Deriving Hyperparameter Scaling Laws via Modern Optimization Theory

Egor Shulgin
King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia

Jörg K.H. Franke
University of Freiburg, OpenEuroLLM team at ELLIS Institute Tübingen, Tübingen, Germany

Dimitri von Rütte
ETH Zürich, Zürich, Switzerland

Tianyue H. Zhang
Mila, Quebec AI Institute & Université de Montréal, Quebec, Canada
OpenEuroLLM team at ELLIS Institute Tübingen, Tübingen, Germany

Niccolò Ajroldi
OpenEuroLLM team at ELLIS Institute Tübingen, Tübingen, Germany

Korbinian Pöppel
ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning Johannes Kepler University Linz, Austria

Bernhard Schölkopf
ELLIS Institute Tübingen, MPI-IS, Tübingen AI Center, Tübingen, Germany
ETH Zürich, Zürich, Switzerland

Aaron Klein
OpenEuroLLM team at ELLIS Institute Tübingen, Tübingen, Germany

Peter Richtárik
King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia

Antonio Orvieto ANTONIO@TUE.ELLIS.EU
ELLIS Institute Tübingen, MPI-IS, Tübingen AI Center, Tübingen, Germany

Abstract

Hyperparameter transfer has become an important component of modern large-scale training recipes. Existing methods, such as μP, primarily focus on transfer between model sizes, with transfer across batch sizes and training horizons often relying on empirical scaling rules informed by insights from timescale preservation, quadratic proxies, and continuous-time approximations. We study hyperparameter scaling laws for modern first-order optimizers through the lens of recent convergence bounds for methods based on the Linear Minimization Oracle (LMO), a framework that includes normalized SGD, signSGD (approximating Adam), and Muon. Treating bounds in recent literature as a proxy and minimizing them across different tuning regimes yields closed-form power-law schedules for learning rate, momentum, and batch size as functions of the iteration or token budget. Our analysis, holding model size fixed, recovers several recurring empirical trends under a unified and principled perspective, with clear directions open for future research. Our results draw particular attention to the interaction between momentum and batch-size scaling, suggesting that optimal performance may be achieved with several scaling strategies.

© E. Shulgin, D. von Rütte, T.H. Zhang, N. Ajroldi, B. Schölkopf & A. Orvieto.

1. Introduction

Scaling has become crucial in modern deep learning, with state-of-the-art performance often driven by massive compute: Large language models (LLMs) have reached training budgets of 5×10^{26} FLOPs, with a projected growth of $5\times$ per year [16]. Given the substantial cost, accurately predicting model performance *before* training begins is of critical importance. Historically, scaling laws based on empirical observations have been relied upon to predict the final performance for a given model and dataset size, as well as optimality trends in key hyperparameters, such as batch size and learning rate [7, 24, 25, 30, 34]. Still, training at scale is often more art than science, and many recommendations remain poorly understood or mutually contradictory.

Given the high cost and practical limitations of deriving empirical scaling rules, hyperparameter transfer informed by theory has become a research area of keen interest, with the most prominent method, μP [54], enabling learning rate transfer across model sizes. However, these results typically require a fixed batch size, momentum, and training horizon, leading practitioners to revert to empirical scaling rules, often guided by quadratic analyses [40], stochastic differential equation (SDE) approximations [14, 35, 38], timescale preservation arguments [4, 39], or norm-based views [18].

In this theoretical study, inspired by recent empirical work on optimal hyperparameter scaling as the token budget increases [41, 42, 47, 56], we reexamine scaling laws using performance bounds from optimization theory, leveraging recent advances that extend beyond convex settings and Euclidean geometry [33]. Compared to Bu et al. [11], Schaipp et al. [45], who demonstrate surprising agreement between SGD performance on convex problems and LLM training, our LMO framework aligns theory more closely with modern optimization practice, employing Adam [32] and Muon [29].

We study how learning rate, batch size, and momentum determine the best achievable performance across training horizons and compute budgets at a fixed model size. Our analysis yields explicit scaling predictions for key hyperparameters across training horizons, which have previously been observed only empirically or were not motivated by a unified, theoretically principled setup. Methodologically, we use a standard optimization-theoretic idea—minimizing convergence bounds over algorithmic parameters—and apply it systematically to hyperparameter transfer for LMO-based optimizers.

(i) With fixed momentum, the proxy recovers square-root learning-rate scaling with batch size and predicts $\eta^\star(b,T) \propto b^{1/2}T^{-1/2}$. Jointly tuning (η, b) yields a non-trivial token-optimal batch size $b_T^\star \propto T^{1/2}$, unlike the analogous classical SGD proxy, where the leading dependence on b cancels.

(ii) With fixed batch size, the same $T^{-1/4}$ proxy rate can be recovered by tuning momentum and learning rate. The resulting schedules, $\alpha_T^\star(b) \propto bT^{-1/2}$ and $\eta_T^\star(b) \propto bT^{-3/4}$, align with recent momentum-scaling and timescale-preservation perspectives.

(iii) When (η, α, b) are tuned jointly, the exact proxy selects $b_T^\star \propto T^{1/6}$, $\alpha_T^\star \propto T^{-1/3}$, and $\eta_T^\star \propto T^{-7/12}$. However, this batch law is selected by lower-order terms: many batch-growth paths remain asymptotically near-optimal once (η, α) are retuned.

On the practical side, our results provide insights along the promising direction of momentum scaling, explored in the contemporary literature [17, 39]. On the theoretical side, our work (especially point (iii) above) opens up several directions for research on scaling theory under modified initialization and gradient noise assumptions (App. C). More directly, revised rates and insights can be derived by incorporating weight decay, learning-rate scheduling, and warmup into our analysis.

A note on statistical generalization. Our analysis assumes an unbiased stochastic gradient oracle for the population objective and therefore does not model finite-sample generalization. The proxy should be read as an optimization/stationarity proxy; in small-epoch language-model training, optimization improvements are often empirically correlated with downstream performance [2].

2. Preliminaries

Consider the optimization problem $\min_{x\in\mathbb{R}^d} f(x)$, where we assume access to mini-batch estimates g of the gradient $\nabla f(\cdot)$ [10], with f potentially being non-convex.

Optimizers. Let $b \in \mathbb{N}_{>0}$ be the batch size. We denote by g_b the stochastic gradient of loss f. Fix the stepsize (learning rate) $\eta > 0$ and momentum parameter $\beta := 1-\alpha \in [0,1)$. Following [44], consider a norm $\|\cdot\|$, and the Linear Minimization Oracle (LMO)[1] method[2]:

$$m^{k+1} = (1-\alpha)m^k + \alpha\, g_b^k, \qquad x^{k+1} = x^k + \eta \arg\min_{\|d\|\le 1} \langle m^{k+1}, d\rangle. \tag{1}$$

Choosing $\|\cdot\|$ recovers: (i) Euclidean $\|\cdot\| = \|\cdot\|_2$: normalized SGD with momentum; (ii) $\|\cdot\| = \|\cdot\|_\infty$: signSGD with momentum [6]; (iii) spectral norm: Muon (orthogonalized update) [5, 12, 29].

SignSGD can be easily linked to Adam [32], both theoretically [3, 5] and performance-wise [43, 58]. Many works (e.g. μP derivations; 54) directly derive results using this approximation.

Convergence bounds. Consider a fixed momentum $\beta = 1-\alpha$, batch size b and step size η, and run the algorithm for K iterations. We assume that (1) the gradient noise variance $\mathbb{E}\|g_b - \nabla f\|_2^2$ is upper bounded by a constant σ^2/b, (2) the loss f has L-Lipschitz gradients, with respect to the general $\|\cdot\|$ norm, (3) f is lower bounded by $f^{\inf}$ and $\Delta_0 = f(x^0) - f^{\inf}$.

Let $\|\cdot\|$ be any norm, with dual norm $\|\cdot\|_\star$, Kovalev [33, Theorem 2] proves that under the LMO method (equation 1),

$$\min_{1\le k\le K} \mathbb{E}\left[\|\nabla f(x^k)\|_\star\right] \le \frac{\Delta_0}{\eta K} + \frac{2\rho\sigma}{\alpha\sqrt{b}K} + 2\rho\sigma\sqrt{\frac{\alpha}{b}} + \frac{7L\eta}{2} + \frac{2L\eta}{\alpha}, \tag{2}$$

where $\rho \ge 1$ is a norm equivalence constant (defined from $\|v\|_\star \le \rho\|v\|_2$ which always holds in finite dimensional spaces), dependent on the chosen norm. The right-hand side contains: (i) a purely deterministic optimization term $\Delta_0/(\eta K)$; (ii) momentum "burn-in" / averaging term $2\rho\sigma/(\alpha\sqrt{b}K)$; (iii) a noise floor term $2\rho\sigma\sqrt{\alpha/b}$; (iv) smoothness/trust-region error terms proportional to η, including an η/α coupling.

Relation to the loss. Our proxy controls a dual-gradient norm and should therefore be interpreted primarily as a stationarity proxy in the general non-convex setting. Under additional structure, such as star-convexity in Kovalev [33] or the μ-KL condition $\mu\big(f(x) - f^\star\big) \le \|\nabla f(x)\|_\star$ studied by Islamov et al. [26], dual-gradient-norm control can be converted into function-value control, linking the proxy to training loss. Islamov et al. [26] empirically validate this relation during NanoGPT training by comparing train loss and dual gradient norm. Thus, while our main derivations remain in the general non-convex stationarity setting, their work provides evidence that the same type of proxy can be meaningfully related to loss in structured large-model training regimes.

We focus on the LMO/normalized-gradient framework because it captures norm-based optimizers directly relevant to modern practice, including sign-based abstractions of Adam and Muon/Scion-

1. With a slight abuse of notation for arg min denoting any element from the set.
2. Also known as Unconstrained Stochastic Conditional Gradient method [13, 19, 22, 27, 44]

style operator-norm updates [5, 18, 29, 44]. It is also a setting where momentum has a provably nontrivial role: stochastic normalized updates can require a minimal mini-batch size to converge [23], whereas momentum can remove this requirement [15]. This contrasts with vanilla SGD, where momentum is not known to improve worst-case rates beyond constants [36]; we return to this comparison in Appendix E.

