to bring Δ^t down to Δ', and it is known only after training has started. The warmup and decay functional forms (e.g., linear, cosine, WSD) enter only as parameters of η_{trgt}: the user specifies a preferred shape, and the procedure finds the transition point Δ' that best matches $\eta(\Delta)$ to it. The rise and fall of the schedule and its dependence on the gap come from the analysis, while the calibration, the decay shape and the switch to a conventional decay below Δ' (Algorithm 1) are practical choices. The inferred warmup duration is largely insensitive to which schedule family is chosen. Section 6.2 confirms this across six schedule types.

Rather than matching $\eta(\Delta)$ to $\eta_{\mathrm{trgt}}(\Delta)$ over the entire interval $[0, \Delta^0]$, we focus on a neighborhood around Δ': this is the warmup-to-decay switching point, where the approximation error has the strongest influence on training, and by constraint (7) the error vanishes at the center. We formalize this by introducing an exponential weight $\exp\big[-(\Delta - \Delta')^2\kappa/(2\sigma_F^2)\big]$ centered at Δ'. The width is set via $\kappa := \sup_{\|u\|=1} \|u\|_F^2$, the worst-case Frobenius-norm scale of the LMO step: for a given Δ and gradient g^t: $\big\|\eta(\Delta)\,\mathrm{LMO}(g^t)\big\|_F^2 \;\leq\; \kappa\,\eta(\Delta)^2$. Requiring the effective step $\sqrt{\kappa}\,\eta(\Delta)$ to have



(a) Muon (b) Lion (c) normSGD

Figure 2: Validation loss on LLaMA 124M model with $\mathrm{bs} = 32$ (top) and 256 (bottom) as a function of manually selected warmup length (solid) vs. adaptive warmup (dashed). Across optimizers, the adaptive method outperforms or at least matches the best manually tuned value without any grid search. All hyperparameter setup is provided in Appendix N.2.

fixed variance σ_F^2 gives $\sigma^2 = \sigma_F^2/\kappa$, normalizing the neighborhood across optimizers (e.g., $\kappa = d$ for signSGD, $\kappa = L$ for normSGD). Closed-form expressions for all methods are in Appendix F.1.

Combining these considerations, we formulate the objective function that we wish to minimize with respect to Δ':

$$\int_0^{\Delta^0} \exp\left[-\frac{(\Delta - \Delta')^2\kappa}{2\sigma_F^2}\right]\big(\eta(\Delta) - \eta_{\mathrm{trgt}}(\Delta)\big)^2 d\Delta, \tag{9}$$

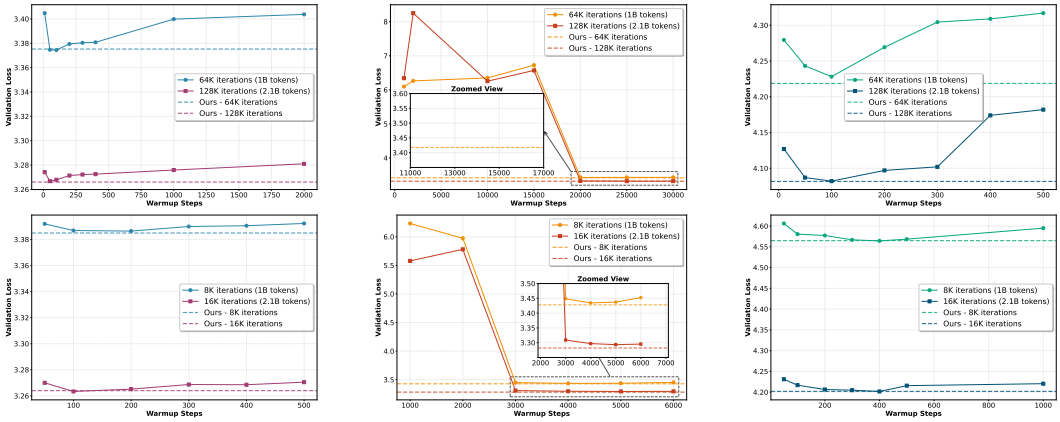
where σ_F is a fixed constant (we use $\sigma_F^2 = 10^{-3}$ in all experiments, Section 6.2 studies the sensitivity to this value). This condition is then enforced numerically: we sample 1000 candidate values of Δ', evaluate the objective (9) for each candidate, and select the minimizer. Since this procedure is executed only once at the beginning of training, it contributes no observable overhead to the overall runtime. Once K_0, K_1, K_2 and Δ' are determined from constraints (7)–(9), the scheduler is fully specified. The complete pseudocode is given as Algorithm 1 in Appendix B.

Positivity of the calibrated schedule. The calibrated coefficients are not the constants of Assumption 3.2, and (16) can give $K_1 < 0$. The scheduler only needs the denominator $K_0 + K_1\Delta + K_2\Delta^2$ to be positive. Its discriminant is negative in every configuration of the paper, so this holds for all $\Delta \geq 0$ (Appendix K).

6 Experiments

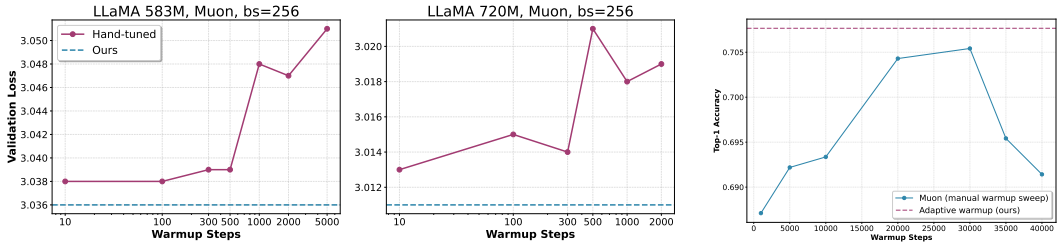
We follow the protocol of Semenov et al. [2025], which benchmarks optimizers on LLaMA-style decoder-only transformers with rotary position embeddings [Fedus et al., 2022] trained on FineWeb [Penedo et al., 2024]. For Muon and Lion we copy all hyperparameters (learning rate, weight decay, etc.) from that setup. normSGD is not covered there, so we tune its hyperparameters separately. As the target loss $f^\star$ we use a value close to the expected final loss of each configuration on FineWeb, which depends on the optimizer and the token budget (between 3.2 and 4.3, Table 10). Section 6.2 studies the sensitivity to this value. Full configuration tables are provided in Appendix N.2.

We consider models with 124M and 210M parameters, and evaluate 32 and 256 batch-size regimes. For each optimizer, we sweep over a range of manually specified warmup lengths and compare the resulting final validation loss against the value obtained from our adaptive warmup scheduler. At 124M and 210M alone the curves aggregate 225 manual warm-up runs, roughly 1.2k H200 GPU-hours. Figures 2 and 3 summarize the trends for Muon, Lion and normSGD for 124M and 210M



(a) Muon (b) Lion (c) normSGD

Figure 3: Validation loss on LLaMA 210M model with $\mathrm{bs} = 32$ (top) and 256 (bottom) as a function of manually selected warmup length (solid) vs. adaptive warmup (dashed). Across optimizers, the adaptive method outperforms or at least matches the best manually tuned value without search. All hyperparameter setup is provided in Appendix N.2.



(a) LLaMA 583M and 720M, Muon, $\mathrm{bs} = 256$ (b) Swin-B, ImageNet-1K, Muon

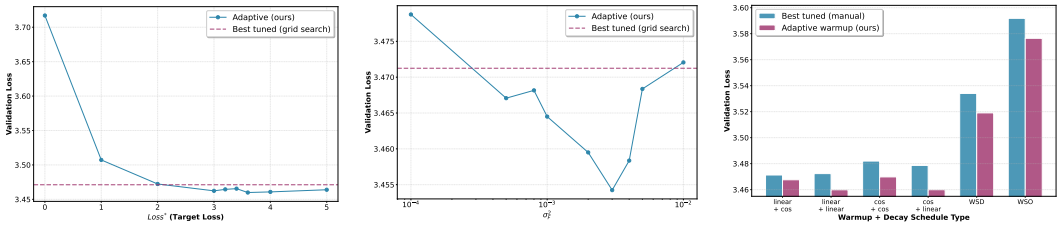
Figure 4: (a) Final validation loss at 583M and 720M parameters (4.2B tokens) as a function of a fixed warm-up length (solid) and for the adaptive warm-up (dashed). Run-to-run variation at 583M is about 0.001. (b) Top-1 accuracy of Swin-B as a function of a fixed warm-up length (solid) and for the adaptive warm-up (dashed).

model sizes, respectively. We then extend the Muon setting to 583M and 720M parameters, trained for 4.2B tokens each (Section 6.1).

6.1 Results

Language model pretraining. Across all optimizers, model sizes, and batch sizes, the adaptive schedule matches or improves on the best hand-tuned warm-up without any grid search. For Muon and normSGD the dashed line lies below the sweep minimum or within 0.001 of it, which is the size of the run-to-run variation we measure at larger scale. The Lion results show that too short a warm-up causes divergence, while the adaptive method avoids it. The best fixed length itself is hard to predict. For Muon at 2.1B tokens it is 3000 steps at 124M with $\mathrm{bs} = 32$, 400 at 124M with $\mathrm{bs} = 256$, 50 at 210M with $\mathrm{bs} = 32$ and 100 at 210M with $\mathrm{bs} = 256$, so a value tuned on a smaller proxy would not transfer.

Larger models. We extend the Muon setting to LLaMA models with 583M and 720M parameters, batch size 256 and 4.2B tokens (Figure 4a, details in Appendix L). The adaptive run ends at 3.036 at 583M and 3.011 at 720M, against 3.038 and 3.013 for the best fixed warm-up of the sweep, and against 3.048 and 3.018 for the common 1000-step default. With run-to-run variation near 0.001 (second seeds at 583M), we read the first comparison as a tie and the second as a clear improvement. These adaptive runs use $\mathrm{div} = 10$, which alone does not explain the gain: at 720M, cosine schedules with $\mathrm{div} = 10$ give 3.013, 3.015 and 3.016 for 10, 100 and 1000 warm-up steps.



(a) Target loss $f^\star$ (b) Frobenius variance σ_F^2 (c) Schedule type η_{trgt}

Figure 5: Ablation studies on LLaMA 124M pretraining with $\mathrm{bs} = 32$ using Muon. The dashed line in (a) and (b) shows the best manually tuned warm-up, in (c), teal bars show the best manual warmup and pink bars show the adaptive scheduler.

Other optimizers and a warm-up-free baseline. On 124M LLaMA/FineWeb with $\mathrm{bs} = 32$ and 64k steps (about 1B tokens), RAdam [Liu et al., 2020] without warm-up ends at 3.496, above both Muon with its best tuned warm-up (3.471, 2000 steps) and Muon with the adaptive warm-up (3.467). For Scion [Pethick et al., 2025], which changes only the closed-form κ of the scheduler, the adaptive schedule ends at 3.718, below every fixed warm-up length in $\{100, 1000, 2000, 4000\}$ (best 3.777).

Vision transformers. We evaluate a Swin Transformer Base (Swin-B) [Liu et al., 2021] trained from scratch on ImageNet-1K [Deng et al., 2009] with Muon for 50K iterations at batch size 128, using $f^\star = 0.4$ as the target loss. This short budget serves to compare warm-up choices, not to reach the final accuracy of the architecture. Figure 4b shows Top-1 accuracy vs. warmup length. The adaptive scheduler reaches a higher accuracy than the best manually tuned length in this single-seed comparison.

6.2 Ablations

Target loss. The target loss $f^\star$ is the only parameter requiring prior task knowledge. We study the sensitivity on LLaMA 124M pretraining with $\mathrm{bs} = 32$ using Muon (Figure 5a). Any $f^\star \in [3.0, 4.0]$ outperforms the best manually tuned warm-up, only extreme values ($f^\star = 0$ or $f^\star \geq 5$) cause noticeable degradation. Compare this with the warm-up length that $f^\star$ replaces. At 583M, moving $f^\star$ from 3.1 to 3.3 changes the final loss by 0.001, while the fixed warm-up grid spans 0.013. At 720M both spreads are 0.008. On Swin-B, $f^\star \in \{0.30, 0.35, 0.45\}$ around the submitted 0.40 gives top-1 accuracy between 69.0% and 69.8% (Appendix M). In practice $f^\star$ can be read from a scaling-law fit or a short proxy run.