3. Derivation of Scaling Laws

We now treat the right-hand side of equation 2 as a finite-horizon stationarity proxy and minimize it over (η, α, b). Write $C_1 := \Delta_0$, $C_2 := 2\rho\sigma$, and $C_3 := 4L$, so that $\frac{7L}{2}\eta + \frac{2L}{\alpha}\eta \lesssim C_3\eta(1 + 1/\alpha)$. For token budget $T = bK$, substituting $K = T/b$ in the K-horizon proxy gives

$$\mathrm{Rate}_T(\eta, \alpha, b) = C_1 \frac{b}{\eta T} + \frac{C_2}{\sqrt{b}} \frac{b + \alpha^{3/2} T}{\alpha T} + C_3 \eta \left(1 + \frac{1}{\alpha}\right). \tag{3}$$

3.1. Optimization of the Proxy Objective

We first consider a fixed momentum, independent of (b, η, K). In the large-horizon regime $\alpha^{3/2}K \gg 1$, the burn-in part of the stochastic term is dominated, and we can use a simplified proxy:

$$\mathrm{Rate}_K(\eta, b) \approx C_1 \frac{1}{\eta K} + \tilde{C}_2 \frac{1}{\sqrt{b}} + \tilde{C}_3 \eta, \qquad \mathrm{Rate}_T(\eta, b) \approx C_1 \frac{b}{\eta T} + \tilde{C}_2 \frac{1}{\sqrt{b}} + \tilde{C}_3 \eta, \tag{4}$$

where $\tilde{C}_2 := C_2\sqrt{\alpha}$ and $\tilde{C}_3 := C_3(1 + 1/\alpha)$. We have the following result, proved in Appendix B.1.

Theorem 1 (Fixed momentum, large-horizon proxy) *Fix $\alpha \in (0, 1]$ and consider equation 4.*

1. *(Iteration scaling.) For fixed (K, b), with K large, the proxy is minimized by*
$$\eta_K^\star(b) \propto K^{-1/2}, \qquad \mathrm{Rate}_K^\star(b) \propto K^{-1/2} + b^{-1/2}. \tag{5}$$
Thus at fixed K (ignoring token cost), the optimal learning rate is batch size independent, and increasing b improves the bound.

2. *(Token-budget scaling.) For fixed large T, at a fixed batch size b, the optimal learning rate scales as $\eta_T^\star(b) \propto b^{1/2}T^{-1/2}$. Moreover, the joint minimizer $(\eta_T^\star, b_T^\star)$ satisfies*
$$b_T^\star \propto T^{1/2}, \qquad \eta_T^\star(b_T^\star) \propto (b_T^\star)^{1/2}T^{-1/2} \propto T^{-1/4}, \qquad \mathrm{Rate}_T^\star \propto T^{-1/4}. \tag{6}$$
In particular, under a fixed token budget we find a non-trivial token-optimal batch size.

Note that equation 5 shows that for fixed batch size b, optimization can saturate. As shown in equation 6, one can scale b with T to fix this issue. However, there is another option: as discussed in Cutkosky and Mehta [15] and Shulgin et al. [48] – scaling momentum has a similar effect.

Theorem 2 (Fixed batch size, large horizon proxy) *At a fixed batch size b and momentum $\beta = 1 - \alpha$, the optimal learning rate scales with T as $\eta_T^\star(b, \alpha) \propto b^{1/2}\alpha^{1/2}T^{-1/2}$. A subsequent minimization w.r.t. α (at a fixed T and b) then leads to*

$$\alpha_T^\star(b) \propto b \cdot T^{-1/2}, \qquad \eta_T^\star(b) = \eta_T^\star(b, \alpha_T^\star(b)) \propto b \cdot T^{-3/4}, \qquad \mathrm{Rate}_T^\star \propto T^{-1/4}. \tag{7}$$

The proof can be found in Appendix B.2. Our last result considers tuning learning rate, momentum and batch size jointly at a given token budget. The proof, to be found in Appendix B.3, is substantially more involved, since the stochastic burn-in term in equation 3 cannot be dropped.

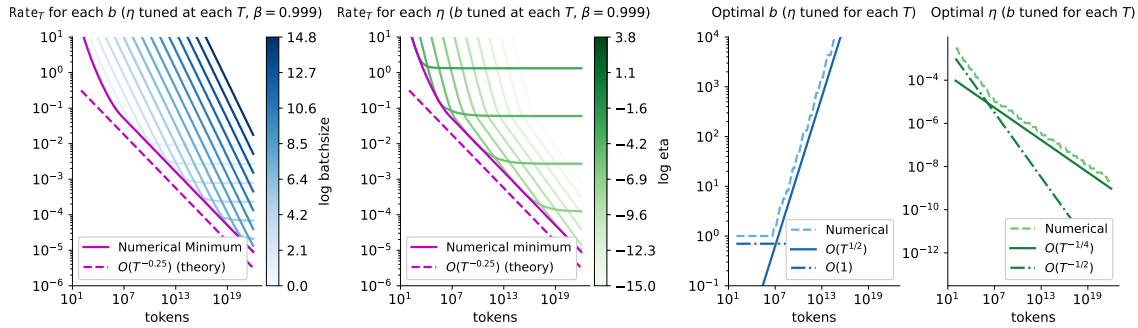

Figure 1: Verification of Theorem 1. Shown are the trends of equation 3 ($C_1 = C_2 = C_3 = 1$) under the choice $\beta = 1 - \alpha = 0.999$. Additional αs can be found in Appendix F. In the two plots on the left, we show in magenta performance at the best value of (η, b) for each token budget, following $\mathcal{O}(T^{-1/4})$. Plotted in blue are also performances for a fixed batch size, minimizing over η at each token budget, and in green performances for a fixed learning rate, minimizing b at each token budget. In the two plots on the right, we show how the optimal batch size and learning rate scale with tokens. The trends predicted by Theorem 1 hold after a burn-in phase, where the optimal batch size is $b = 1$.

Theorem 3 (Jointly tuned (η, α, b) under a fixed token budget) *For large T, minimizing equation 3 over $\eta > 0$, $\alpha \in (0, 1]$, and $b \geq 1$ yields the asymptotic scalings*

$$b_T^\star \propto T^{1/6}, \qquad \eta_T^\star \propto T^{-7/12}, \qquad \alpha_T^\star \propto T^{-1/3}, \qquad \mathrm{Rate}_T^\star \propto T^{-1/4}. \tag{8}$$

Moreover, these schedules are consistent with equation 7 after plugging in $b = b_T^\star$.

The exponent $1/6$ should be read as the batch-growth law selected by lower-order terms; at leading order, several batch-growth paths remain asymptotically equivalent once (α, η) are re-tuned.

3.2. Optimal asymptotics

While Theorem 3 gives the asymptotic minimizer of the exact proxy Rate_T, all three tuning regimes attain the same $\mathrm{Rate}_T^\star \propto T^{-1/4}$ rate. The practical question is therefore whether following Theorem 3 materially improves over simpler scaling rules.

The answer is negative. Appendix B.4 makes this tradeoff precise. If batch size can be scaled freely, retuning α changes the fixed-momentum proxy only by a constant factor: $\mathrm{Rate}_T^\star(\alpha) \propto (1+\alpha)^{1/4} T^{-1/4}$, so the maximal gain is at most $2^{1/4}$. If batch size is capped, however, fixed α leaves a non-vanishing noise floor $\propto \sqrt{\alpha / b_{\max}}$, while the scaling $\alpha_T^\star(b) \propto b T^{-1/2}$ from Theorem 2 removes this floor.

Next, since both scaling b like $T^{1/6}$ and $T^{1/2}$ lead to optimal asymptotics (up to a constant < 1.2), it is natural to ask whether other batch-size scaling laws can still achieve the optimal rate. The answer is positive, with the caveat that some choices may lead to extremely fast (and likely numerically unstable) scaling of momentum or learning rates. The next result is shown in Appendix B.5.

Corollary 4 (Several batch size scalings are near-optimal) *For large T, consider minimizing equation 3 under the choice $b(T) = T^\phi$. If $\phi \leq 1/2$, then the choice $\alpha_T^*(b) \propto b(T) T^{-1/2}$,*

$\eta(T) \propto b(T)T^{-3/4}$ proposed in Theorem 2 leads to a rate $T^{-1/4}$. If instead $\phi \in (1/2, 1)$, then the maximum achievable rate is $\mathrm{Rate}_T \propto b(T)^{1/2}T^{-1/2}$, slower than $T^{-1/4}$.

Finally, we ask: Why does minimizing Rate_T in Theorem 3 predict such a specific scaling? What is special about it? The answer relies on the particular nature of equation 3, comprising terms evolving at different speeds even after optimal tuning: the suboptimality landscape with respect to the variable b, once learning rate and momentum are tuned, is flat, as shown in Appendix C.2. Numerical minimizations of the proxy, including the regimes of Theorems 1–3, are reported in Appendix F.

4. Discussion

Fixed momentum is practically relevant because momentum is often fixed in modern training pipelines, while batch size is constrained by throughput and hardware. The fixed-batch and fixed-momentum regimes therefore give complementary transfer rules: one describes what can still be achieved when batch growth is capped, and the other describes how to scale batch size and learning rate when momentum is held fixed.

The proxy recovers square-root learning-rate scaling with batch size and decreasing learning rates with longer token horizons at fixed batch size. It also explains why momentum tuning matters when batch growth is capped: decreasing α with the token budget removes the fixed-batch noise floor.

Empirical studies sometimes report positive learning-rate exponents when batch size and token budget are increased together. This is a path-conditioned quantity, not an intrinsic $\eta^\star(T)$ exponent. Detailed comparisons are in Appendix D, and the path-conditioned mechanism is discussed in Appendix C.3.

5. Conclusion and limitations

We developed an optimization-theoretic framework for deriving hyperparameter scaling laws at fixed model size. By treating recent LMO convergence bounds as a finite-horizon stationarity proxy, we obtained explicit schedules for learning rate, batch size, and momentum across token budgets. The analysis recovers several recurring empirical trends, including square-root learning-rate scaling with batch size, non-trivial token-optimal batch growth under fixed momentum, and the ability of momentum tuning to substitute for batch growth when batch size is constrained.