Frobenius-scale variance. Values of σ_F^2 from 10^{-3} to 3×10^{-3} outperform the manually tuned baseline, while $\sigma_F^2 \leq 10^{-4}$ underestimates the required warm-up (Figure 5b). All experiments use $\sigma_F^2 = 10^{-3}$ without per-setup tuning.

Schedule type. Varying the target η_{trgt} over linear and cosine warm-up and decay, WSD [Hu et al., 2024] and WSO [Yano et al., 2026] (Figure 5c), the adaptive method outperforms the manually tuned baseline for all six families.

7 Conclusion

Under a suboptimality-dependent smoothness assumption, warm-up followed by decay emerges from the convergence proof for LMO-based optimizers, with the schedule $\eta(\Delta) = \Delta/\mathcal{K}(\Delta)$. Our scheduler calibrates its constants from standard hyperparameters, so the warm-up length follows from the trajectory. On LLaMA from 124M to 720M parameters with Muon, Lion, normSGD and Scion, and on Swin-B/ImageNet, one adaptive run matches or improves on the best grid-searched warm-up length in each main configuration. Because the rule is indexed by the gap and not by the step, it complements hyperparameter-transfer methods [Yang et al., 2021, Dey et al., 2025, Mlodozeniec et al., 2026], which set the peak learning rate but not the warm-up length. Limitations are discussed in a separate section after the conclusion.

Acknowledgments

The work of A. Riabinin, A. Veprikov and A. Beznosikov was supported by the Ministry of Economic Development of the Russian Federation (agreement No. 139-15-2025-013, dated June 20, 2025, IGK 000000C313925P4B0002).

Limitations

The scheduler needs an estimate of $f^\star$. Its sensitivity is no larger than that of the warm-up length it replaces, but we have no formal bound on the admissible error. The stochastic analysis covers only Euclidean normalized SGD under interpolation. Assumption 3.2 is a trajectory-level condition whose measurement is noisy, and the sign of the fitted quadratic coefficient depends on the visited range of Δ. Most sweeps are single-seed, and the margins over the best sweep point are of the order of the run-to-run variation, so the evidence supports removing the search, not a systematic gain in loss. With $\mathrm{div} = 100$, the adaptive run at 720M ended 0.008 to 0.016 above the best fixed warm-up (Appendix L).

References

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, Zhuofu Chen, Jialei Cui, Hao Ding, Mengnan Dong, Angang Du, Chenzhuang Du, Dikang Du, Yulun Du, Yu Fan, Yichen Feng, Kelin Fu, Bofei Gao, Hongcheng Gao, Peizhong Gao, Tong Gao, Xinran Gu, Longyu Guan, Haiqing Guo, Jianhang Guo, Hao Hu, Xiaoru Hao, Tianhong He, Weiran He, Wenyang He, Chao Hong, Yangyang Hu, Zhenxing Hu, Weixiao Huang, Zhiqi Huang, Zihao Huang, Tao Jiang, Zhejun Jiang, Xinyi Jin, Yongsheng Kang, Guokun Lai, Cheng Li, Fang Li, Haoyang Li, Ming Li, Wentao Li, Yanhao Li, Yiwei Li, Zhaowei Li, Zheming Li, Hongzhan Lin, Xiaohan Lin, Zongyu Lin, Chengyin Liu, Chenyu Liu, Hongzhang Liu, Jingyuan Liu, Junqi Liu, Liang Liu, Shaowei Liu, T. Y. Liu, Tianwei Liu, Weizhou Liu, Yangyang Liu, Yibo Liu, Yiping Liu, Yue Liu, Zhengying Liu, Enzhe Lu, Lijun Lu, Shengling Ma, Xinyu Ma, Yingwei Ma, Shaoguang Mao, Jie Mei, Xin Men, Yibo Miao, Siyuan Pan, Yebo Peng, Ruoyu Qin, Bowen Qu, Zeyu Shang, Lidong Shi, Shengyuan Shi, Feifan Song, Jianlin Su, Zhengyuan Su, Xinjie Sun, Flood Sung, Heyi Tang, Jiawen Tao, Qifeng Teng, Chensi Wang, Dinglu Wang, Feng Wang, Haiming Wang, Jianzhou Wang, Jiaxing Wang, Jinhong Wang, Shengjie Wang, Shuyi Wang, Yao Wang, Yejie Wang, Yiqin Wang, Yuxin Wang, Yuzhi Wang, Zhaoji Wang, Zhengtao Wang, Zhexu Wang, Chu Wei, Qianqian Wei, Wenhao Wu, Xingzhe Wu, Yuxin Wu, Chenjun Xiao, Xiaotong Xie, Weimin Xiong, Boyu Xu, Jing Xu, Jinjing Xu, L. H. Xu, Lin Xu, Suting Xu, Weixin Xu, Xinran Xu, Yangchuan Xu, Ziyao Xu, Junjie Yan, Yuzi Yan, Xiaofei Yang, Ying Yang, Zhen Yang, Zhilin Yang, Zonghan Yang, Haotian Yao, Xingcheng Yao, Wenjie Ye, Zhuorui Ye, Bohong Yin, Longhui Yu, Enming Yuan, Hongbang Yuan, Mengjie Yuan, Haobing Zhan, Dehao Zhang, Hao Zhang, Wanlu Zhang, Xiaobin Zhang, Yangkun Zhang, Yizhi Zhang, Yongting Zhang, Yu Zhang, Yutao Zhang, Yutong Zhang, Zheng Zhang, Haotian Zhao, Yikai Zhao, Huabin Zheng, Shaojie Zheng, Jianren Zhou, Xinyu Zhou, Zaida Zhou, Zhen Zhu, Weiyu Zhuang, and Xinxing Zu. Kimi k2: Open agentic intelligence, 2025. URL `https://arxiv.org/abs/2507.20534`.

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`.

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

Andrey Veprikov, Arman Bolatov, Aleksandr Bogdanov, Samuel Horváth, Aleksandr Beznosikov, Martin Takáč, and Slavomir Hanzely. Preconditioned norms: A unified framework for steepest descent, quasi-newton and adaptive methods, 2026. URL `https://arxiv.org/abs/2510.10777`.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In *Advances in Neural Information Processing Systems*, vol-

ume 28, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/934815ad542a4a7c5e8a2dfa04fea9f5-Abstract.html.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 560–569. PMLR, 2018. URL https://proceedings.mlr.press/v80/bernstein18a.html.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2302.06675.

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

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. URL https://arxiv.org/abs/1706.02677.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1608.03983.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.03762.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 49069–49104. PMLR, 2025. URL https://proceedings.mlr.press/v267/pethick25a.html.

Artem Riabinin, Egor Shulgin, Kaja Gruntkowska, and Peter Richtárik. Gluon: Making muon & scion great again! (bridging theory and practice of lmo-based optimizers for llms). *arXiv preprint arXiv:2505.13416*, 2025. URL https://arxiv.org/abs/2505.13416.

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. *arXiv preprint arXiv:2505.16932*, 2025. URL https://arxiv.org/abs/2505.16932.

Lizhang Chen, Jonathan Li, and Qiang Liu. Muon optimizes under spectral norm constraints. *Transactions on Machine Learning Research*, 2026, 2026. URL https://openreview.net/forum?id=Blz4hjxLwU.

Feihu Huang, Yuning Luo, and Songcan Chen. LiMuon: Light and fast Muon optimizer for large models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2509.14562.

Dmitry Kovalev. Understanding gradient orthogonalization for deep learning via non-euclidean trust-region optimization. *arXiv preprint arXiv:2503.12645*, 2025. URL https://arxiv.org/abs/2503.12645.

Chuan He, Zhanwang Deng, and Zhaosong Lu. Low-rank orthogonalization for large-scale matrix optimization with applications to foundation model training. *arXiv preprint arXiv:2509.11983*, 2025. URL https://arxiv.org/abs/2509.11983.

Ruijie Zhang, Yequan Zhao, Ziyue Liu, Zhengyang Wang, Dongyang Li, Yupeng Su, Sijia Liu, and Zheng Zhang. Teon: Tensorized orthonormalization beyond layer-wise muon for large language model pre-training. *arXiv preprint arXiv:2601.23261*, 2026.

Arman Bolatov, Artem Riabinin, Nikita Kornilov, Andrey Veprikov, Samuel Horváth, Martin Takáč, and Aleksandr Beznosikov. LionMuon: Alternating spectral and sign descent for efficient training, 2026. URL https://arxiv.org/abs/2605.19811.

Dmitrii Feoktistov, Timofey Belinsky, Andrey Veprikov, Amir Zainullin, and Aleksandr Beznosikov. Softsign: Smooth sign in your optimizer for better parameter heterogeneity handling. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. *arXiv preprint arXiv:2301.07733*, 2023. URL https://arxiv.org/abs/2301.07733.

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*, pages 35779–35804. PMLR, 2024. URL https://proceedings.mlr.press/v235/mishchenko24a.html.

Aaron Defazio, Xingyu Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2405.15682.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. URL https://arxiv.org/abs/1512.03385.

Akhilesh Gotmare, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1810.13243.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1908.03265.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Rotational equilibrium: How weight decay balances learning across neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024a. URL https://arxiv.org/abs/2305.17212.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Analyzing & reducing the need for learning rate warmup in gpt training. *Advances in Neural Information Processing Systems*, 37:2914–2942, 2024b.

Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats L Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to (re) warm your model? *arXiv preprint arXiv:2308.04014*, 2023.

Justin Gilmer, Behrooz Ghorbani, Ankush Garg, Sneha Kudugunta, Behnam Neyshabur, David Cardoze, George Dahl, Zachary Nado, and Orhan Firat. A loss curvature perspective on training instabilities of deep learning models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2110.04369.

Dayal Singh Kalra and Maissam Barkeshli. Why warmup the learning rate? underlying mechanisms and improvements. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2406.09405.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/1905.11881.

Foivos Alimisis, Rustem Islamov, and Aurelien Lucchi. Why do we need warm-up? a theoretical perspective. *arXiv preprint arXiv:2510.03164*, 2025. URL https://arxiv.org/abs/2510.03164.

Yuxing Liu, Yuze Ge, Rui Pan, An Kang, and Tong Zhang. Theoretical analysis on how learning rate warmup accelerates convergence. *arXiv preprint arXiv:2509.07972*, 2025b. URL https://arxiv.org/abs/2509.07972.

Boris T Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864–878, 1963.

Francesco Orabona and Ryan D'Orazio. New perspectives on the Polyak stepsize: Surrogate functions and negative results. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://arxiv.org/abs/2505.20219.

Nicolas Loizou, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. In *International Conference on Artificial Intelligence and Statistics*, pages 1306–1314. PMLR, 2021.

Jun-Kai You, Hao-Chung Cheng, and Yen-Huan Li. Minimizing quantum rényi divergences via mirror descent with polyak step size. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 252–257. IEEE, 2022.

Elad Hazan and Sham Kakade. Revisiting the polyak step size. *arXiv preprint arXiv:1905.00313*, 2019.

Jun-Kai You and Yen-Huan Li. Two polyak-type step sizes for mirror descent. *arXiv preprint arXiv:2210.01532*, 2022.

Yi Zhou, Junjie Yang, Huishuai Zhang, Yingbin Liang, and Vahid Tarokh. SGD converges to global minimum in deep learning via star-convex path. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1901.00451.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021.

Siyuan Ma, Raef Bassily, and Mikhail Belkin. The power of interpolation: Understanding the effectiveness of SGD in modern over-parametrized learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3325–3334. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/ma18a.html.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1912.01703.

Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. Benchmarking optimizers for large language model pretraining. *arXiv preprint arXiv:2509.01440*, 2025.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

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, Datasets and Benchmarks Track*, volume 37, pages 30811–30849, 2024.