Several limitations remain. The theory uses a constant-learning-rate proxy, assumes matched momentum initialization, and does not model warmup, annealing, weight decay, or finite-sample generalization. The results are proxy-optimal rather than direct loss-optimal, and the exponents depend on the assumed $b^{-1/2}$ noise scaling. Extending the framework to include model-size dependence, schedules, weight decay, sharper loss proxies, and a clearer account of assumption mismatch and protocol-dependence is the most direct next step.

Acknowledgments

This research was funded by the European Commission under the grant No. 10119523 (**OpenEuroLLM**). The authors also acknowledge the financial support of the Hector Foundation. Tianyue H. Zhang acknowledges the support of the Natural Sciences and Engineering Research Council of Canada

(NSERC). The research reported in this publication was also supported by funding from King Abdullah University of Science and Technology (KAUST): i) KAUST Baseline Research Scheme, ii) Center of Excellence for Generative AI, under award number 5940, iii) SDAIA-KAUST Center of Excellence in Artificial Intelligence and Data Science.

References

[1] Niccolò Ajroldi. plainlm: Language model pretraining in pytorch. https://github.com/Niccolo-Ajroldi/plainLM, 2024. (Cited on page 30)

[2] Maksym Andriushchenko, Francesco D'Angelo, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning?, 2023. (Cited on page 3)

[3] Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients. In *ICML*, 2018. (Cited on page 3)

[4] Shane Bergsma, Nolan Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Power lines: Scaling laws for weight decay and batch size in llm pre-training. *arXiv preprint arXiv:2505.13738*, 2025. (Cited on page 2)

[5] Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv:2409.20325*, 2024. (Cited on pages 3 and 4)

[6] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *ICML*, 2018. (Cited on page 3)

[7] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. DeepSeek LLM: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024. (Cited on pages 2 and 28)

[8] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *ICML*, 2023. (Cited on page 26)

[9] Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal lr across token horizons. In *The Thirteenth International Conference on Learning Representations*, 2025. (Cited on pages 26 and 28)

[10] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, 2010. (Cited on page 3)

[11] Zhiqi Bu, Shiyun Xu, and Jialin Mao. Convex dominance in deep learning i: A scaling law of loss and learning rate. *arXiv preprint arXiv:2602.07145*, 2026. (Cited on pages 2 and 26)

[12] David Carlson, Volkan Cevher, and Lawrence Carin. Stochastic spectral descent for restricted boltzmann machines. In *AISTATS*, 2015. (Cited on page 3)

[13] Kenneth L Clarkson. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. *ACM Transactions on Algorithms*, 6(4):1–30, 2010. (Cited on page 3)

[14] Enea Monzio Compagnoni, Tianlin Liu, Rustem Islamov, Frank Norbert Proske, Antonio Orvieto, and Aurelien Lucchi. Adaptive methods through the lens of sdes: Theoretical insights on the role of noise. *arXiv:2411.15958*, 2024. (Cited on pages 2 and 26)

[15] Ashok Cutkosky and Harsh Mehta. Momentum improves normalized SGD. In *ICML*, 2020. (Cited on pages 4, 13, and 29)

[16] Epoch AI. Key trends and figures in machine learning, 2023. URL https://epoch.ai/trends. (Cited on page 2)

[17] Damien Ferbach, Courtney Paquette, Gauthier Gidel, Katie Everett, and Elliot Paquette. Logarithmic-time schedules for scaling language models with momentum. *arXiv preprint arXiv:2602.05298*, 2026. (Cited on pages 2 and 27)

[18] Oleg Filatov, Jiangtao Wang, Jan Ebert, and Stefan Kesselheim. Optimal scaling needs optimal norm. *arXiv preprint arXiv:2510.03871*, 2025. (Cited on pages 2, 4, 27, and 28)

[19] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956. (Cited on page 3)

[20] Guillaume Garrigos and Robert M Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*, 2023. (Cited on pages 27 and 29)

[21] Euhanna Ghadimi, Hamid Reza Feyzmahdavian, and Mikael Johansson. Global convergence of the heavy-ball method for convex optimization. In *2015 European control conference (ECC)*, pages 310–315. IEEE, 2015. (Cited on page 29)

[22] Elad Hazan. Sparse approximate solutions to semidefinite programs. In *Latin American Symposium on Theoretical Informatics*, pages 306–316. Springer, 2008. (Cited on page 3)

[23] Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *NeurIPS*, 2015. (Cited on page 4)

[24] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv:2203.15556*, 2022. (Cited on pages 2 and 26)

[25] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. *arXiv:2404.06395*, 2024. (Cited on page 2)

[26] Rustem Islamov, Roman Machacek, Aurelien Lucchi, Antonio Silveti-Falls, Eduard Gorbunov, and Volkan Cevher. On the role of batch size in stochastic conditional gradient methods. *arXiv preprint arXiv:2603.21191*, 2026. (Cited on pages 3 and 23)

[27] Martin Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435. PMLR, 2013. (Cited on page 3)

[28] Stanisław Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017. (Cited on pages 26 and 27)

[29] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cecista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon. (Cited on pages 2, 3, and 4)

[30] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020. (Cited on page 2)

[31] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016. (Cited on page 26)

[32] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (Cited on pages 2, 3, and 28)

[33] Dmitry Kovalev. Understanding gradient orthogonalization for deep learning via non-Euclidean trust-region optimization. *arXiv preprint arXiv:2503.12645*, 2025. (Cited on pages 2, 3, and 27)

[34] Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Zhenyu Ding, Haoying Wang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i, step law – optimal hyperparameter scaling law in large language model pretraining. *arXiv:2503.04715*, 2025. (Cited on pages 2, 25, and 28)

[35] Zhiyuan Li, Sadhika Malladi, and Sanjeev Arora. On the validity of modeling sgd with stochastic differential equations (sdes). *Advances in Neural Information Processing Systems*, 34:12712–12725, 2021. (Cited on pages 2 and 26)

[36] Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum. *Advances in Neural Information Processing Systems*, 33:18261–18271, 2020. (Cited on page 4)

[37] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7. (Cited on page 28)

[38] Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms. *NeurIPS*, 2022. (Cited on pages 2 and 26)

[39] Martin Marek, Sanae Lotfi, Aditya Somasundaram, Andrew Gordon Wilson, and Micah Goldblum. Small batch size training for language models: When vanilla sgd works, and why gradient accumulation is wasteful. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. (Cited on pages 2, 26, and 27)

[40] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv:1812.06162*, 2018. (Cited on pages 2, 14, and 27)

[41] William Merrill, Shane Arora, Dirk Groeneveld, and Hannaneh Hajishirzi. Critical batch size revisited: A simple empirical approach to large-batch language model training. *arXiv:2505.23971*, 2025. (Cited on page 2)

[42] 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. *arXiv:2512.22382*, 2025. (Cited on pages 2, 26, and 27)

[43] Antonio Orvieto and Robert M Gower. In search of adam's secret sauce. *arXiv:2505.21829*, 2025. (Cited on pages 3 and 27)

[44] Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. *arXiv preprint arXiv:2502.07529*, 2025. (Cited on pages 3, 4, and 28)

[45] Fabian Schaipp, Alexander Hägele, Adrien Taylor, Umut Simsekli, and Francis Bach. The surprising agreement between convex optimization theory and learning-rate scheduling for large model training. *arXiv:2501.18965*, 2025. (Cited on page 2)

[46] Christopher J Shallue, Jaehoon Lee, Joseph Antognini, Jascha Sohl-Dickstein, Roy Frostig, and George E Dahl. Measuring the effects of data parallelism on neural network training. *Journal of Machine Learning Research*, 20(112):1–49, 2019. (Cited on page 26)

[47] Xian Shuai, Yiding Wang, Yimeng Wu, Xin Jiang, and Xiaozhe Ren. Scaling law for language models training considering batch size. *arXiv preprint arXiv:2412.01505*, 2024. (Cited on pages 2 and 26)

[48] Egor Shulgin, Sultan AlRashed, Francesco Orabona, and Peter Richtárik. Beyond the ideal: Analyzing the inexact muon update. *arXiv:2510.19933*, 2025. (Cited on pages 4 and 13)

[49] Umut Simsekli, Levent Sagun, and Mert Gurbuzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. *arXiv preprint arXiv:1901.06053*, 2019. (Cited on page 24)

[50] Samuel Smith, Erich Elsen, and Soham De. On the generalization benefit of noise in stochastic gradient descent. In *International Conference on Machine Learning*, pages 9058–9067. PMLR, 2020. (Cited on pages 26 and 27)

[51] Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. Slimpajama: A 627b token cleaned and deduplicated version of redpajama. *Blog post*, 2023. (Cited on page 30)

[52] Teodora Srećković, Jonas Geiping, and Antonio Orvieto. Is your batch size the problem? revisiting the adam-sgd gap in language modeling. *arXiv preprint arXiv:2506.12543*, 2025. (Cited on page 27)

[53] Dimitri von Rütte, Janis Fluri, Omead Pooladzandi, Bernhard Schölkopf, Thomas Hofmann, and Antonio Orvieto. Scaling behavior of discrete diffusion language models. *arXiv:2512.10858*, 2025. (Cited on pages 25, 27, 28, and 35)

[54] Greg Yang and Etai Littwin. Tensor programs ivb: Adaptive optimization in the infinite-width limit. *arXiv preprint arXiv:2308.01814*, 2023. (Cited on pages 2 and 3)

[55] Tianbao Yang, Qihang Lin, and Zhe Li. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *arXiv preprint arXiv:1604.03257*, 2016. (Cited on page 29)

[56] Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham Kakade. How does critical batch size scale in pre-training? *arXiv:2410.21676*, 2024. (Cited on pages 2, 26, and 27)

[57] Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. arXiv:1912.03194. (Cited on page 24)

[58] Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models. *arXiv:2407.07972*, 2024. (Cited on page 3)

[59] Liu Ziyin, Zhikang T Wang, and Masahito Ueda. Laprop: Separating momentum and adaptivity in adam. *arXiv preprint arXiv:2002.04839*, 2020. (Cited on page 28)

APPENDIX

Contents

1 Introduction **2**

2 Preliminaries **3**

3 Derivation of Scaling Laws **4**
3.1 Optimization of the Proxy Objective 4
3.2 Optimal asymptotics 5

4 Discussion **6**

5 Conclusion and limitations **6**

A Budget transfer summary **13**

B Proofs **14**
B.1 Fixed momentum, large budget 14
B.2 Fixed batch size, large budget 15
B.3 Full Tuning, large budget 16
B.4 Should you tune momentum? 18
B.5 General power-law scaling under a fixed budget 20

C How modified assumptions can change the exponents **22**
C.1 Dependency on Initialization 22
C.2 Noise Model and Flatness of the batch size landscape 22
C.3 Why can empirical fits show an optimal learning rate increasing with token budget? 24

D Detailed connections with empirical scaling laws **26**

E Comparison with SGD **29**

F Additional Plots **30**
F.1 Empirical sanity check 30

Appendix A. Budget transfer summary

In this section, we give a practical summary of the takeaways provided by Theorems 1, 2, and 3.