Ze Liu, Yutong Lin, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Lin, Baining Guo, and Han Hu. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Kazuki Yano, Shun Kiyono, Sosuke Kobayashi, Sho Takase, and Jun Suzuki. Pre-training LLM without learning rate decay enhances supervised fine-tuning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2603.16127.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, volume 34, pages 17084–17097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8df7c2e3c3c3be098ef7b382bd2c37ba-Paper.pdf.

Nolan Dey, Bin Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: CompleteP enables compute-efficient deep transformers. In *Advances in Neural Information Processing Systems*, volume 38, pages 137707–137739. Curran Associates, Inc., 2025. doi: 10.52202/085713-4597. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/c917d8b9e01427f3184d80ade22f4d1f-Paper-Conference.pdf.

Bruno Mlodozeniec, Pierre Ablin, Louis Béthune, Dan Busbridge, Michal Klein, Jason Ramapuram, and Marco Cuturi. Completed hyperparameter transfer across modules, width, depth, batch and duration. In *International Conference on Learning Representations*, pages 45300–45325, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4b0c1645f3d6a1730931e65ecbf91ac3-Paper-Conference.pdf.

Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *International Conference on Learning Representations*, 2023.

Kfir Y. Levy. The power of normalization: Faster evasion of saddle points. *arXiv preprint arXiv:1611.04831*, 2016. URL https://arxiv.org/abs/1611.04831.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. In *NeurIPS 2024 Workshop on Optimization for Machine Learning (OPT)*, 2024. URL https://arxiv.org/abs/2409.20325.

Contents

1 Introduction **1**

2 Related work **2**

3 Theoretical analysis **3**

3.1 Assumptions and practical motivation 3

3.2 Deterministic case without weight decay 5

4 Stochastic extensions **6**

5 Practical scheduler with adaptive warmup **7**

6 Experiments **8**

6.1 Results 9

6.2 Ablations 10

7 Conclusion **10**

A Deterministic case with weight decay **18**

B Adaptive warmup scheduler: full algorithm and illustration **18**

C Automatic boundedness for the euclidean norm **19**

D Empirical motivation for Assumption 3.2 **20**

D.1 Relation to edge of stability 21

E Connection to specific optimizers: detailed derivations **22**

F Details of the adaptive scheduler from Section 5 **23**

F.1 Closed-form expressions for κ 23

F.2 Closed-form solution for K_0, K_1, K_2 and selection of Δ' 24

G Connection (ρ, K_0, K_1, K_ρ)-smoothness with (ρ, L_0, L_ρ)-smoothness **25**

H Proofs **26**

H.1 Proof of Theorem 3.5 26

H.2 Proof of Theorem A.1 26

H.3 Proof of Theorem 4.3 27

I Curvature measurements on trajectories without warm-up **28**

J A function satisfying Assumption 3.2 with all constants positive **30**

K Positivity of the calibrated schedule **30**

L Larger-scale experiments **30**

M Target-loss sensitivity **31**

N Experimental details **33**

N.1 Assumption validation experiments 33

N.2 Main experiments hyperparameters 34

N.3 Vision transformer experiment hyperparameters 34

A Deterministic case with weight decay

We extend Theorem 3.5 by incorporating weight decay with coefficient $\lambda > 0$:

$$x^{t+1} = (1 - \lambda\eta^t)x^t + \eta^t \operatorname{LMO}(g^t). \tag{10}$$

This form of weight decay is standard for improving generalization in large-scale networks [Loshchilov and Hutter, 2019].

Theorem A.1. *Suppose Assumptions 3.1 and 3.2 hold with $\rho > 1$ and the iterates x^t are generated by (10) with $g^t = \nabla f(x^t)$, $\lambda \in \left(0, \frac{1}{\max(\|x^0\|, \|x^\star\|, 1/\lambda_{\max})}\right]$, where $\lambda_{\max} = \left[8\left(\rho\left(\frac{K_0}{\rho-1}\right)^{\frac{\rho-1}{\rho}} K_\rho^{\frac{1}{\rho}} + K_1\right)\right]^{1/2}$.*

If we use the learning rate scheduler

$$\eta^t = \frac{\lambda\Delta^t}{8\mathcal{K}(x^t)} = \frac{\lambda\Delta^t}{8\left(K_0 + K_1\Delta^t + K_\rho(\Delta^t)^\rho\right)},$$

then $\Delta^{t+1} \le \Delta^t$ and $\mathcal{K}(x^{t+1}) \le \mathcal{K}(x^t)$ for all t, and

$$\Delta^T \le \frac{16\sum_{t=0}^{T-1}\mathcal{K}(x^t)}{\lambda^2 T^2} = \mathcal{O}\left(\frac{1}{T}\right).$$

Discussion.

• **Relaxation of Assumption 3.4.** Unlike Theorem 3.5, boundedness of the iterates is not required: the regularization term $-\lambda\eta^t x^t$ implicitly keeps the iterates bounded.

• **Relation to Theorem 3.5.** All implications of Theorem 3.5 hold here by setting $D \sim 1/\lambda$.

B Adaptive warmup scheduler: full algorithm and illustration

Algorithm 1 gives the complete pseudocode for the scheduler derived in Section 5. On the first call to `get_lr`, it solves constraints (7)–(9) to determine K_0, K_1, K_2, and Δ'. While $\Delta^t \ge \Delta'$, it returns the rational schedule $\eta(\Delta^t)$ from (6). When Δ^t falls below Δ', it switches to a standard decay scheduler for the remaining steps. The switch is needed for two reasons: (6) produces near-zero learning rates at small Δ, and the parabolic smoothness approximation breaks down there (see Figure 1).

Algorithm 1 Adaptive warmup scheduler

Input: total steps T, target loss $f^\star$, peak lr, divisor div, warmup and decay schedulers = (linear, cos), optimizer $\in$ {Muon, Lion, normSGD}.
Initialize: `is_init` $\leftarrow$ `False`, `is_decay` $\leftarrow$ `False`, `warmup_steps` $\leftarrow 0$.
function `get_lr(loss)`
  $\Delta \leftarrow$ `loss` $- f^\star$
  if not `is_init` **then**
    `is_init` $\leftarrow$ `True`
    determine K_0, K_1, K_2, Δ' via constraints (7)–(9)
  end if
  if $\Delta \ge \Delta'$ **and** not `is_decay` **then**
    `warmup_steps` $\leftarrow$ `warmup_steps` $+ 1$
    return $\dfrac{\Delta}{K_0 + K_1\Delta + K_2\Delta^2}$
  else
    if not `is_decay` **then**
      `is_decay` $\leftarrow$ `True`
      initialize decay scheduler with $T_{\text{decay}} \leftarrow T -$ `warmup_steps` and starting lr = lr
    end if
    return `decay_scheduler.get_lr()`
  end if
end function

Illustration of the Adaptive Scheduler

Figure 6 shows the learning rate profile for Lion on FineWeb. During warmup, η_t follows $\eta(\Delta^t)$ from (6). After Δ^t crosses Δ', it switches to cosine decay.

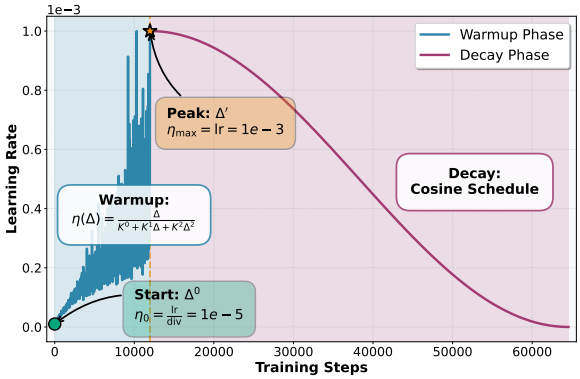


Figure 6: Example of learning rate schedule produced by our adaptive warmup procedure for the Lion optimizer on FineWeb. The warm-up phase (left) follows the rational schedule $\eta(\Delta)$ in (6) up to the peak at Δ', while the decay phase (right) uses the standard cosine scheduler.

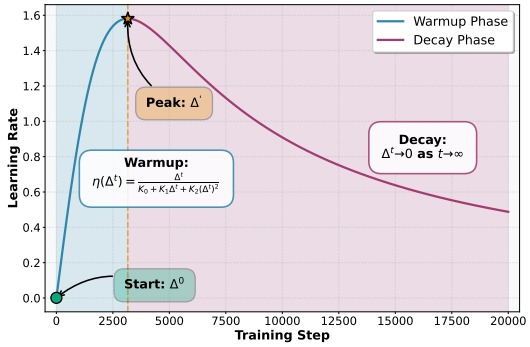


Figure 7: Asymptotic behavior of η^t for $D = 1$, $\rho = 2$, $K_0 = 10^{-4}$, $K_1 = 0$, $K_\rho = 10^3$ with $\Delta^t = 1/t$, showing the warm-up phase followed by decay.

C Automatic boundedness for the euclidean norm

The following lemma confirms that for the Euclidean norm, Assumption 3.4 is automatically satisfied by our learning rate schedule.

Lemma C.1. *Suppose Assumptions 3.1 and 3.2 hold under the Euclidean norm* $\|\cdot\| = \|\cdot\|_2$. *Consider the iterates* x^t *generated by* (2) *with* $g^t = \nabla f(x^t)$ *and learning rate* $\eta^t = \frac{\Delta^t}{D \cdot \mathcal{K}(x^t)}$, *where* $D := \|x^0 - x^\star\|_2$. *Then* $\|x^t - x^\star\|_2 \le D$ *for all* $t \ge 0$.

Proof. We proceed by induction on t. The base case $t = 0$ holds trivially since $\|x^0 - x^\star\|_2 = D$.

For the inductive step, assume $\|x^t - x^\star\|_2 \le D$. From the update rule (2) with the Euclidean norm:

$$x^{t+1} = x^t - \eta^t \frac{\nabla f(x^t)}{\|\nabla f(x^t)\|_2}.$$

Using the star-convexity condition (Assumption 3.1):

$$\langle \nabla f(x^t), x^t - x^\star \rangle \ge f(x^t) - f^\star = \Delta^t > 0.$$

We compute:

$$\begin{aligned}\|x^{t+1}-x^\star\|_2^2 &= \|x^t-x^\star\|_2^2 - 2\eta^t\frac{\langle\nabla f(x^t),x^t-x^\star\rangle}{\|\nabla f(x^t)\|_2} + (\eta^t)^2\\ &\le \|x^t-x^\star\|_2^2 - 2\eta^t\frac{\Delta^t}{\|\nabla f(x^t)\|_2} + (\eta^t)^2.\end{aligned}$$

By Assumption 3.2, $\|\nabla f(x^t)\|_2 \le \mathcal{K}(x^t)\|x^t-x^\star\|_2 \le \mathcal{K}(x^t)D$. With $\eta^t = \frac{\Delta^t}{D\cdot\mathcal{K}(x^t)}$:

$$\begin{aligned}\|x^{t+1}-x^\star\|_2^2 &\le D^2 - 2\cdot\frac{\Delta^t}{D\cdot\mathcal{K}(x^t)}\cdot\frac{\Delta^t}{\mathcal{K}(x^t)D} + \frac{(\Delta^t)^2}{D^2\mathcal{K}(x^t)^2}\\ &= D^2 - \frac{2(\Delta^t)^2}{D^2\mathcal{K}(x^t)^2} + \frac{(\Delta^t)^2}{D^2\mathcal{K}(x^t)^2}\\ &= D^2 - \frac{(\Delta^t)^2}{D^2\mathcal{K}(x^t)^2} \le D^2.\end{aligned}$$

Thus $\|x^{t+1}-x^\star\|_2 \le D$, completing the induction. □

D Empirical motivation for Assumption 3.2

To complement the main-text visualization based on Lion (Figure 1), we report the same Lipschitz–style diagnostic for Muon and normalized SGD. For each optimizer, we plot the empirical ratio

$$\mathcal{K}^t \;=\; \frac{\|\nabla f(x^{t+1})-\nabla f(x^t)\|_\star}{\|x^{t+1}-x^t\|} \qquad \text{versus} \qquad \Delta^t = f(x^t)-f^\star,$$

and overlay a fitted curve of the form

$$K_0 + K_1\Delta + K_2\Delta^2.$$

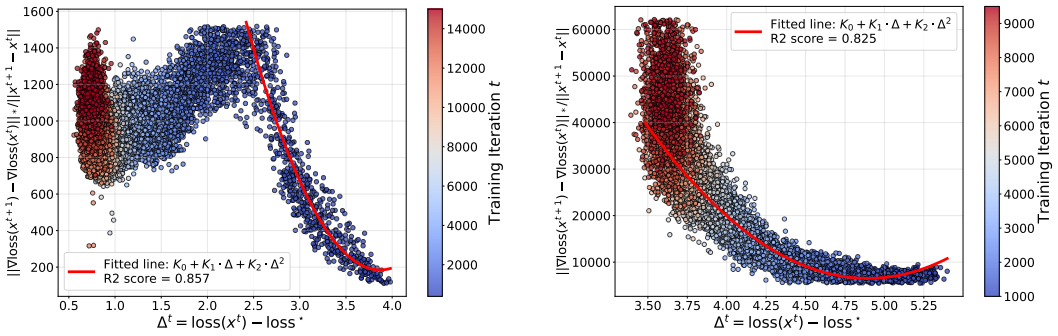
The resulting trajectories, shown in Figures 1 and 8, reveal three consistent phenomena.