It is often feasible to tune hyperparameters only at a relatively small token budget T_0, but then one wishes to run a much longer training at $T_1 \gg T_0$. Since the proxy bound Rate_T in equation 3 depends explicitly on the horizon $T = bK$, a naive reuse of the short-run optimum (η_0, α_0, b_0) at T_1 is generally suboptimal. A simple alternative is to (i) tune at T_0 (e.g. by grid search) within a chosen hyperparameter family, and (ii) *extrapolate* the best configuration to T_1 using the power-law scalings suggested by the proxy analysis. We consider four natural transfer regimes depending on whether batch size b and momentum α are tuned.

Regime	**Tuned at T_0**	**Transfer rule to T_1**	**Comments**
(A) fixed b, fixed α	η	$b_1 = b_0,\ \alpha_1 = \alpha_0,$ $\eta_1 = \eta_0 \left(\frac{T_0}{T_1}\right)^{1/2}$	Safest; only rescales stepsize.
(B) fixed b, tuned α	(η, α)	$b_1 = b_0,\ \alpha_1 = \alpha_0 \left(\frac{T_0}{T_1}\right)^{1/2},$ $\eta_1 = \eta_0 \left(\frac{T_0}{T_1}\right)^{3/4}$	Fixed-batch large-horizon scaling ($\eta \propto K^{-3/4}$, $\alpha \propto K^{-1/2}$).
(C) tuned b, fixed α	(η, b)	$\alpha_1 = \alpha_0,\ b_1 = b_0 \left(\frac{T_1}{T_0}\right)^{1/2},$ $\eta_1 = \eta_0 \left(\frac{T_0}{T_1}\right)^{1/4}$	Token-optimal b under fixed α proxy (aggressive batch growth).
(D) tuned b, tuned α	(η, α, b)	$b_1 = b_0 \left(\frac{T_1}{T_0}\right)^{1/6},$ $\alpha_1 = \alpha_0 \left(\frac{T_0}{T_1}\right)^{1/3},$ $\eta_1 = \eta_0 \left(\frac{T_0}{T_1}\right)^{7/12}$	Joint token-optimal proxy (burn-in term retained); milder batch growth.

Table 1: Budget transfer rules for LMO methods under token budget $T = bK$. Here (η_0, α_0, b_0) is the best configuration found at T_0 within the chosen regime (e.g. via grid search), and (η_1, α_1, b_1) is the extrapolated configuration for T_1. All scalings are asymptotic and should be combined with feasibility constraints (e.g. $b_1 \geq 1$ integer, hardware caps, $\alpha_1 \in (0, 1]$, and stepsize stability limits).

In terms of the more common momentum coefficient $\beta = 1 - \alpha$, decreasing α as T grows corresponds to increasing momentum $\beta \uparrow 1$. Regime (B) corresponds to the standard fixed-batch large-horizon tuning where the dominant terms of the proxy bound are balanced (see, e.g., [15, 48] for related large-horizon momentum scalings). Regimes (C)–(D) additionally allow batch size to change with T, which is what pins down a token-optimal $b(T)$.

Setting	**Calibrated invariants at** (T_0, b_0)	**Extrapolation to** (T_1, b_1)
LMO, fixed α (Regime A)	$c_\eta := \eta_0\sqrt{\frac{T_0}{b_0}}$	$\eta_1 = c_\eta\sqrt{\frac{b_1}{T_1}} \;=\; \eta_0\sqrt{\frac{b_1}{b_0}}\sqrt{\frac{T_0}{T_1}}$
LMO, tuned α (Regime B)	$c_\alpha := \alpha_0\frac{\sqrt{T_0}}{b_0}, \quad c_\eta := \eta_0\frac{T_0^{3/4}}{b_0}$	$\alpha_1 = c_\alpha\frac{b_1}{\sqrt{T_1}} = \alpha_0\dfrac{b_1}{b_0}\sqrt{\frac{T_0}{T_1}}$ $\eta_1 = c_\eta\frac{b_1}{T_1^{3/4}} = \eta_0\dfrac{b_1}{b_0}\left(\frac{T_0}{T_1}\right)^{3/4}$
SGD	$c_\eta := \eta_0\frac{\sqrt{T_0}}{b_0}$	$\eta_1 = c_\eta\frac{b_1}{\sqrt{T_1}} \;=\; \eta_0\frac{b_1}{b_0}\sqrt{\frac{T_0}{T_1}}$

Table 2: Budget transfer when batch size changes between short-run tuning at (T_0, b_0) and long-run training at (T_1, b_1). Here (η_0, α_0) are obtained by tuning at (T_0, b_0) in the specified setting, and then extrapolated to (T_1, b_1). All formulas are asymptotic and should be combined with feasibility constraints (e.g. $\alpha_1 \in (0, 1]$, integer batch sizes, hardware limits, and stepsize stability caps).

Contrast to SGD. For non-convex Euclidean SGD, the classical bound yields $\eta^\star(T, b) \propto b/\sqrt{T}$ (before stability caps), so a simple budget transfer is $\eta_1 = \eta_0\frac{b_1}{b_0}\sqrt{\frac{T_0}{T_1}}$ (and if $b_1 = b_0$, then $\eta_1 = \eta_0\sqrt{\frac{T_0}{T_1}}$). In this bound, batch size mainly trades iterations for parallelism [40], whereas the LMO bound contains additional batch-dependent terms that can induce a non-trivial token-optimal $b(T)$.

Changing batch size between tuning and the long run. Table 1 assumes the batch size is held fixed between tuning at T_0 and the long run at T_1. If instead the available hardware increases and one runs the long run with a larger batch, Table 2 summarizes the corresponding transfer rules (obtained by re-expressing the K-optimal schedules under $K(T) = T/b(T)$).

Appendix B. Proofs

We provide here proofs for the results in Section 3.

B.1. Fixed momentum, large budget

For a fixed K,

$$\mathrm{Rate}_K(\eta, b) \sim C_1\frac{1}{\eta K} + \tilde{C}_2\frac{1}{\sqrt{b}} + \tilde{C}_3\eta.$$

Minimizing w.r.t. η gives

$$\eta_K^\star = \sqrt{\frac{C_1}{\tilde{C}_3\, K}}.$$

Thus

$$\mathrm{Rate}_K^\star \sim 2\sqrt{\frac{C_1\tilde{C}_3}{K}} + \tilde{C}_2\frac{1}{\sqrt{b}},$$

and the performance therefore improves with b. $b_K^\star \to \infty$.

For a fixed T,

$$\mathrm{Rate}_T(\eta, b) \sim C_1 \frac{b}{\eta T} + \tilde{C}_2 \frac{1}{\sqrt{b}} + \tilde{C}_3 \eta.$$

Minimizing w.r.t. η:

$$\eta_T^\star(b) = \sqrt{\frac{C_1 b}{\tilde{C}_3\, T}}.$$

Plugging in and minimizing w.r.t. b yields

$$b_T^\star = \frac{\tilde{C}_2}{2\sqrt{C_1 \tilde{C}_3}} \sqrt{T}, \qquad \eta_T^\star = \frac{C_1^{1/4} \tilde{C}_2^{1/2}}{\sqrt{2} \tilde{C}_3^{3/4}} \, \frac{1}{T^{1/4}}.$$

Moreover,

$$\mathrm{Rate}_T^\star \sim 2\sqrt{2} (C_1 \tilde{C}_3)^{1/4} \tilde{C}_2^{1/2} \frac{1}{T^{1/4}}.$$

B.2. Fixed batch size, large budget

Write Rate_K in expanded form:

$$\mathrm{Rate}_K(\eta, \alpha, b) = C_1 \frac{1}{\eta K} + C_2 \frac{1}{\alpha \sqrt{b}\, K} + C_2 \sqrt{\frac{\alpha}{b}} + C_3\, \eta \frac{1+\alpha}{\alpha}.$$

In the large-horizon regime and at the optimizer (where α is small), we consider the leading proxy

$$\mathrm{Rate}_K^{\mathrm{lead}}(\eta, \alpha, b) = C_1 \frac{1}{\eta K} + C_2 \sqrt{\frac{\alpha}{b}} + C_3 \frac{\eta}{\alpha}.$$

We check that this proxy is asymptotically correct at the end of the proof.

For fixed (α, b), minimization w.r.t. η leads to

$$C_1 \frac{1}{\eta K} + C_3 \frac{\eta}{\alpha} \quad \Rightarrow \quad \eta^\star(\alpha, b, K) = \sqrt{\frac{C_1 \alpha}{C_3 K}}.$$

Substituting this gives

$$\min_{\eta > 0} \mathrm{Rate}_K^{\mathrm{lead}} = 2\sqrt{\frac{C_1 C_3}{K \alpha}} + C_2 \sqrt{\frac{\alpha}{b}}.$$

Next, we minimize w.r.t. α. Let $p = 2\sqrt{\frac{C_1 C_3}{K}}$ and $q = \frac{C_2}{\sqrt{b}}$. Then we minimize $p\, \alpha^{-1/2} + q\, \alpha^{1/2}$, whose minimizer is $\alpha = p/q$, hence

$$\alpha_K^\star(b) = \frac{2\sqrt{C_1 C_3}}{C_2} \frac{\sqrt{b}}{\sqrt{K}}.$$

Plugging back yields

$$\min_{\alpha, \eta} \mathrm{Rate}_K^{\mathrm{lead}} = 2\sqrt{pq} = 2\sqrt{2}\, C_2^{1/2} (C_1 C_3)^{1/4} \, \frac{1}{b^{1/4} K^{1/4}}.$$

Finally, using $\eta^\star(\alpha, b, K) = \sqrt{\frac{C_1 \alpha}{C_3 K}}$ with $\alpha = \alpha_K^\star(b)$ gives

$$\eta_K^\star(b) = \sqrt{2}\, \frac{C_1^{3/4}}{C_2^{1/2} C_3^{1/4}} \, \frac{b^{1/4}}{K^{3/4}}.$$

For $b = 1$ this recovers $\alpha \propto K^{-1/2}$ and $\eta \propto K^{-3/4}$.