1. **Suboptimality-dependent curvature is essential.** Across all optimizers, the empirical dependence of $\mathcal{K}^t$ on Δ^t exhibits clear curvature, and the quadratic term dominates the linear component except near the beginning of training. This is consistent with $K_\rho > 0$ in Assumption 3.2. Appendix I discusses how the fitted coefficients depend on the visited range of Δ.
2. **Different optimizers induce different curvature profiles.** The magnitude and slope of $\mathcal{K}^t$ vary substantially across methods. Muon produces smoother trajectories with a tight parabolic fall-off, while normSGD displays a steeper and higher-variance decrease as training progresses. These differences reflect the geometry of each update rule (LMO direction, normalization scale and implicit conditioning), and a single global Lipschitz constant would approximate them poorly.
3. **The trend is optimizer-universal.** Even though the shapes differ, both optimizers exhibit the same qualitative pattern as Lion in the main text: the curvature decreases together with Δ^t and cannot be explained by a constant Lipschitz term alone. All three optimizers behave as predicted by Assumption 3.2, but with optimizer-dependent constants.

Overall, these measurements support $K_\rho > 0$ along the trajectories of LMO optimizers and show that the precise smoothness structure is optimizer-specific. This is why the scheduler calibrates K_0, K_1, K_2 instead of assuming a fixed curvature model.

Experimental setup. All measurements are obtained using the same model, data pipeline, and optimizer hyperparameters as in Section 6, with settings inherited from Semenov et al. [2025]. The only aspect that varies across optimizers is the number of optimization steps observed. For each iteration, we estimate $\Delta^t = f_{\xi^t}(x^t) - f^\star$ and compute

$$R^t \;=\; \frac{\|\nabla f_{\xi^t}(x^{t+1})-\nabla f_{\xi^t}(x^t)\|_\star}{\|x^{t+1}-x^t\|}$$



(a) Muon (b) normSGD

Figure 8: Empirical smoothness ratio $\mathcal{K}^t$ versus suboptimality gap Δ^t for Muon and normalized SGD. The fitted curve $K_0 + K_1\Delta + K_2\Delta^2$ (red) describes the observed trajectory, and the magnitude and curvature differ across optimizers.

on the same mini-batch ξ^t of size 64 by performing an additional forward and backward pass, ensuring that the difference is not corrupted by batch-to-batch noise. While both Δ^t and R^t remain stochastic due to minibatch sampling, the batch is sufficiently large for these quantities to provide faithful estimates of the smoothness trend under the same training regime used in the main experiments.

D.1 Relation to edge of stability

The Edge of Stability (EoS) phenomenon [Cohen et al., 2021, Damian et al., 2023] refers to the empirical observation that, during full-batch gradient descent, the top Hessian eigenvalue $\lambda_{\max}(\nabla^2 f(x^t))$ rises to approximately $2/\eta$ and then stabilizes. At first glance this appears to contradict Assumption 3.2, under which the local smoothness $\mathcal{K}(x^t) = K_0 + K_1\Delta^t + K_\rho(\Delta^t)^\rho$ decreases as training proceeds. Four observations resolve this.

First, $\mathcal{K}^t$ and $\lambda_{\max}(\nabla^2 f(x^t))$ measure different things. The top eigenvalue is the worst-case curvature over all directions. The ratio $\mathcal{K}^t = \|\nabla f(x^{t+1}) - \nabla f(x^t)\|_* / \|x^{t+1} - x^t\|$ is the gradient change specifically along the LMO update direction $x^{t+1} - x^t$, which is constrained to the unit ball of the optimization norm. For LMO methods, this direction avoids the high-curvature subspace where $\lambda_{\max}$ is attained. The two quantities can move in opposite directions simultaneously.

Second, our scheduler uses the rational rule only before the inferred transition point Δ'. When $\Delta^t < \Delta'$, Algorithm 1 switches to a standard decay scheduler. Thus the method uses the decreasing directional ratio only to determine the warmup phase, and it does not model the late-training regime where EoS effects are expected to appear. This design choice is consistent with the empirical role of warmup: the adaptive part is active while the optimization trajectory moves from the initial high-loss region toward the stable training regime, then it is turned off.

Third, EoS has been studied mainly for full-batch gradient descent [Cohen et al., 2021, Damian et al., 2023]. LMO methods such as Muon apply Newton-Schulz orthogonalization, projecting the gradient onto the spectral norm ball [Jordan et al., 2024, Chen et al., 2026]. This fundamentally changes the update direction relative to gradient descent, and no EoS analog has been established for such methods.

Finally, Assumption 3.2 is a trajectory-level condition (Remark 3.3), and the quantity the scheduler relies on is measured along the trajectory. Our validation in Figures 1 and 8 measures $\mathcal{K}^t$ along the actual LMO optimization trajectory. Worst-case sharpness over all directions can grow while directional smoothness along LMO steps decreases, and this is precisely the distinction between the EoS measurement and the quantity used by our warmup rule.

E Connection to specific optimizers: detailed derivations

This general LMO-based framework in (2) unifies several popular optimizers by instantiating different norms $\|\cdot\|$ on the parameter space [Veprikov et al., 2026]. Table 1 summarizes the correspondence.

Optimizer	Norm	Dual	$\mathrm{LMO}(g)$
normSGD	ℓ_2	ℓ_2	$-g/\|g\|$
signSGD/Lion	ℓ_∞	ℓ_1	$-\mathrm{sign}(g)$
Muon	Spectral	Nuclear	$-UV^\top$
Layer-wise	$\max_i \|\cdot\|_{(i)}$	$\sum_i \|\cdot\|_{(i),\star}$	per-layer

Table 1: Norm choice and corresponding optimizers. The update rule (2) recovers each optimizer.

Recall that the linear minimization oracle is

$$\mathrm{LMO}(g) = \arg\min_{\|t\|=1} \langle g, t\rangle = -\frac{g^\star}{\|g^\star\|},$$

where $g^\star$ is the subgradient of the dual norm $\|g\|_\star = \max_{\|t\|=1}\langle g, t\rangle$. Below we derive the LMO for each norm choice.

Normalized Gradient Descent (ℓ_2 norm). For the Euclidean norm $\|\cdot\| = \|\cdot\|_2$, the dual norm is also ℓ_2 (self-duality), and $\mathrm{LMO}(g) = -g/\|g\|_2$. The update (2) becomes:

$$x^{t+1} = x^t - \eta^t \frac{\nabla f(x^t)}{\|\nabla f(x^t)\|_2},$$

which is exactly **Normalized Gradient Descent** [Hazan et al., 2015, Levy, 2016]. This method decouples the direction from the magnitude, ensuring unit-norm steps.

signSGD (ℓ_∞ norm). For the ℓ_∞ norm $\|t\|_\infty = \max_i |t_i|$, the dual norm is ℓ_1: $\|g\|_\star = \|g\|_1 = \sum_i |g_i|$. The LMO has the closed form:

$$\mathrm{LMO}(g) = -\,\mathrm{sign}(g),$$

where $\mathrm{sign}(g)_i = \mathrm{sign}(g_i) \in \{-1, 0, +1\}$. The update (2) becomes:

$$x^{t+1} = x^t - \eta^t\,\mathrm{sign}(\nabla f(x^t)),$$

which is exactly **signSGD** [Bernstein et al., 2018]. This connection was noted by Bernstein and Newhouse [2024], who observed that signSGD is steepest descent under the ℓ_∞ norm.

Lion (ℓ_∞ norm with momentum). Lion [Chen et al., 2023] extends signSGD by incorporating momentum:

$$m^{t+1} = \beta_2 m^t + (1-\beta_2)\nabla f(x^t), \quad x^{t+1} = x^t - \eta^t\,\mathrm{sign}(\beta_1 m^t + (1-\beta_1)\nabla f(x^t)).$$

Since the update direction is still given by the sign function, Lion corresponds to the ℓ_∞ norm geometry with a momentum-averaged gradient. This optimizer was discovered via symbolic search and has demonstrated strong performance across vision, language, and diffusion models while being more memory-efficient than Adam.

Muon (spectral norm on matrices). For matrix-valued parameters $W \in \mathbb{R}^{m\times n}$, the spectral norm is $\|W\|_{\mathrm{op}} = \sigma_{\max}(W)$ (largest singular value). Its dual is the nuclear norm $\|G\|_\star = \sum_i \sigma_i(G)$ (sum of singular values). The LMO is the orthogonalized gradient:

$$\mathrm{LMO}(G) = -UV^\top,$$

where $G = U\Sigma V^\top$ is the SVD. The update (2) becomes:

$$W^{t+1} = W^t - \eta^t\,\mathrm{Ortho}(\nabla f(W^t)),$$

where $\mathrm{Ortho}(G) = UV^\top$ is the nearest (semi-)orthogonal matrix to G. This is exactly the **Muon** optimizer [Jordan et al., 2024], which uses Newton-Schulz iterations to efficiently approximate the orthogonalization. Recent work has scaled Muon to large language models [Liu et al., 2025a].

Layer-wise optimizers (supremum norm over layers). In practice, modern neural networks consist of L layers with parameters $(W_1, \dots, W_L)$, where each layer may have different dimensions and structure. A natural approach is to apply different norms to different layers and combine them via the supremum:

$$\|(W_1, \dots, W_L)\| := \max_{i=1,\dots,L} \|W_i\|_{(i)},$$

where $\|\cdot\|_{(i)}$ denotes the norm chosen for the i-th layer. The dual norm is then

$$\|(G_1, \dots, G_L)\|_\star = \sum_{i=1}^{L} \|G_i\|_{(i),\star},$$

Since the unit ball of the max-norm is the Cartesian product of the per-layer unit balls, the LMO decomposes layer-wise and acts on all layers at once:

$$\mathrm{LMO}(G_1, \dots, G_L) = \big(\mathrm{LMO}_{(1)}(G_1), \dots, \mathrm{LMO}_{(L)}(G_L)\big),$$

where $\mathrm{LMO}_{(i)}$ is the LMO corresponding to the norm $\|\cdot\|_{(i)}$. This is the geometry used in all our experiments. In particular, the normSGD we run is layer-wise, with the global norm $\max_\ell \|W_\ell\|_F$ (Appendix F.1), and the ℓ_2 row of Table 1 corresponds to a single block.

This framework allows combining different optimizers for different layers—for example, using Muon (spectral norm) for large attention matrices, signSGD/Lion (ℓ_∞ norm) for embedding layers, and Normalized GD (ℓ_2 norm) for small dense layers. Such hybrid strategies are commonly used in practice [Jordan et al., 2024, Liu et al., 2025a]. Spectral and sign oracles can also be alternated over iterations instead of over layers [Bolatov et al., 2026].

F Details of the adaptive scheduler from Section 5

F.1 Closed-form expressions for κ

The width parameter σ in the exponential weight of constraint (9) is set as $\sigma^2 = \sigma_F^2/\kappa$. The motivation: the Frobenius-norm step satisfies $\|\eta(\Delta)\,\mathrm{LMO}(g)\|_F^2 \le \kappa\,\eta(\Delta)^2$, so requiring the effective step $\sqrt{\kappa}\,\eta(\Delta)$ to have fixed variance σ_F^2 gives variance σ_F^2/κ for $\eta(\Delta)$ itself. For signSGD the LMO outputs ± 1 vectors of dimension d, giving $\kappa = d$; for normSGD the output has unit Frobenius norm per layer, giving $\kappa = L$.