Consistency of dropping lower-order terms. At $(\eta_K^\star, \alpha_K^\star)$, the dropped burn-in term scales as

$$C_2 \frac{1}{\alpha\sqrt{b}\,K} = \mathcal{O}\left(\frac{1}{b\,K^{1/2}}\right),$$

while the dropped additive smoothness term $C_3\eta$ scales as $\mathcal{O}(b^{1/4}K^{-3/4})$, both lower order than the leading $\mathcal{O}(b^{-1/4}K^{-1/4})$ term for large K.

Remark 5 (Continuum of feasible batch-growth paths under Theorem 2) *Although Theorem 2 is stated for fixed b, the schedules are explicit in b: $\alpha_T^\star(b) \propto bT^{-1/2}$ and $\eta_T^\star(b) \propto bT^{-3/4}$ with $\mathrm{Rate}_T^\star \propto T^{-1/4}$. Therefore, along any path $b(T)$ such that $\alpha_T^\star(b(T)) \le 1$ (equivalently $b(T) \lesssim \sqrt{T}$), one retains the same token exponent $T^{-1/4}$, while the induced scalings of α and η depend on the chosen $b(T)$. This "continuum" is broken (and a specific $b_T^\star$ is selected) once the burn-in term is retained, as in Theorem 3.*

B.3. Full Tuning, large budget

The following discussion is self-contained and general enough to also describe the results we presented in simpler settings, as we show after the proof.

Convergence bound. We start from the non-convex bound (up to universal constants)

$$\min_{1\le k\le K} \mathbb{E}\big[\|\nabla f(x^k)\|_\star\big] \;\le\; \underbrace{\frac{\Delta_0}{\eta K} \;+\; \frac{2\rho\sigma}{\alpha\sqrt{b}\,K} \;+\; 2\rho\sigma\sqrt{\frac{\alpha}{b}} \;+\; \frac{7L\eta}{2} \;+\; \frac{2L\eta}{\alpha}}_{=:\,\mathcal{U}(\alpha,\eta;b,K)}. \tag{9}$$

Here $\eta > 0$ is the step size, $\alpha \in (0, 1]$ is the momentum "update" parameter ($\beta = 1 - \alpha$), b is the batch size, and K is the number of iterations.

Token budget. Fix a total budget $T = bK$. Substitute $K = T/b$ into $\mathcal{U}(\alpha, \eta; b, K)$ and define

$$\mathcal{U}_T(\alpha,\eta;b) := \mathcal{U}(\alpha,\eta;b,T/b) = \frac{b\Delta_0}{\eta T} + \frac{2\rho\sigma\sqrt{b}}{\alpha T} + 2\rho\sigma\sqrt{\frac{\alpha}{b}} + \frac{7L\eta}{2} + \frac{2L\eta}{\alpha}. \tag{10}$$

We now minimize equation 10 over (η, α, b) (treating b as a continuous variable; in practice b is an integer and must respect hardware caps).

Step 1: minimize over η (for fixed α, b). For fixed (α, b), the η-dependent part of equation 10 is

$$\frac{b\Delta_0}{T}\cdot\frac{1}{\eta} \;+\; L\Big(\frac{7}{2}+\frac{2}{\alpha}\Big)\eta.$$

This function, with respect to η, is easy to minimize. The minimizer is

$$\eta_T^\star(\alpha,b) \;=\; \sqrt{\frac{b\Delta_0}{T\,L\big(\frac{7}{2}+\frac{2}{\alpha}\big)}} \propto \sqrt{b/T}, \qquad \min_{\eta>0}\Big\{\frac{A}{\eta}+B\eta\Big\} = 2\sqrt{AB}. \tag{11}$$

Plugging equation 11 into equation 10 yields the η-optimized bound

$$\min_{\eta>0}\mathcal{U}_T(\alpha,\eta;b) = 2\sqrt{\frac{b\Delta_0 L}{T}\Big(\frac{7}{2}+\frac{2}{\alpha}\Big)} + \frac{2\rho\sigma\sqrt{b}}{\alpha T} + 2\rho\sigma\sqrt{\frac{\alpha}{b}} =:\; \Phi_T(\alpha,b). \tag{12}$$

Step 2: minimize $\Phi_T(\alpha, b)$ over b (for fixed α). Let $s := \sqrt{b} > 0$. Then equation 12 can be written as

$$\Phi_T(\alpha,b) = A_T(\alpha)\,s \;+\; \frac{B(\alpha)}{s}, \tag{13}$$

where

$$A_T(\alpha) = 2\sqrt{\frac{\Delta_0 L}{T}}\sqrt{\frac{7}{2}+\frac{2}{\alpha}} + \frac{2\rho\sigma}{\alpha T}, \qquad B(\alpha) = 2\rho\sigma\sqrt{\alpha}. \tag{14}$$

Since $As + B/s$ is minimized at $s^\star = \sqrt{B/A}$, we obtain

$$\sqrt{b_T^\star(\alpha)} = \sqrt{\frac{B(\alpha)}{A_T(\alpha)}}, \qquad b_T^\star(\alpha) = \frac{B(\alpha)}{A_T(\alpha)} = \frac{2\rho\sigma\sqrt{\alpha}}{2\sqrt{\frac{\Delta_0 L}{T}}\sqrt{\frac{7}{2}+\frac{2}{\alpha}} + \frac{2\rho\sigma}{\alpha T}}. \tag{15}$$

The corresponding minimized value (for fixed α) is

$$\min_{b>0} \Phi_T(\alpha, b) = 2\sqrt{A_T(\alpha)\, B(\alpha)}. \tag{16}$$

Using $\sqrt{\alpha}\sqrt{\frac{7}{2}+\frac{2}{\alpha}} = \sqrt{\frac{7}{2}\alpha + 2}$, we can simplify:

$$\min_{b>0} \Phi_T(\alpha, b) = 4\sqrt{\rho\sigma\sqrt{\frac{\Delta_0 L}{T}}\sqrt{\frac{7}{2}\alpha+2} + \frac{(\rho\sigma)^2}{T\sqrt{\alpha}}} =: \Psi_T(\alpha). \tag{17}$$

Step 3: minimize $\Psi_T(\alpha)$ over α (exact cubic condition). Because the outer square-root in equation 17 is monotone, minimizing $\Psi_T(\alpha)$ is equivalent to minimizing the inner expression

$$g_T(\alpha) := \rho\sigma\sqrt{\frac{\Delta_0 L}{T}}\sqrt{\frac{7}{2}\alpha+2} + \frac{(\rho\sigma)^2}{T\sqrt{\alpha}}. \tag{18}$$

Differentiate:

$$g_T'(\alpha) = \rho\sigma\sqrt{\frac{\Delta_0 L}{T}} \cdot \frac{\frac{7}{2}}{2\sqrt{\frac{7}{2}\alpha+2}} - \frac{(\rho\sigma)^2}{T} \cdot \frac{1}{2}\alpha^{-3/2}.$$

Setting $g_T'(\alpha) = 0$ and rearranging gives

$$\rho\sigma\sqrt{\frac{\Delta_0 L}{T}} \cdot \frac{\frac{7}{2}}{\sqrt{\frac{7}{2}\alpha+2}} = \frac{(\rho\sigma)^2}{T}\alpha^{-3/2}.$$

Squaring both sides (valid for $\alpha > 0$) yields the *exact cubic*:

$$\left(\frac{7}{2}\right)^2 \Delta_0 L\, T\, \alpha^3 - \left(\frac{7}{2}\right)(\rho\sigma)^2\alpha - 2(\rho\sigma)^2 = 0. \tag{19}$$

This equation has a unique positive real root; denote it by $\alpha_T^\star$. Then $\alpha_T^\star$ is the unique positive root of equation 19, $b_T^\star = b_T^\star(\alpha_T^\star)$ from equation 15, $\eta_T^\star = \eta_T^\star(\alpha_T^\star, b_T^\star)$ from equation 11, $K_T^\star = \frac{T}{b_T^\star}$.

$$\begin{aligned}
\alpha_T^\star &= \text{the unique positive root of equation 19,}\\
b_T^\star &= b_T^\star(\alpha_T^\star) \text{ from equation 15,}\\
\eta_T^\star &= \eta_T^\star(\alpha_T^\star, b_T^\star) \text{ from equation 11,} \quad K_T^\star = T/b_T^\star.
\end{aligned}$$

Asymptotic scalings for large T. We now extract an explicit approximation for $\alpha_T^\star$ from equation 19. Write the cubic in the condensed form

$$A\,T\,\alpha^3 - B\,\alpha - C = 0, \qquad A := \left(\frac{7}{2}\right)^2 \Delta_0 L,\ B := \left(\frac{7}{2}\right)(\rho\sigma)^2,\ C := 2(\rho\sigma)^2. \tag{20}$$

Step (a): identify the correct exponent. Assume α decays polynomially, $\alpha \propto T^{-p}$. Then the three terms scale as

$$AT\alpha^3 \propto T^{1-3p}, \qquad B\alpha \propto T^{-p}, \qquad C \propto T^0.$$

To balance the *constant* term C with the leading term $AT\alpha^3$ we require $1 - 3p = 0$, hence $p = \frac{1}{3}$. This predicts $\alpha_T^\star = \Theta(T^{-1/3})$.