Formally,

$$\kappa \;:=\; \sup_{\|u\|=1} \big\{\|u\|_F^2\big\},$$

where $\|\cdot\|$ is the norm used inside the LMO update and $\|\cdot\|_F$ is the Frobenius norm. In our implementation, parameters are grouped by layers, and the global optimization norm is defined as the maximum over layers of the per-layer norm. Consider a model consisting of layers indexed by $\ell = 1, \dots, L$, where the ℓ-th weight matrix has dimensions $m_\ell \times n_\ell$. Under the max-over-layers geometry used throughout, the closed-form expressions for κ are:

- **Muon (spectral norm).** Each layer uses the spectral norm $\|\cdot\|_{\mathrm{op}}$ and its dual, the nuclear norm $\|\cdot\|_*$. For matrices with operator norm bounded by 1,

  $$\sup_{\|A\|_{\mathrm{op}}=1} \|A\|_F^2 = \mathrm{rank}(A) = \min(m_\ell, n_\ell),$$

  and the nuclear norm satisfies $\|A\|_* \le \mathrm{rank}(A)\|A\|_{\mathrm{op}}$. With a max-over-layers constraint, the total worst-case Frobenius contribution sums across layers:

  $$\kappa_{\mathrm{Muon}} = \sum_{\ell=1}^{L} \min(m_\ell, n_\ell).$$

- **Lion / Sign-type (entrywise ℓ_∞ norm).** Here the layerwise optimization norm is $\|\cdot\|_\infty$, and the dual is $\|\cdot\|_1$. For a matrix with $d_\ell = m_\ell n_\ell$ entries,

  $$\sup_{\|A\|_\infty=1} \|A\|_F^2 = d_\ell, \qquad \|A\|_1 \le d_\ell \|A\|_\infty.$$

Accumulating across layers yields

$$\kappa_{\text{Lion}} = \sum_{\ell=1}^{L} m_\ell n_\ell.$$

- **Normalized SGD (Frobenius norm).** In this case, the per-layer norm is already the Frobenius norm, and the global norm is

$$\|W\| = \max_\ell \|W_\ell\|_F.$$

The corresponding dual norm aggregates Frobenius contributions over layers, giving

$$\sup_{\max_\ell \|U_\ell\|_F = 1} \sum_{\ell=1}^{L} \|U_\ell\|_F^2 = L.$$

Thus,

$$\kappa_{\text{normSGD}} = L.$$

Intuitively, κ measures the worst-case amplification of the effective step when mapping from the LMO geometry to Frobenius scale. The differences above reflect the fundamental geometry of the optimizers: spectral structure for Muon, elementwise bounds for Lion, and layerwise normalization for normalized SGD. These forms of κ are used in all our experiments (see Section 6).

F.2 Closed-form solution for K_0, K_1, K_2 and selection of Δ'

We work with the practical schedule

$$\eta(\Delta) = \frac{\Delta}{K_0 + K_1\Delta + K_2\Delta^2},$$

with three unknown coefficients K_0, K_1, K_2 and three constraints:

(i) $\eta'(\Delta') = 0$ (unique critical point at Δ'),
(ii) $\eta(\Delta') = \text{lr}$ (peak learning rate),
(iii) $\eta(\Delta^0) = \text{lr}/\text{div}$ (initial warmup value).

Step 1: critical point. Differentiating $\eta(\Delta)$,

$$\eta'(\Delta) = \frac{(K_0 + K_1\Delta + K_2\Delta^2) - \Delta(K_1 + 2K_2\Delta)}{(K_0 + K_1\Delta + K_2\Delta^2)^2} = \frac{K_0 - K_2\Delta^2}{(K_0 + K_1\Delta + K_2\Delta^2)^2}.$$

Setting $\eta'(\Delta') = 0$ yields

$$K_0 = K_2(\Delta')^2. \tag{11}$$

Step 2: value constraints. Condition (ii) gives

$$\frac{\Delta'}{K_0 + K_1\Delta' + K_2(\Delta')^2} = \text{lr}.$$

Using (11),

$$\frac{\Delta'}{2K_2(\Delta')^2 + K_1\Delta'} = \text{lr} \quad \Longrightarrow \quad 1 = \text{lr}(2K_2\Delta' + K_1).$$

Solving for K_1:

$$K_1 = \frac{1}{\text{lr}} - 2K_2\Delta'. \tag{12}$$

Condition (iii) gives

$$\frac{\Delta^0}{K_0 + K_1\Delta^0 + K_2(\Delta^0)^2} = \frac{\text{lr}}{\text{div}}.$$

Substituting (11),

$$\frac{\Delta^0}{K_2(\Delta')^2 + K_1\Delta^0 + K_2(\Delta^0)^2} = \frac{\text{lr}}{\text{div}}. \tag{13}$$

Step 3: solving for K_2, K_1, K_0. Insert (12) into (13) and rearrange to obtain

$$(\mathrm{div} - 1)\Delta^0 = \mathrm{lr} K_2 (\Delta^0 - \Delta')^2.$$

Thus

$$K_2 = \frac{\Delta^0(\mathrm{div} - 1)}{\mathrm{lr}(\Delta^0 - \Delta')^2}. \tag{14}$$

Then

$$K_0 = K_2(\Delta')^2 = \frac{\Delta^0(\Delta')^2(\mathrm{div} - 1)}{\mathrm{lr}(\Delta^0 - \Delta')^2}. \tag{15}$$

Finally use (12) and (14) to obtain

$$K_1 = \frac{(\Delta^0)^2 - 2\Delta^0\Delta'\mathrm{div} + (\Delta')^2}{\mathrm{lr}(\Delta^0 - \Delta')^2}. \tag{16}$$

It is immediate to verify that (11), (16), (14) jointly satisfy (i)–(iii).

Step 4: selecting Δ'. The coefficients above hold for any Δ' with $\Delta' \neq \Delta^0$, leaving Δ' as a free scalar. We choose it by minimizing the MSE between the schedule and a target warmup+decay curve:

$$\Delta' = \arg\min_{\Delta > 0} \left\{ \mathbb{E}_{\tilde{\Delta} \sim \mathcal{N}(\Delta; \sigma_F^2/\kappa)} \left[(\eta(\tilde{\Delta}) - \eta_{\mathrm{trgt}}(\tilde{\Delta}))^2 \right] \right\},$$

In practice, we evaluate 1000 candidate values of Δ', form (K_0, K_1, K_2) using (14)–(16), enforce constraints numerically, and choose the minimizer. This is done once at initialization and incurs negligible runtime overhead.

G Connection (ρ, K_0, K_1, K_ρ)-smoothness with (ρ, L_0, L_ρ)-smoothness

The following Lemma G.2 connects the standard (ρ, L_0, L_ρ)-smoothness (Assumption G.1) with the (ρ, K_0, K_1, K_ρ)-smoothness (Assumption 3.2).

Assumption G.1. The function $f : \mathcal{X} \to \mathbb{R}$ is (ρ, L_0, L_ρ)-smooth, i.e., there exist $L_0, L_\rho \geq 0$, $\rho > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\|_\star \leq (L_0 + L_\rho \|\nabla f(x)\|_\star^\rho)\|x - y\|$$

for all $x, y \in \mathcal{X}$.

Lemma G.2. *If the function f is (ρ, L_0, L_ρ)-smooth with $0 < \rho < 2$, then it is $\left(\frac{\rho}{2-\rho}, K_0, K_1, K_\rho\right)$-smooth.*

Proof. It is easy to show that if f is (ρ, L_0, L_ρ)-smooth, then for all $x, y \in \mathcal{X}$,

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L_0 + L_\rho \|\nabla f(x)\|_\star^\rho}{2} \|y - x\|^2.$$

By taking $y = x + \frac{\|\nabla f(x)\|_\star}{L_0 + L_\rho \|\nabla f(x)\|_\star^\rho} \mathrm{LMO}(\nabla f(x))$, we have $f^\star \leq f(y) \leq f(x) - \frac{\|\nabla f(x)\|_\star^2}{2L_0 + 2L_\rho \|\nabla f(x)\|_\star^\rho}$. Denoting $g := \|\nabla f(x)\|_\star$ and $\Delta := f(x) - f^\star$, this gives $\Delta \geq \frac{g^2}{2(L_0 + L_\rho g^\rho)}$.

We now upper bound g^ρ in terms of Δ. If $g^\rho \leq \frac{L_0}{L_\rho}$, then trivially $g^\rho \leq \frac{L_0}{L_\rho}$. Otherwise, $L_0 + L_\rho g^\rho \leq 2L_\rho g^\rho$, hence $\Delta \geq \frac{g^{2-\rho}}{4L_\rho}$. Since $0 < \rho < 2$, raising both sides to the power $\frac{\rho}{2-\rho}$ gives $g^\rho \leq (4L_\rho \Delta)^{\frac{\rho}{2-\rho}}$. Combining both cases via $\max\{a, b\} \leq a + b$:

$$g^\rho \leq \frac{L_0}{L_\rho} + (4L_\rho \Delta)^{\frac{\rho}{2-\rho}}, \quad \text{so} \quad L_0 + L_\rho g^\rho \leq 2L_0 + L_\rho (4L_\rho)^{\frac{\rho}{2-\rho}} \Delta^{\frac{\rho}{2-\rho}}.$$

Plugging this into Assumption G.1 yields Assumption 3.2 with exponent $\frac{\rho}{2-\rho}$, constants $K_0 := 2L_0$, $K_1 := 0$, and $K_\rho := L_\rho (4L_\rho)^{\frac{\rho}{2-\rho}}$. □

H Proofs

Corollary H.1. *If f is differentiable and star-convex (Assumption 3.1), then for all $x \in \mathcal{X}$:*

$$f(x) - f^\star \leq \langle \nabla f(x), x - x^\star \rangle.$$

Proof. From star-convexity, $f(x + \beta(x^\star - x)) - f(x) \leq -\beta(f(x) - f^\star)$ for $\beta \in [0, 1]$. Dividing by β and taking $\beta \to 0^+$ yields the directional derivative $\langle \nabla f(x), x^\star - x \rangle \leq -(f(x) - f^\star)$. □

H.1 Proof of Theorem 3.5

Proof. Denote $\mathcal{K}^t := K_0 + K_1\Delta^t + K_\rho(\Delta^t)^\rho$. By Assumption 3.2,

$$\Delta^{t+1} \leq \Delta^t + \langle \nabla f(x^t), x^{t+1} - x^t \rangle + \frac{\mathcal{K}^t}{2}\|x^{t+1} - x^t\|^2.$$

Since

$$\langle \nabla f(x^t), x^{t+1} - x^t \rangle = \eta^t \langle \nabla f(x^t), \operatorname{LMO}(\nabla f(x^t)) \rangle = -\eta^t \|\nabla f(x^t)\|_\star,$$

then

$$\Delta^{t+1} \leq \Delta^t - \eta^t \|\nabla f(x^t)\|_\star + \frac{\mathcal{K}^t}{2}(\eta^t)^2.$$

By Corollary H.1 and the boundedness of the iterates x^t, we have $\Delta^t = f(x^t) - f^\star \leq \langle \nabla f(x^t), x^t - x^\star \rangle \leq \|\nabla f(x^t)\|_\star \|x^t - x^\star\| \leq D\|\nabla f(x^t)\|_\star$, where $D > 0$. Therefore,

$$\Delta^{t+1} \leq \Delta^t - \frac{\Delta^t}{D}\eta^t + \frac{\mathcal{K}^t}{2}(\eta^t)^2.$$

The right hand side is minimized by setting $\eta^t = \frac{\Delta^t}{D\mathcal{K}^t}$, thus $\Delta^{t+1} \leq \Delta^t$ and

$$\Delta^{t+1} \leq \Delta^t - \frac{(\Delta^t)^2}{2D^2\mathcal{K}^t} \leq \Delta^t - \frac{\Delta^t\Delta^{t+1}}{2D^2\mathcal{K}^t}.$$

Telescoping the sum after dividing both sides by $\Delta^t\Delta^{t+1}$ yields $\sum_{t=0}^{T-1} \frac{1}{2D^2\mathcal{K}^t} \leq \frac{1}{\Delta^T} - \frac{1}{\Delta^0}$. By the Cauchy-Schwarz inequality, $\sum_{t=0}^{T-1} \frac{1}{2D^2\mathcal{K}^t} \geq \frac{T^2}{\sum_{t=0}^{T-1} 2D^2\mathcal{K}^t}$, which gives $\frac{T^2}{\sum_{t=0}^{T-1} 2D^2\mathcal{K}^t} \leq \frac{1}{\Delta^T} - \frac{1}{\Delta^0} \leq \frac{1}{\Delta^T}$. Therefore,

$$\Delta^T \leq \frac{2D^2 \sum_{t=0}^{T-1} \mathcal{K}^t}{T^2}.$$

It is easy to show for $\rho > 1$ that since Δ^t is non-increasing, then the learning rate $\eta^t = \frac{\Delta^t}{D\mathcal{K}^t}$ is non-decreasing for $\Delta^t \geq \left(\frac{K_0}{(\rho-1)K_\rho}\right)^{\frac{1}{\rho}}$ (warm-up stage) and non-increasing thereafter (decay stage). □

H.2 Proof of Theorem A.1

Proof. Denote $\mathcal{K}^t := K_0 + K_1\Delta^t + K_\rho(\Delta^t)^\rho$. By Assumption 3.2,

$$\Delta^{t+1} \leq \Delta^t + \langle \nabla f(x^t), x^{t+1} - x^t \rangle + \frac{\mathcal{K}^t}{2}\|x^{t+1} - x^t\|^2.$$

Note that $g(u) = \frac{u}{8(K_0 + K_1 u + K_\rho u^\rho)}$, $u > 0$, $\rho > 1$, is maximized at $u = \left(\frac{K_0}{K_\rho(\rho-1)}\right)^{\frac{1}{\rho}}$ with a maximum value of $\left[8\left(\rho\left(\frac{K_0}{\rho-1}\right)^{\frac{\rho-1}{\rho}} K_\rho^{\frac{1}{\rho}} + K_1\right)\right]^{-1}$, so $0 < \lambda\eta^t \leq 1$ for $0 < \lambda \leq \frac{1}{\max(\|x^0\|, \|x^\star\|, 1/\lambda_{\max})}$, where $\lambda_{\max} = \left[8\left(\rho\left(\frac{K_0}{\rho-1}\right)^{\frac{\rho-1}{\rho}} K_\rho^{\frac{1}{\rho}} + K_1\right)\right]^{1/2}$.

Let us prove $\|x^t\| \leq 1/\lambda$ for all $t \geq 0$ by induction. The induction base ($t = 0$) holds trivially. Assuming $\|x^t\| \leq 1/\lambda$ holds, the update rule and the bound $\|\operatorname{LMO}(g^t)\| \leq 1$ yield:

$$\|x^{t+1}\| = \|(1 - \lambda\eta^t)x^t + \eta^t \operatorname{LMO}(g^t)\| \leq (1 - \lambda\eta^t)\|x^t\| + \eta^t \leq (1 - \lambda\eta^t)\frac{1}{\lambda} + \eta^t = \frac{1}{\lambda},$$

where $1 - \lambda\eta^t \geq 0$, since $0 < \lambda\eta^t \leq 1$. Thus, $\lambda\|x^t\| \leq 1$ for all $t \geq 0$.

Using these bounds, we obtain

$$\|x^{t+1} - x^t\| = \eta^t\|\text{LMO}(g^t) - \lambda x^t\| \leq \eta^t(1 + \lambda\|x^t\|) \leq 2\eta^t,$$

and

$$\|x^t - x\| = \lambda\eta^t\|x^t - x^\star\| \leq \eta^t(\lambda\|x^t\| + \lambda\|x^\star\|) \leq 2\eta^t,$$

where $x := (1 - \lambda\eta^t)x^t + \lambda\eta^t x^\star$.

Therefore,

$$\Delta^{t+1} \leq \Delta^t + \langle\nabla f(x^t), x^{t+1} - x^t\rangle + 2\mathcal{K}^t(\eta^t)^2.$$

Let us construct an upper bound for $\langle\nabla f(x^t), x^{t+1} - x^t\rangle$:

$$\begin{aligned}\langle\nabla f(x^t), x^{t+1} - x^t\rangle &= \langle\nabla f(x^t), x - x^t\rangle + \eta^t\langle\nabla f(x^t), \text{LMO}(\nabla f(x^t)) - \lambda x^\star\rangle \\ &\leq \langle\nabla f(x^t), x - x^t\rangle - \eta^t\|\nabla f(x^t)\|_\star + \eta^t\lambda\|x^\star\|\|\nabla f(x^t)\|_\star \\ &\leq \langle\nabla f(x^t), x - x^t\rangle.\end{aligned}$$

By Assumption 3.2 with first argument x^t, integrating ∇f along the segment from x^t to x gives the two-sided bound $|f(x) - f(x^t) - \langle\nabla f(x^t), x - x^t\rangle| \leq \frac{\mathcal{K}^t}{2}\|x - x^t\|^2$, which requires no convexity. Its lower half yields

$$\langle\nabla f(x^t), x - x^t\rangle \leq f(x) - f(x^t) + \frac{\mathcal{K}^t}{2}\|x^t - x\|^2 \leq f(x) - f(x^t) + 2\mathcal{K}^t(\eta^t)^2.$$

By Assumption 3.1,

$$f(x) - f(x^t) = f((1 - \lambda\eta^t)x^t + \lambda\eta^t x^\star) - f(x^t) \leq -\lambda\eta^t\Delta^t.$$

Combining these:

$$\langle\nabla f(x^t), x^{t+1} - x^t\rangle \leq -\lambda\eta^t\Delta^t + 2\mathcal{K}^t(\eta^t)^2,$$

and thus

$$\Delta^{t+1} \leq \Delta^t - \lambda\eta^t\Delta^t + 4\mathcal{K}^t(\eta^t)^2.$$

The right hand side is minimized by setting $\eta^t = \frac{\lambda\Delta^t}{8\mathcal{K}^t}$, thus $\Delta^{t+1} \leq \Delta^t$ and

$$\Delta^{t+1} \leq \Delta^t - \frac{\lambda^2(\Delta^t)^2}{16\mathcal{K}^t} \leq \Delta^t - \frac{\lambda^2\Delta^t\Delta^{t+1}}{16\mathcal{K}^t}.$$

The rest of the proof is straightforward.

□

H.3 Proof of Theorem 4.3

Proof. Denote $\mathcal{K}_\xi^t := K_0 + K_1\Delta_\xi^t + K_\rho(\Delta_\xi^t)^\rho$ and $\bar{\mathcal{K}} := K_0 + K_1 M + K_\rho M^\rho$. By the update rule,

$$\|x^{t+1} - x^\star\|^2 = \|x^t - x^\star\|^2 - \frac{2\eta^t}{\|\nabla f_{\xi^t}(x^t)\|}\langle\nabla f_{\xi^t}(x^t), x^t - x^\star\rangle + (\eta^t)^2$$

Applying Corollary H.1 to f_{ξ^t}, which is star-convex about the common point $x^\star$ with $f_{\xi^t}(x^\star) = f_{\xi^t}^\star$ by Assumption 4.1, gives $\langle\nabla f_{\xi^t}(x^t), x^t - x^\star\rangle \geq \Delta_\xi^t$. Together with Assumption 4.2, we have

$$\|x^{t+1} - x^\star\|^2 \leq \|x^t - x^\star\|^2 - \frac{2\eta^t\Delta_\xi^t}{\|\nabla f_{\xi^t}(x^t)\|} + (\eta^t)^2 \leq \|x^t - x^\star\|^2 - \frac{2\eta^t\Delta_\xi^t}{\mathcal{K}_\xi^t\|x^t - x^\star\|} + (\eta^t)^2.$$

For $t = 0$ we have $\|x^0 - x^\star\| = D$, so the display gives $\|x^1 - x^\star\| \leq D$ almost surely. By induction, $\|x^t - x^\star\| \leq D$ for all $t \geq 0$ almost surely. Consequently,

$$\|x^{t+1} - x^\star\|^2 \leq \|x^t - x^\star\|^2 - \frac{2\eta^t\Delta_\xi^t}{D \cdot \mathcal{K}_\xi^t} + (\eta^t)^2 = \|x^t - x^\star\|^2 - \frac{(\Delta_\xi^t)^2}{D^2(\mathcal{K}_\xi^t)^2}.$$

After taking expectation and summing from $t = 0$ to $T - 1$, we obtain

$$\mathbb{E}\big[\|x^T - x^\star\|^2\big] \le D^2 - \frac{1}{D^2} \sum_{t=0}^{T-1} \mathbb{E}\left[\frac{(\Delta_\xi^t)^2}{(\mathcal{K}_\xi^t)^2}\right].$$

Since $\mathbb{E}[\|x^T - x^\star\|^2] \ge 0$, we have $\sum_{t=0}^{T-1} \mathbb{E}[(\Delta_\xi^t)^2/(\mathcal{K}_\xi^t)^2] \le D^4$.

We use the inequality: for real X and positive Y with finite expectations, $\mathbb{E}[X^2/Y] \ge (\mathbb{E}[X])^2/\mathbb{E}[Y]$. By applying this with $X = \Delta_\xi^t$ and $Y = (\mathcal{K}_\xi^t)^2$, we have

$$\mathbb{E}\left[\frac{(\Delta_\xi^t)^2}{(\mathcal{K}_\xi^t)^2}\right] \ge \frac{(\mathbb{E}[\Delta_\xi^t])^2}{\mathbb{E}[(\mathcal{K}_\xi^t)^2]}.$$

Combining with the Cauchy-Schwarz inequality gives

$$\frac{\big(\sum_{t=0}^{T-1} \mathbb{E}[\Delta_\xi^t]\big)^2}{\mathbb{E}\big[\sum_{t=0}^{T-1} (\mathcal{K}_\xi^t)^2\big]} \le D^4.$$

Dividing by T^2 and taking square root yields the general bound:

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\Delta_\xi^t] \le \frac{D^2}{T} \sqrt{\mathbb{E}\Big[\sum_{t=0}^{T-1} (\mathcal{K}_\xi^t)^2\Big]}. \tag{17}$$

If $\Delta_\xi^t \le M$ a.s. for all t, then $(\mathcal{K}_\xi^t)^2 \le \bar{\mathcal{K}}^2$, and thus

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[\Delta_\xi^t] \le \frac{D^2 \bar{\mathcal{K}}}{\sqrt{T}} = \mathcal{O}(T^{-1/2}).$$

□

I Curvature measurements on trajectories without warm-up

This appendix gives the details behind Table 2.

Table 2: Quadratic fit of $\mathcal{K}^t$ against Δ^t on trajectories trained without warm-up or without decay. R^2 is mean $\pm$ standard deviation over runs, computed on $t \ge 256$.

Control	Setup	Runs	R^2
No warm-up, no decay	124M LLaMA/FineWeb, Lion, 3 seeds	3	0.954 ± 0.004
Warm-up 256, then constant	same setup, 3 seeds	3	0.960 ± 0.002
No warm-up, varying lr	lr from $5 \cdot 10^{-5}$ to $2 \cdot 10^{-4}$	4	0.953 ± 0.009
Deterministic, non-LLM	CIFAR-10 CNN, full-batch GD, 3 seeds	3	0.946 ± 0.040

Language model runs. All runs train the 124M LLaMA model of Table 8 on FineWeb with Lion ($\beta_1 = 0.9$, $\beta_2 = 0.99$, weight decay 0.1, gradient clipping 0.5), batch size 64×4 accumulation steps, sequence length 512, for 4096 iterations. The learning rate is 10^{-4} unless stated otherwise. In the no-warm-up runs the learning rate is constant from the first step, with no decay. In the constant-learning-rate control it rises linearly over 256 steps and then stays constant. At every step we compute $\mathcal{K}^t$ on the same mini-batch as described in Appendix D, with $f^\star = 3.5$. The quadratic model $K_0 + K_1\Delta + K_2\Delta^2$ is fitted by least squares on all points with $t \ge 256$, a fixed rule that removes the initialization transient in the same way for every run. Table 3 lists the individual runs and Figure 9 shows the first seed.