Step (b): compute the leading constant. Set the rescaled variable $u := \alpha\, T^{1/3}$, i.e. $\alpha = u\, T^{-1/3}$. Plugging into equation 20 gives

$$A\,T\,(u^3 T^{-1}) \;-\; B\,(uT^{-1/3}) \;-\; C \;=\; 0 \quad\Longleftrightarrow\quad Au^3 \;-\; C \;=\; Bu\,T^{-1/3}.$$

As $T \to \infty$, the right-hand side vanishes, so u converges to the unique positive root of $Au^3 - C = 0$, i.e. $u_0 = (C/A)^{1/3}$. Therefore,

$$\alpha_T^\star \;\approx\; u_0\, T^{-1/3} = \left(\frac{C}{A}\right)^{1/3} T^{-1/3} = \left(\frac{2(\rho\sigma)^2}{\left(\frac{7}{2}\right)^2 \Delta_0 L}\right)^{1/3} T^{-1/3} = \frac{2}{7^{2/3}}\,\frac{(\rho\sigma)^{2/3}}{(\Delta_0 L)^{1/3}}\, T^{-1/3}. \tag{21}$$

First correction term. The same rescaling gives $Au^3 - C = Bu\,T^{-1/3}$, so one may expand $u = u_0 + u_1 T^{-1/3} + \mathcal{O}(T^{-2/3})$. Keeping the order-$T^{-1/3}$ terms yields $3Au_0^2 u_1 = Bu_0$, hence

$$u_1 = \frac{B}{3Au_0} = \frac{B}{3A^{2/3}C^{1/3}}, \quad \text{and thus} \quad \alpha_T^\star = u_0 T^{-1/3} + u_1 T^{-2/3} + \mathcal{O}(T^{-1}).$$

Step (c): induced batch size and step-size scalings. Using equation 15 and the fact that $\alpha_T^\star \to 0$, we have $\frac{7}{2} + \frac{2}{\alpha} \sim \frac{2}{\alpha}$, and also the term $\frac{2\rho\sigma}{\alpha T}$ in $A_T(\alpha)$ becomes lower order at $\alpha = \alpha_T^\star$. A short calculation then yields the scalings

$$b_T^\star = \Theta(T^{1/6}), \qquad K_T^\star = \Theta(T^{5/6}), \qquad \eta_T^\star = \Theta(T^{-7/12}), \tag{22}$$

and substituting the optimized parameters back into equation 17 gives the rate

$$\min_{1 \le k \le K_T^\star} \mathbb{E}\big[\|\nabla f(x^k)\|_\star\big] = \mathcal{O}\Big((\Delta_0 L)^{1/4}\sqrt{\rho\sigma}\; T^{-1/4}\Big).$$

Recovering earlier regimes as special cases. All previously discussed tuning regimes are obtained by *restricting* the optimization above:

- *Fixed b (no batch tuning):* keep b fixed in equation 10 and optimize only over (η, α) (equivalently, apply Steps 1 and 3 but without Step 2). This recovers the familiar large-horizon schedules $\alpha \propto K^{-1/2}$ and $\eta \propto K^{-3/4}$ (up to b-dependent constants) once one rewrites $K = T/b$.
- *Fixed α (no momentum tuning):* keep α fixed and optimize over (η, b). Then Step 3 is skipped and the minimizer in Step 2 yields the "fixed-α" token-optimal batch scaling (typically $b_T^\star = \Theta(\sqrt{T})$ in the simplified proxy).
- *Fixed K:* set $b = T/K$ (a re-parameterization) and optimize over (η, α).

B.4. Should you tune momentum?

Corollary 6 (Momentum tuning and batch size constraints) *As the token budget $T \to \infty$, we have the following properties following directly from the proofs of Theorems 1 and 2.*

1. *Assume optimal tuning of batch size and learning rate for a fixed arbitrary momentum $\beta = 1 - \alpha$. We have that $\mathrm{Rate}_T^\star(\alpha) \propto (1+\alpha)^{1/4} T^{-1/4}$. Hence, re-tuning α can only lead to a $2^{1/4} \approx 1.19$ improvement in the rate constant.*

2. *If instead the batch size is capped by hardware, $b \le b_{\max}$, then optimal tuning of the learning rate at an arbitrary momentum $\beta = 1 - \alpha$ leads to $\mathrm{Rate}_T \propto \alpha^{1/2} b_{\max}^{-1/2}$ – a non-vanishing noise floor. Allowing α to decrease with T (Theorem 2) removes this floor and restores $\mathrm{Rate}_T^\star \propto T^{-1/4}$ even at fixed batch size.*

Our results show that a rate of $T^{-1/4}$ can be achieved both with and without momentum tuning. This naturally raises the question: if we keep α fixed when moving from T_0 to $T_1 \gg T_0$, **how far are we from the token-optimal performance that one would obtain by re-tuning the momentum parameter** α? We develop this in a few points, proving Corollary 6.

(1) Performance gap at the proxy level is only a constant factor (when b can scale). In the fixed-momentum large-horizon proxy,

$$\mathrm{Rate}_T(\eta, b; \alpha) \;\approx\; C_1 \frac{b}{\eta T} + \tilde{C}_2(\alpha) \frac{1}{\sqrt{b}} + \tilde{C}_3(\alpha)\eta, \qquad \tilde{C}_2(\alpha) = C_2\sqrt{\alpha}, \quad \tilde{C}_3(\alpha) = C_3\Big(1 + \frac{1}{\alpha}\Big).$$

Optimizing over (η, b) for fixed α yields (App. B.1):

$$b_T^\star(\alpha) = \frac{\tilde{C}_2(\alpha)}{2\sqrt{C_1 \tilde{C}_3(\alpha)}} \sqrt{T}, \tag{23}$$

$$\eta_T^\star(\alpha) = \frac{C_1^{1/4} \tilde{C}_2(\alpha)^{1/2}}{\sqrt{2}\, \tilde{C}_3(\alpha)^{3/4}}\, T^{-1/4}, \tag{24}$$

$$\mathrm{Rate}_T^\star(\alpha) = 2\sqrt{2}\, (C_1 \tilde{C}_3(\alpha))^{1/4} \tilde{C}_2(\alpha)^{1/2}\, T^{-1/4}. \tag{25}$$

Substituting $\tilde{C}_2, \tilde{C}_3$ gives the explicit α-dependence of the *constant*:

$$\mathrm{Rate}_T^\star(\alpha) = 2\sqrt{2}\, (C_1 C_3)^{1/4} C_2^{1/2}\, (1+\alpha)^{1/4}\, T^{-1/4}. \tag{26}$$

Thus, in this proxy regime, keeping α fixed does *not* change the exponent in T (it remains $T^{-1/4}$), and re-tuning α **can only improve the *constant factor***. In fact, since $\alpha \in (0, 1]$, one has

$$(1+\alpha)^{1/4} \in [1, 2^{1/4}],$$

so the maximal improvement from changing a fixed α is at most a factor $2^{1/4} \approx 1.19$ in performance. Equivalently, since T enters as $T^{-1/4}$, this constant-factor improvement corresponds to at most a factor of 2 in token budget to reach a fixed target tolerance ε:

$$T_{\mathrm{req}}(\alpha; \varepsilon) \propto (1+\alpha) \cdot \varepsilon^{-4}.$$

For typical momentum values (e.g. $\beta = 0.9 \Rightarrow \alpha = 0.1$), this predicts only a mild potential gain in the proxy bound from re-tuning α at larger budgets.

(2) The gap can be huge under a batch-size cap: fixed α causes saturation. The previous conclusion assumes one can scale b with T as in equation 23. If instead the batch size is capped by hardware, $b \le b_{\max}$, then the fixed-α proxy optimized over η satisfies

$$\min_{\eta > 0} \mathrm{Rate}_T(\eta, b_{\max}; \alpha) \;\approx\; 2\sqrt{\frac{C_1 \tilde{C}_3(\alpha)\, b_{\max}}{T}} \;+\; \tilde{C}_2(\alpha) \frac{1}{\sqrt{b_{\max}}}.$$

As $T \to \infty$, the first term vanishes but the second term remains:

$$\liminf_{T \to \infty} \min_{\eta > 0} \mathrm{Rate}_T(\eta, b_{\max}; \alpha) \;\gtrsim\; \frac{\tilde{C}_2(\alpha)}{\sqrt{b_{\max}}} \;=\; \frac{C_2\sqrt{\alpha}}{\sqrt{b_{\max}}}.$$

That is, *with fixed α and capped batch size, the proxy exhibits a non-vanishing noise floor.* By contrast, allowing α to decrease with T (Regime (B) / Theorem 2 in the main text) removes this floor and restores $\mathrm{Rate}_T^\star \propto T^{-1/4}$ even at fixed b.

(3) What is left on the table is primarily* batch growth *(Regime (C) vs (D)). Comparing the token-optimal batch scaling under fixed α (Theorem 1: $b_T^\star \propto T^{1/2}$) to the jointly tuned scaling (Theorem 3 : $b_T^\star \propto T^{1/6}$) shows that re-tuning α mostly reduces the required batch growth with budget. A convenient way to quantify this is via the maximal budget that can be run *near-optimally* under a batch-size cap $b \le b_{\max}$:

$$\text{Regime (C):} \quad b_T^\star \propto T^{1/2} \;\Rightarrow\; T \lesssim \Theta(b_{\max}^2), \qquad \text{Regime (D):} \quad b_T^\star \propto T^{1/6} \;\Rightarrow\; T \lesssim \Theta(b_{\max}^6),$$

(up to constant factors hidden in the proxy). Hence, even though both regimes achieve the same proxy exponent $T^{-1/4}$ in principle, *tuning momentum can dramatically extend the range of token budgets that remain feasible before hitting batch-size saturation.*

B.5. General power-law scaling under a fixed budget

In this section, we prove Corollary 4. Recall that under the token (samples) budget $T = bK$ we can rewrite the bound as

$$\mathcal{U}_T(\alpha, \eta; b) := \mathcal{U}\left(\alpha, \eta; b, \frac{T}{b}\right) = \frac{b\Delta_0}{\eta T} + \frac{2\rho\sigma\sqrt{b}}{\alpha T} + 2\rho\sigma\sqrt{\frac{\alpha}{b}} + \frac{7L\eta}{2} + \frac{2L\eta}{\alpha}. \tag{27}$$

Power-law schedules. Assume that the algorithmic parameters follow power laws in T:

$$b(T) = \Theta(T^\beta), \qquad \alpha(T) = \Theta(T^{-\gamma}), \qquad \eta(T) = \Theta(T^{-\delta}), \tag{28}$$

with $\beta \in [0, 1]$ (since $1 \le b \le T$) and $\gamma, \delta \in \mathbb{R}$. *Note that here β is not the momentum parameter!*