Table 3: Per-run R^2 of the quadratic fit of $\mathcal{K}^t$ against Δ^t ($t \geq 256$).

Control	Run	R^2
No warm-up, no decay, lr 10^{-4}	seeds 0, 1, 2	0.9516, 0.9519, 0.9578
Warm-up 256, then constant, lr 10^{-4}	seeds 0, 1, 2	0.9605, 0.9572, 0.9614
No warm-up, no decay, seed 0	lr $5 \cdot 10^{-5}$, 10^{-4}, $1.5 \cdot 10^{-4}$, $2 \cdot 10^{-4}$	0.9506, 0.9516, 0.9641, 0.9437
No warm-up, no decay, $\beta_2 = 0.95$	seed 0	0.9360
CIFAR-10 CNN, full-batch GD, no warm-up	seeds 0, 1, 2	0.9003, 0.9746, 0.9636
CIFAR-10 CNN, full-batch GD, warm-up 256 then constant	seed 2	0.9346

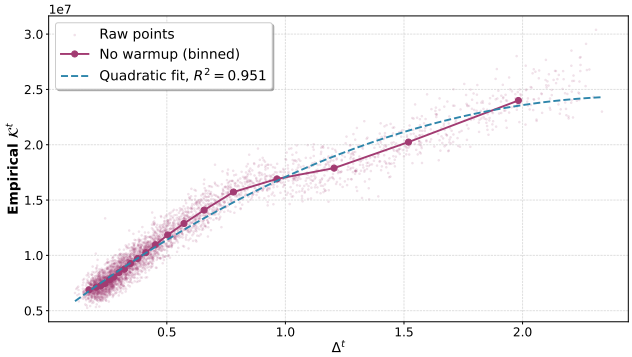


Figure 9: Empirical smoothness ratio $\mathcal{K}^t$ versus Δ^t for 124M LLaMA/FineWeb trained with Lion and no warm-up or decay (seed 0, $t \geq 256$). Points are individual steps, the solid line averages them in 20 quantile bins, and the dashed line is the least-squares quadratic fit.

Non-LLM control. The CIFAR-10 control trains a small convolutional network (two 3×3 convolutions with 32 channels and $\tanh$ activations, average pooling, and a two-layer head) with full-batch gradient descent on a fixed subset of 1280 training images (128 per class), with no data augmentation, learning rate 0.03, $f^\star = 0$ and 4096 iterations. Because the batch is the full fixed dataset, the trajectory is deterministic up to initialization. The fit uses the window $256 \leq t < 2560$, fixed in advance for all runs, after which the training loss is close to zero and the ratio is dominated by numerical effects. Here the update is gradient descent rather than an LMO method, so this control tests the curvature-gap relation of the objective outside language modeling, not the LMO step.

What the measurement shows and what it does not. On all of these trajectories $\mathcal{K}^t$ increases with Δ^t and the quadratic model describes most of its variance. The fitted curvature of the quadratic depends on the visited range, however. On the no-warm-up Lion runs the fitted K_2 is negative (the curve in Figure 9 is concave), and a linear model already reaches $R^2 = 0.92$ on seed 0. Least squares returns a central curve rather than an upper bound, so its coefficients need not satisfy $K_i \geq 0$. Assumption 3.2 asks for a bound by nonnegative constants over the range the trajectory visits, and such constants exist on every trajectory we measured. What these controls establish is therefore that the decrease of curvature with the gap is present without warm-up. The superlinear term is supported by the warm-up trajectories of Figures 1 and 8 and by the example of Appendix J.

The Muon panel. For Muon, the norms in Figure 8 run over all matrix parameters, including the tied embedding and output matrix, which Muon does not update through the oracle but through its internal AdamW. That update is about two orders of magnitude larger than the oracle step and drifts during training, so it dominates the primal norm in the denominator of $\mathcal{K}^t$. Restricting both norms to the parameters governed by the oracle makes the step length constant to within 0.2%, as an LMO step must be, and raises the R^2 of the quadratic fit on a no-warm-up Muon run from 0.75 to 0.86 and 0.87 on two seeds, against 0.61 for a linear model. As for Lion, the fitted K_2 on this short window is negative. The apparent mirroring between the parabolas of Figures 1 and 8 has the same origin: an unconstrained least-squares fit places its vertex wherever it fits best, and whether that vertex falls inside the plotted range changes the visual impression, not the measured ratio.

J A function satisfying Assumption 3.2 with all constants positive

On the open unit ball $\mathcal{X} = \{x \in \mathbb{R}^d : \|x\|_2 < 1\}$ consider

$$f(x) = \left(1 - \|x\|_2^2\right)^{-2} - 1.$$

It is convex, with $f^\star = 0$ attained at $x^\star = 0$. Writing $u = 1 - \|x\|_2^2$, we have $\nabla f(x) = 4xu^{-3}$, $\nabla^2 f(x) = 4u^{-3}I + 24u^{-4}xx^\top$ and $\Delta = u^{-2} - 1$, so in every dimension

$$\left\|\nabla^2 f(x)\right\|_2 = 24(1+\Delta)^2 - 20(1+\Delta)^{3/2}.$$

Hence Assumption 3.2 holds in the Euclidean norm, in the trajectory form of Remark 3.3, with $\rho = 2$, $K_0 = 4$, $K_1 = 18$ and $K_\rho = 24$. This majorant is attained at the minimizer and its ratio to the true curvature tends to one as Δ grows, and we verified it numerically on random points in dimensions 1 to 20. Neither term can be dropped. With $K_1 = 0$ the bound fails near the minimizer, where the curvature grows linearly in Δ, and with $K_\rho = 0$ it fails for large Δ. The two-term conditions of Alimisis et al. [2025] and Liu et al. [2025b] therefore both fail on this function, while the three-term form holds. More generally, $(1 - \|x\|_2^2)^{-a} - 1$ satisfies the assumption with $\rho = 1 + 2/a$, so every $\rho > 1$ is realized. A bounded domain is needed for $\rho > 1$: for $f = \|x\|^p$ one finds $\rho = (p-2)/p < 1$, so polynomially growing objectives never require a superlinear term.

With the constants known, the schedule of Theorem 3.5 can be run with none of the calibration of Section 5. Table 4 compares it with other step-size rules for normalized steepest descent in dimension $d = 10$ from $\|x^0\|_2 = 0.9$ ($\Delta^0 = 26.7$), over 2000 iterations.

Table 4: Iterations needed to reach a given gap on the function above. A dash means the target was not reached within 2000 iterations.

Schedule	$\Delta \le 1$	$\Delta \le 10^{-2}$	$\Delta \le 10^{-4}$
Theorem 3.5, $\eta^t = \Delta^t / (D\,\mathcal{K}(\Delta^t))$	53	83	311
Decay only, curvature frozen at $\mathcal{K}(\Delta^0)$	–	–	–
Constant step, best of 61 values	359	830	893
Step size of Liu et al. [2025b] (gradient descent)	388	444	465

The schedule of Theorem 3.5 reaches every target three to seven times sooner than the best constant step, without the sweep that selected that step. The decay-only schedule ends at $\Delta = 1.04$, against $1.7 \cdot 10^{-6}$ for ours. Below $\Delta \approx 10^{-6}$ the constant step and the step of Liu et al. [2025b] continue to machine precision while ours slows down. This is expected: the minimizer is locally quadratic, our step is proportional to the gap, and our guarantee is $\mathcal{O}(1/T)$ rather than linear.

K Positivity of the calibrated schedule

Substituting $K_0 = K_2(\Delta')^2$ and $K_1 = 1/\mathrm{lr} - 2K_2\Delta'$ from Appendix F.2 into the discriminant of the denominator gives

$$K_1^2 - 4K_0K_2 = \frac{1}{\mathrm{lr}}\left(\frac{1}{\mathrm{lr}} - 4K_2\Delta'\right),$$

which is negative if and only if

$$4\,\Delta^0\Delta'\,(\mathrm{div} - 1) > (\Delta^0 - \Delta')^2.$$

Under this condition $K_0 + K_1\Delta + K_2\Delta^2 > 0$ for every Δ, so the calibrated $\eta(\Delta)$ is positive and finite, even when $K_1 < 0$. Table 5 reports the calibrated values for every configuration in the paper. The condition holds in all of them with a wide margin.

These coefficients are calibration outputs. They are not estimates of the smoothness constants of Assumption 3.2, which are required to be nonnegative only in the analysis.

L Larger-scale experiments

Setup. The 583M model has $n_{\text{layer}} = 11$, $n_{\text{embd}} = 1920$ and $n_{\text{head}} = 15$, and the 720M model has $n_{\text{layer}} = 12$, $n_{\text{embd}} = 2048$ and $n_{\text{head}} = 16$, with the remaining architecture as in Table 8. Both are

Table 5: Calibrated coefficients and the positivity condition. The last column is $4\Delta^0\Delta'(\mathrm{div}-1)/(\Delta^0-\Delta')^2$, which must exceed 1. The configurations with identical inputs (124M and 210M Lion at both batch sizes, normSGD at $\mathrm{bs}=32$) give the same row.

Setup	Optimizer	div	Δ'	K_0	K_1	K_2	condition
124M, bs 256	Muon	100	2.70	$1.09\cdot10^5$	$-8.01\cdot10^4$	$1.49\cdot10^4$	322
124M, bs 32	Muon	100	2.70	$2.18\cdot10^5$	$-1.60\cdot10^5$	$2.98\cdot10^4$	322
124M	Lion	100	0.38	$2.17\cdot10^3$	$-1.03\cdot10^4$	$1.47\cdot10^4$	22.6
124M, bs 256	normSGD	5	2.20	$6.45\cdot10^2$	$-4.85\cdot10^2$	$1.33\cdot10^2$	11.7
124M/210M, bs 32	normSGD	2	3.49	$7.29\cdot10^2$	$-3.17\cdot10^2$	$5.97\cdot10^1$	8.3
210M, bs 256	Muon	100	2.01	$2.97\cdot10^4$	$-2.92\cdot10^4$	$7.36\cdot10^3$	177
210M, bs 32	Muon	100	2.01	$8.90\cdot10^4$	$-8.77\cdot10^4$	$2.21\cdot10^4$	177
210M	Lion	100	0.36	$1.87\cdot10^3$	$-9.31\cdot10^3$	$1.42\cdot10^4$	20.6
210M, bs 256	normSGD	5	2.12	$5.78\cdot10^2$	$-4.44\cdot10^2$	$1.28\cdot10^2$	10.9
583M, bs 256	Muon	10	1.32	$1.36\cdot10^3$	$-1.56\cdot10^3$	$7.80\cdot10^2$	8.2
720M, bs 256	Muon	10	1.76	$2.72\cdot10^3$	$-2.60\cdot10^3$	$8.82\cdot10^2$	12.4

trained on FineWeb with Muon, sequence length 512, batch size 256 (16×16 accumulation steps), peak learning rate $2\cdot10^{-3}$, cosine decay, weight decay 0.1, for 32k iterations (4.2B tokens). The fixed-warm-up sweeps use $\mathrm{div}=100$. The adaptive runs reported in Section 6.1 use $f^\star=3.0$, $\mathrm{div}=10$ and $\sigma_F^2=10^{-3}$.

Table 6: Final validation loss at 583M and 720M parameters. Unless stated otherwise, runs use seed 0.