Then the five terms in equation 27 scale as

$$\begin{aligned}
\frac{b\Delta_0}{\eta T} &= \Theta\Big(T^{\beta-1+\delta}\Big) = \Theta\Big(T^{-(1-\beta-\delta)}\Big), \\
\frac{2\rho\sigma\sqrt{b}}{\alpha T} &= \Theta\Big(T^{\beta/2-1+\gamma}\Big) = \Theta\Big(T^{-(1-\beta/2-\gamma)}\Big), \\
2\rho\sigma\sqrt{\frac{\alpha}{b}} &= \Theta\Big(T^{-(\beta+\gamma)/2}\Big), \\
\frac{7L\eta}{2} &= \Theta\Big(T^{-\delta}\Big), \\
\frac{2L\eta}{\alpha} &= \Theta\Big(T^{-\delta+\gamma}\Big) = \Theta\Big(T^{-(\delta-\gamma)}\Big).
\end{aligned} \tag{29}$$

Equivalently, defining the decay exponents

$$r_1 := 1 - \beta - \delta, \quad r_2 := 1 - \frac{\beta}{2} - \gamma, \quad r_3 := \frac{\beta + \gamma}{2}, \quad r_4 := \delta, \quad r_5 := \delta - \gamma, \tag{30}$$

we have (up to absolute constants)

$$\mathcal{U}_T(\alpha(T), \eta(T); b(T)) = \Theta\Big(T^{-r_1} + T^{-r_2} + T^{-r_3} + T^{-r_4} + T^{-r_5}\Big) = \Theta\Big(T^{-\min_i r_i}\Big), \tag{31}$$

provided all $r_i > 0$ (i.e., each term decays).

Guaranteeing a $T^{-1/4}$ bound for a prescribed batch scaling. A convenient way to enforce a $T^{-1/4}$ rate is to equalize the three coupled terms $\frac{b}{\eta T}$, $\sqrt{\frac{\alpha}{b}}$, $\frac{\eta}{\alpha}$. Specifically, for any batch schedule $b(T)$ satisfying

$$b(T) \;\le\; c\sqrt{T} \qquad (\text{equivalently } \beta \le \tfrac{1}{2}), \tag{32}$$

choose

$$\alpha(T) \propto \frac{b(T)}{\sqrt{T}}, \qquad \eta(T) \propto \frac{b(T)}{T^{3/4}}. \tag{33}$$

Then

$$\frac{b}{\eta T} \;=\; \Theta(T^{-1/4}), \qquad \sqrt{\frac{\alpha}{b}} \;=\; \Theta(T^{-1/4}), \qquad \frac{\eta}{\alpha} \;=\; \Theta(T^{-1/4}), \tag{34}$$

and the remaining terms satisfy

$$\frac{\sqrt{b}}{\alpha T} = \Theta\Big(\frac{1}{\sqrt{bT}}\Big) \le \Theta(T^{-1/2}), \qquad \eta = \Theta\Big(\frac{b}{T^{3/4}}\Big) \le \Theta(T^{-1/4}) \quad (\text{by equation 32}). \tag{35}$$

Consequently,

$$\mathcal{U}_T(\alpha(T), \eta(T); b(T)) \;\lesssim\; C_1\, T^{-1/4} \;+\; C_2\, (bT)^{-1/2} \;=\; \mathcal{O}(T^{-1/4}), \tag{36}$$

for constants C_1, C_2 depending only on $\Delta_0, L, \rho, \sigma$ (and the hidden constants in equation 33).

Remark (batch-size scaling $b = T/K$). If $b(T) = \Theta(T^{\beta})$, then $K(T) = T/b(T) = \Theta(T^{1-\beta})$ and $T/K = b = \Theta(T^{\beta})$. Condition equation 32 is precisely $\beta \le \frac{1}{2}$ (i.e., b cannot grow faster than $\sqrt{T}$) to maintain the $T^{-1/4}$ guarantee under power-law tuning.

Aggressive batch-size growth: $\frac{1}{2} < \beta < 1$. Assume the power-law batch schedule $b(T) = \Theta(T^{\beta})$ with $\frac{1}{2} < \beta < 1$ (hence $K(T) = T/b(T) = \Theta(T^{1-\beta})$). In this regime, the bound cannot in general maintain the $T^{-1/4}$ decay: the two terms $\frac{b}{\eta T}$ and η already impose a hard rate ceiling. Indeed, writing $\eta(T) = \Theta(T^{-\delta})$, their decay exponents are $r_1 = 1 - \beta - \delta$ and $r_4 = \delta$, so for any δ,

$$\min\{r_1, r_4\} \;\le\; \max_{\delta} \min\{1 - \beta - \delta, \delta\} \;=\; \frac{1-\beta}{2}, \tag{37}$$

with equality achieved by balancing them at

$$\delta^{\star} = \frac{1-\beta}{2} \qquad \Longleftrightarrow \qquad \eta(T) = \Theta\Big(T^{-(1-\beta)/2}\Big). \tag{38}$$

Taking, e.g., $\alpha(T) = \Theta(1)$ and $\eta(T)$ as in equation 38 yields

$$\mathcal{U}_T\big(\alpha(T), \eta(T); b(T)\big) = \Theta\Big(T^{-(1-\beta)/2}\Big), \qquad \frac{1}{2} < \beta < 1, \tag{39}$$

since the remaining terms decay strictly faster:

$$\frac{\sqrt{b}}{\alpha T} = \Theta\Big(T^{-(1-\beta/2)}\Big), \qquad \sqrt{\frac{\alpha}{b}} = \Theta\Big(T^{-\beta/2}\Big), \qquad \frac{\eta}{\alpha} = \Theta\Big(T^{-(1-\beta)/2}\Big).$$

Equivalently, using $K(T) = T/b(T) = \Theta(T^{1-\beta})$, the achievable rate can be written as

$$\Theta\Big(T^{-(1-\beta)/2}\Big) \;=\; \Theta\Big(K^{-1/2}\Big) \;=\; \Theta\left(\sqrt{\frac{b}{T}}\right), \tag{40}$$

highlighting that when b grows faster than $\sqrt{T}$, the bound becomes iteration-limited.

Appendix C. How modified assumptions can change the exponents

C.1. Dependency on Initialization

Matched initialization and the burn-in term. We assume *matched* initialization, namely that the momentum buffer is initialized from a stochastic mini-batch gradient of the *same* batch size b as used during training, e.g. $m^0 = g_b^0$. Under the bounded-variance model, this implies

$$E_0 \;:=\; \mathbb{E}\big[\|g_b^0 - \nabla f(x^0)\|_\star\big] \;\lesssim\; \frac{\rho\sigma}{\sqrt{b}},$$

and this is exactly what yields the $\frac{1}{\alpha\sqrt{b}K}$ "burn-in" dependence in the non-convex LMO bound below. If m^0 is not matched, this term changes; see next paragraph.

On non-matched initialization. If m^0 is *not* initialized from a stochastic batch-b gradient, the burn-in term can change from $\propto \frac{1}{\alpha\sqrt{b}K}$ to $\propto \frac{E_0}{\alpha K}$ with an E_0 that may not decay as $1/\sqrt{b}$. Under a fixed budget $T = bK$ this changes the $\sqrt{b}/(\alpha T)$ structure in equation 10 and can alter the token-optimal $b(T)$ scaling.

C.2. Noise Model and Flatness of the batch size landscape

The power-law exponents obtained by optimizing convergence upper bounds are not universal: they are a direct consequence of (i) the functional form of the bound and (ii) how the stochastic error term scales with mini-batching, geometry, and moment assumptions. We record a simple sensitivity analysis that makes the dependence explicit.

I. A one-parameter model for how noise shrinks with batch size. The equation 2 bound assumes a *finite-variance* (sub-Gaussian/sub-exponential) scaling, where the typical noise magnitude decreases as $b^{-1/2}$. To capture deviations (e.g. correlations or heavy tails), we introduce a generic exponent q:

Assumption 7 (Effective mini-batch noise scaling) *There exist $q \in (0, 1]$ and a scale parameter $\sigma_q > 0$ such that the mini-batch gradient satisfies*

$$\mathbb{E}\big[\|g_b(x) - \nabla f(x)\|_\star\big] \;\lesssim\; \frac{\sigma_q}{b^q} \qquad \textit{(uniformly over } x\textit{).}$$

The bounded-variance/i.i.d. setting corresponds to $q = \frac{1}{2}$. Correlations or other inefficiencies can lead to $q < \frac{1}{2}$.

Under Assumption 7, the two "noise" terms in the LMO bound scale as

$$\frac{1}{\alpha K} \cdot \frac{\sigma_q}{b^q} \qquad \text{and} \qquad \sqrt{\alpha} \cdot \frac{\sigma_q}{b^q},$$

rather than with $b^{-1/2}$.

II. Why the leading-order bound can become "flat" in b (and when it does not). To isolate the mechanism, consider the dominant three-term proxy (dropping constants and the burn-in term for the moment)

$$\mathcal{U}(\alpha, \eta; b, K) \;\approx\; \frac{\Delta_0}{\eta K} \;+\; L\frac{\eta}{\alpha} \;+\; \frac{\sigma_q}{b^q}\sqrt{\alpha}. \tag{41}$$

Optimizing equation 41 over η gives

$$\eta^\star(\alpha) \;\propto\; \sqrt{\frac{\Delta_0\,\alpha}{L\,K}}, \qquad \min_{\eta>0}\Big\{\frac{\Delta_0}{\eta K} + L\frac{\eta}{\alpha}\Big\} \;\propto\; \sqrt{\frac{\Delta_0 L}{K}} \cdot \alpha^{-1/2}.$$

Then the α-problem becomes $c_1\alpha^{-1/2} + c_2(b)\alpha^{1/2}$ with $c_2(b) \propto \sigma_q/b^q$, hence

$$\alpha^\star(b,K) \;\propto\; \frac{b^q}{\sqrt{K}}, \qquad \eta^\star(b,K) \;\propto\; \frac{b^{q/2}}{K^{3/4}}, \qquad \min_{\alpha,\eta}\mathcal{U} \;\propto\; \frac{b^{-q/2}}{K^{1/4}}. \tag{42}$$

Now impose a fixed token budget $T = bK$ (so $K = T/b$). Plugging into equation 42 yields

$$\min_{\alpha,\eta}\mathcal{U} \;\propto\; T^{-1/4}\, b^{\,1/4-q/2}. \tag{43}$$

Interpretation of equation 43.