Schedule	583M	720M
Cosine, warm-up 10	3.038	3.013
Cosine, warm-up 100	3.038	3.015
Cosine, warm-up 300	3.039	3.014
Cosine, warm-up 500	3.039	3.021
Cosine, warm-up 1000	3.048	3.018
Cosine, warm-up 2000	3.047	3.019
Cosine, warm-up 5000	3.051	–
Cosine, warm-up 10, seed 42	3.038	–
Cosine, warm-up 500, seed 42	3.040	–
Cosine, $\mathrm{div}=10$, warm-up 10 / 100 / 1000	–	3.013 / 3.015 / 3.016
Adaptive, $f^\star=3.0$, $\mathrm{div}=10$	3.036	3.011
Adaptive, $f^\star=3.1$, $\mathrm{div}=100$	3.042	3.029
Adaptive, $f^\star=3.3$, $\mathrm{div}=100$	3.041	3.021

Reading the table. Run-to-run variation, measured by the two second-seed reruns at 583M, is about 0.001. With $\mathrm{div}=10$ the adaptive run is the lowest configuration at both sizes, by a margin of 0.002 over the best sweep point. With $\mathrm{div}=100$ it lands inside the short-warm-up plateau at 583M but above it at 720M. The $\mathrm{div}=10$ control at 720M shows that lowering the divisor does not by itself improve the fixed-warm-up schedules. We did not run the same control at 583M. In an earlier 720M sweep at peak learning rate 10^{-3}, an adaptive run with $f^\star=3.0$ and $\mathrm{div}=100$ ended at 3.028, against 3.016 for the best fixed warm-up. We report these runs so that the scale results can be judged as a whole: at these sizes the adaptive rule reliably avoids long-warm-up penalties, while matching the best fixed length depends on the divisor.

M Target-loss sensitivity

At 583M and 720M the sensitivity to $f^\star$ is read from Table 6: changing $f^\star$ from 3.1 to 3.3 at fixed $\mathrm{div}=100$ moves the final loss by 0.001 at 583M and 0.008 at 720M, while the fixed warm-up grid spans 0.013 and 0.008. On Swin-B/ImageNet-1K with Muon, learning rate 10^{-3}, effective batch size 128 and 50k iterations, we vary $f^\star$ around the value 0.40 used in Figure 4b.

Table 7: Top-1 accuracy of Swin-B on ImageNet-1K for different $f^\star$ after 50k iterations.

$f^\star$	0.30	0.35	0.45
Top-1 accuracy (%)	69.02	69.81	69.76

The run with $f^\star = 0.30$ was resumed once from a checkpoint after a memory failure, with the same effective batch size. Across this neighborhood the accuracy varies by 0.8 points. A larger value, $f^\star = 0.50$, did not train stably in a screening run, so the scheduler is robust in a neighborhood of a reasonable estimate and not to arbitrary values.

N Experimental details

Our source code is available at `https://github.com/brain-lab-research/adaptive-warmup`.

Model architecture

All experiments were conducted on NVIDIA H200 GPUs with 140GB memory. Table 8 describes the Llama-based architecture used in our experiments. With $n_{\text{layer}} = 12$, this configuration yields a 124M parameter model. Increasing n_{layer} to 24 produces a 210M parameter model.

Table 8: Model architecture hyperparameters. The configuration yields a 124M parameter Llama model.

Parameter	Value	Description
n_{layer}	12 (for 124M model) 24 (for 210M model)	Number of transformer layers
n_{embd}	768	Embedding dimension
n_{head}	12	Number of attention heads
Vocabulary size	50257	Size of tokenizer vocabulary

N.1 Assumption validation experiments

Table 9 lists the training hyperparameters used for validating the generalized smoothness assumption (Section 3.1). The parameters are identical across all three optimizers (Lion, Muon, normalized SGD) except for the learning rate and the number of training iterations, which are shown in separate columns.

Table 9: Training hyperparameters for assumption validation experiments on FineWeb dataset for Figure 1 (Section 3.1) and Figure 8 (Appendix D).

Parameter	Lion	Muon	normSGD
Learning rate (lr)	10^{-4}	10^{-4}	10^{-3}
Training iterations	10k	15k	10k
Batch size	64		
Sequence length	512		
Gradient accumulation steps	1		
Warmup steps	1000		
Decay scheduler	Cosine		
Gradient clipping	0.5		
Weight decay	0.1		
Momentum	0.9		
Divisor (div)	100		
Target loss $f^\star$	3.2		
Smoothness exponent ρ	2		

The common hyperparameters follow the setup from Semenov et al. [2025]. The learning rates are chosen to ensure comparable training dynamics across optimizers: Lion uses $\mathrm{lr} = 10^{-4}$, while normalized SGD requires $\mathrm{lr} = 10^{-3}$ due to its smaller effective step size in Frobenius norm (see Section 5 for a detailed discussion of geometry-dependent scaling). Muon uses $\mathrm{lr} = 10^{-4}$ to carefully capture the smoothness landscape. The number of training iterations is adjusted to capture sufficient data for fitting the smoothness ratio: Muon requires 15k iterations to reach lower loss values (since it uses quite low lr), while signSGD and normalized SGD converge faster and use 10k iterations.

N.2 Main experiments hyperparameters

Table 10: Training hyperparameters for main experiments on FineWeb dataset for Figures 2 and Figures 3 from Section 6.

Optimizer	Parameter	124M, BS=256	210M, BS=256	124M, BS=32	210M, BS=32
Muon	Learning rate	2×10^{-3}	3×10^{-3}	10^{-3}	10^{-3}
	Target loss $f^\star$	3.3 (1B train tokens), 3.2 (2.1 train tokens)			
	Momentum (β_1)	0.8			
	Divisor (div)	100			
Lion	Learning rate	10^{-3}			
	Target loss $f^\star$	3.4 (1B train tokens), 3.3 (2.1 train tokens)			
	Momentum (β_1)	0.9			
	Second momentum (β_2)	0.99			
	Divisor (div)	100			
normSGD	Learning rate	$\{10^{-4}, 10^{-3}, \mathbf{10^{-2}}, 10^{-1}\}$			
	Target loss $f^\star$	4.3 (1B train tokens), 4.1 (2.1 train tokens)			
	Momentum	$\{0.8, 0.9, \mathbf{0.95}, 0.99\}$			
	Divisor (div)	$\{2, \mathbf{5}, 10, 100\}$		$\{\mathbf{2}, 5, 10, 100\}$	
Common hyperparameters (all optimizers)					
Sequence length		512			
Gradient accumulation		1			
Decay scheduler		Cosine			
Gradient clipping		0.5			
Weight decay		0.1			
Dropout		0.0			
Frobenius variance σ_F^2		10^{-3}			

For Muon and Lion optimizers, we use the hyperparameters from Semenov et al. [2025] without additional tuning. For normalized SGD (normSGD), we perform a separate tuning stage. For compute accounting, we use the following simplified convention: the joint $\mathrm{lr} \times \mathrm{momentum}$ sweep is performed only for the 124M, $\mathrm{bs} = 32$ setup, while the divisor div is tuned only for the 124M setups with $\mathrm{bs} \in \{32, 256\}$. The best configurations are highlighted in bold. Interestingly, we observe that the optimal divisor depends on the batch size: $\mathrm{div} = 5$ performs best for $\mathrm{bs} = 256$, while $\mathrm{div} = 2$ is optimal for $\mathrm{bs} = 32$.

N.3 Vision transformer experiment hyperparameters

Table 11: Training hyperparameters for the Swin-B ImageNet-1K experiment (Section 6.1, Figure 4b). Bold indicates the selected value.

Parameter	Value
Model	Swin-B (swin_base_patch4_window7_224)
Dataset	ImageNet-1K
Optimizer	Muon
Learning rate	$\{5 \times 10^{-4},\ \mathbf{10^{-3}},\ 5 \times 10^{-3}\}$
Batch size	128
Iterations	50,000
Weight decay	0.1
Decay scheduler	Cosine
Adaptive scheduler (ours)	
Frobenius variance σ_F^2	10^{-3}
Smoothness exponent ρ	2
Target loss $f^\star$	0.4

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 list four contributions: a suboptimality-dependent smoothness assumption, a convergence analysis in which warm-up followed by decay emerges from the proof, an adaptive scheduler built on this analysis, and an empirical evaluation on LLaMA pretraining and on Swin-B ImageNet training. Each claim is supported in the corresponding part of the paper: the assumption and its empirical fit in Section 3, the stochastic extension in Section 4, the scheduler in Section 5, and the experiments in Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Limitations section after the conclusion lists the limitations: the need for an estimate of $f^\star$ without a formal accuracy bound, the Euclidean-only stochastic analysis, the trajectory-level and noisy nature of the evidence for Assumption 3.2, the 720M upper end of the model sizes, single-seed sweeps, and the dependence on the divisor at larger scale. The assumptions are stated in Section 3.1, Remark 3.3 specifies the trajectory form in which Assumption 3.2 is used, and Appendices I, L and M report the runs that do not favor the method.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The deterministic assumptions (Assumptions 3.1, 3.2, and 3.4) are stated formally in Section 3.1, and the stochastic-setting variants (Assumptions 4.1 and 4.2) are introduced in Section 4. Theorems 3.5, A.1, and 4.3 are stated in the main text with their assumptions referenced explicitly, and the complete proofs are given in Appendix H.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 6 specifies the model family (LLaMA decoder-only with rotary embeddings, 124M to 720M parameters), the dataset (FineWeb), the optimizers (Muon, Lion, normSGD), the batch sizes, and the warm-up sweep used for each figure. Appendix N contains the full hyperparameter tables (Tables 8, 9, 10, and 11) and the adaptive scheduler is given as Algorithm 1. A public code repository is linked in the abstract and in Appendix N.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: A public code repository is linked in the abstract and in Appendix N. The two datasets used, FineWeb [Penedo et al., 2024] and ImageNet-1K [Deng et al., 2009], are publicly available, and the hyperparameters required to reproduce each figure are listed in the tables of Appendix N.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 6 reports the architecture, datasets, optimizer setup, batch-size and warm-up-length sweeps, and the choice of $f^\star$ for each model size. The full hyperparameter values, including learning rate, weight decay, gradient clipping, decay schedule, and Frobenius variance σ_F^2, are listed in Tables 9, 10, and 11 in Appendix N. The ablation settings for $f^\star$, σ_F^2, and the schedule family are described in Section 6.2.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The warm-up sweeps in Figures 2, 3, 4a and 4b are single-run, without error bars, because each point is a full pretraining run (about 1.2k GPU-hours for the 124M and 210M sweeps alone). To calibrate the margins, we report second-seed reruns at 583M (run-to-run variation about 0.001, Appendix L) and interpret differences of that size as ties. The smoothness controls in Table 2 report mean and standard deviation over three seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix N states that all experiments were run on NVIDIA H200 GPUs with 140GB memory. Section 6 gives an aggregate estimate of about 1.2k GPU-hours for the manual warm-up sweep across model sizes, batch regimes, optimizers, and training budgets, including the additional tuning stage for normSGD.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is theoretical and methodological, uses publicly released datasets (FineWeb and ImageNet-1K) under their stated licenses, and involves no human subjects, scraped private data, or deployed system. We have reviewed the NeurIPS Code of Ethics and find no conflict with this study.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The manuscript does not contain a dedicated broader-impact paragraph. The contribution is a learning rate scheduler for LMO-based optimizers with no application-specific component; the closest direct effect is a reduction in the compute cost of pretraining runs that already use Muon, Lion, or normSGD with manually tuned warm-up.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We release training code only. No pretrained model checkpoints, scraped raw data, or generative artifacts that would warrant misuse safeguards are distributed with this submission.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Datasets are credited with their licenses: FineWeb [Penedo et al., 2024] (ODC-BY 1.0) and ImageNet-1K [Deng et al., 2009] (custom non-commercial research license). Our training harness extends the public `llm-baselines` codebase of Semenov et al. [2025]. The optimizers used in the experiments, namely Muon [Jordan et al., 2024], Lion [Chen et al., 2023], and normSGD [Hazan et al., 2015], together with the Swin-B architecture [Liu et al., 2021], are credited to their original publications and used as described there.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The public code repository linked in Appendix N accompanies the paper with an implementation of Algorithm 1 and scripts for the experiments. Hyperparameters for every reported figure are given in the tables of Appendix N, and the repository README documents the entry points and dependencies.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The study does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research, so IRB (or equivalent) approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Transformer language models are the training target of the proposed scheduler (we apply it to LLaMA pretraining), but no LLM is part of the methodology: the

smoothness assumption, the convergence analysis, the scheduler design, and the evaluation protocol were developed by the authors. Any use of an LLM in writing or editing the manuscript is auxiliary and does not affect the technical content.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.