- If $q = \frac{1}{2}$ (bounded variance, i.i.d. mini-batching), then $1/4 - q/2 = 0$ and *the leading-order dependence on b cancels.* This is precisely the "flatness in b" phenomenon: once η and α are retuned for each b, the dominant term depends primarily on $T = bK$ rather than on b itself.
- If $q < \frac{1}{2}$ (noise shrinks *slower* than $b^{-1/2}$), then $1/4 - q/2 > 0$ and the leading term *increases* with b; the token-optimal batch size is pushed toward the smallest feasible b.
- If $q > \frac{1}{2}$ (noise shrinks *faster* than $b^{-1/2}$), then $1/4 - q/2 < 0$ and larger batches become beneficial already at leading order.

Therefore, the existence (and scaling) of an *interior* token-optimal batch size is *not robust*: it relies on the finite-variance $q = \frac{1}{2}$ law, plus lower-order terms (e.g. burn-in) that break the leading-order cancellation. This explains why the joint optimum $b_T^\star = \Theta(T^{1/6})$ should not be read as the only viable scaling: for bounded-variance noise, many choices of $b(T)$ remain near-optimal once $\alpha(T)$ is tuned (momentum compensates for smaller batches), and the burn-in term selects $b_T^\star$ only through lower-order effects.

Batch size versus sequence length. In our main proxy we write the mini-batch size as a single scalar b, implicitly treating it as the number of tokens or samples used to estimate the stochastic gradient. In language-model training, however, the per-step token count typically factors as

$$b_{\text{tok}} = BS,$$

where B is the number of sequences and S is the sequence length. The standard bounded-variance model corresponds to

$$\mathrm{Var}(g_{B,S}) \propto \frac{1}{BS}, \qquad \text{or equivalently} \qquad \|g_{B,S} - \nabla f\| \sim (BS)^{-1/2}.$$

Recent work by Islamov et al. [26] explicitly studies this distinction. They assume $\sigma^2 = \sigma_\star^2/(BS)$, and empirically fit gradient-variance power laws separately in B and S, finding exponents close to, but not exactly, one. This suggests the more general model

$$\mathrm{Var}(g_{B,S}) \;\propto\; \frac{1}{B^{\lambda_B} S^{\lambda_S}}, \qquad \|g_{B,S} - \nabla f\| \;\propto\; B^{-\lambda_B/2} S^{-\lambda_S/2}.$$

Repeating the leading-order calculation from equation 43 with $T = KBS$ yields

$$\min_{\alpha,\eta} U \;\propto\; T^{-1/4}\, B^{(1-\lambda_B)/4}\, S^{(1-\lambda_S)/4}.$$

Thus, when $\lambda_B = \lambda_S = 1$, the leading dependence on the split between B and S cancels, matching the usual $1/(BS)$ variance model. If $\lambda_B \neq \lambda_S$, however, the theory can prefer increasing batch size and sequence length at different rates. This provides another mechanism by which empirical batch-size exponents can differ from the scalar-b predictions in the main text.

III. Heavy-tailed noise: both the b-law and the T-exponent can change. A common empirical observation in deep learning is that stochastic gradient noise can be heavy-tailed, often modeled via α-stable laws [49, 57]; in such regimes the variance may be infinite and the classical $b^{-1/2}$ scaling can fail. In a stylized $\mathfrak{p}$-moment model with $\mathfrak{p} \in (1, 2)$, a typical scaling for sample averages is

$$\text{noise magnitude} \;\propto\; b^{-(1-1/\mathfrak{p})}, \quad \text{i.e.} \quad q = 1 - \frac{1}{\mathfrak{p}} < \frac{1}{2}.$$

Plugging into equation 43 gives

$$\min_{\alpha,\eta} \mathcal{U} \;\propto\; T^{-1/4}\, b^{-1/4+1/(2\mathfrak{p})},$$

which increases with b for any $\mathfrak{p} < 2$, again pushing the token-optimal b toward small batches.

Moreover, under heavy-tailed noise the *optimal* convergence rate in T can itself differ from $T^{-1/4}$ (the finite-variance case), and depends on $\mathfrak{p}$. This indicates that mismatches between empirical scaling-law fits and finite-variance theory can reflect a genuinely different regime rather than just loose constants.

IV. "Variance" in non-Euclidean methods: which norm matters. For LMO/Muon, the stochastic terms are naturally expressed in the dual norm $\|\cdot\|_\star$. Accordingly, a more intrinsic noise proxy is

$$\sigma_\star^2 \;:=\; \sup_x \mathbb{E}\big[\|g(x) - \nabla f(x)\|_\star^2\big],$$

rather than an ℓ_2-variance. Using norm compatibility one can always upper bound $\|v\|_\star \le \rho\|v\|_2$ and therefore $\sigma_\star \le \rho\sigma_2$, but this can be loose and may hide dimension/model-size dependence in ρ (or in $\sigma_\star$ itself). Hence, changing the *noise model* from an ℓ_2 variance bound to a $\|\cdot\|_\star$-variance bound can change effective constants and, when combined with model-size scaling, can affect fitted exponents.

V. Parameter constraints also change apparent exponents. All of the above assumes η and α can be freely retuned as T varies. In practice, hyperparameters are constrained (e.g. α fixed, η capped for stability, discrete grids). Such constraints remove the cancellation behind equation 43 and can lead to different effective power laws (e.g. a "critical batch size" beyond which increasing b is harmful because η cannot be increased accordingly).

C.3. Why can empirical fits show an optimal learning rate increasing with token budget?

Several empirical works report a one-dimensional fit $\eta^\star(T) \propto T^q$ while scaling the token budget via $T = bK$. In contrast, our proxy bound is multi-variate and depends on (b, K, α, η), so an "η vs. T" exponent is *not intrinsic*: it is *conditional on the scaling path* $T \mapsto (b(T), K(T), \alpha(T), \eta(T))$.

A protocol identity for effective exponents. Assume that over the (finite) range probed in practice, the tuned constant step size can be approximated by a separable power law

$$\eta^\star(b, K) \;\propto\; b^\kappa\, K^{-\lambda}, \qquad \kappa \ge 0,\ \lambda \ge 0. \tag{44}$$

Along any batch-growth path $b(T) \propto T^p$ (hence $K(T) = T/b(T) \propto T^{1-p}$), equation 44 implies an *effective* token exponent

$$\eta^\star(T) \;\propto\; T^{q_{\text{eff}}}, \qquad q_{\text{eff}} = \kappa p - \lambda(1-p). \tag{45}$$

In particular,

$$q_{\text{eff}} > 0 \quad\Longleftrightarrow\quad p > \frac{\lambda}{\kappa + \lambda}. \tag{46}$$

Thus, even if $\eta^\star$ decreases with the horizon at fixed batch ($\lambda > 0$), a positive fitted exponent in T can arise if the batch-growth effect dominates.

Instantiations for our LMO proxy. *(i) Fixed momentum proxy.* Optimizing the proxy over η at fixed (b, K, α) gives $\eta^\star(b, K) \propto K^{-1/2}$, i.e. $(\kappa, \lambda) = (0, \frac{1}{2})$. Hence $q_{\text{eff}} = -(1-p)/2 \leq 0$ for any p: in the fixed-momentum proxy, $\eta^\star(T)$ cannot increase with T.

(ii) Fixed-b K-optimal schedules evaluated along $T = bK$. Minimizing the LMO proxy at fixed b yields (up to constants / lower-order terms)

$$\eta^\star(b, K) \;\propto\; b^{1/4}K^{-3/4}, \qquad \alpha^\star(b, K) \;\propto\; \sqrt{b/K}.$$

Thus $(\kappa, \lambda) = (\frac{1}{4}, \frac{3}{4})$ and equation 45 gives

$$q_{\text{eff}} = p - \tfrac{3}{4}. \tag{47}$$

Therefore, a positive fitted exponent is possible under batch-heavy paths $p > 3/4$.

Caveat: saturation of $\alpha \leq 1$. The schedule above also implies $\alpha^\star(T) \propto b/\sqrt{T}$. Along $b(T) \propto T^p$, we have $\alpha^\star(T) \propto T^{p-1/2}$, so for $p > 1/2$ one eventually reaches a regime where α saturates at 1, and the effective η scaling transitions away from equation 47.

Connection to empirical scaling laws. This "path-conditioning" mechanism can explain *why* a study may report $\eta^\star$ increasing with T even though the globally token-optimal schedules in our analysis yield a decreasing $\eta^\star(T)$.

However, it does not guarantee matching exponents. For example, StepLaw reports (at fixed model size) $b^\star(T) \propto T^{0.571}$ and $\eta^\star(T) \propto T^{0.307}$ [34], i.e. $p \simeq 0.571 < 3/4$ but $q \simeq 0.307 > 0$ [34]. This cannot be explained by the LMO K-optimal proxy exponent $q_{\text{eff}} = p - \frac{3}{4}$ unless the effective exponents (κ, λ) in equation 44 differ substantially from $(\frac{1}{4}, \frac{3}{4})$, or the proxy assumptions fail.

As another reference point, von Rütte et al. [53] report $b^\star(T) \propto T^{0.8225}$ and $\eta^\star \propto (b^\star)^{0.3412}$ for diffusion LMs, implying $\eta^\star(T) \propto T^{0.28}$ over their explored range [53]. The batch exponent $p > 3/4$ falls in the batch-heavy regime where positive q_{eff} is possible, but the magnitude still differs from the LMO proxy, suggesting additional effects. So $p - 3/4$ is best viewed as one concrete mechanism (under a specific proxy + tuning protocol), not a general prediction for empirical scaling.

Appendix D. Detailed connections with empirical scaling laws

Among our analyses, fixed momentum is the most practically relevant, as momentum is usually set to a standard value and rarely tuned in modern training pipelines [8, 24, 47], though such practice is slowly changing [39, 42, 56]. On the other hand, batch size is often chosen for hardware efficiency and throughput rather than optimization performance. In these settings, our fixed-batch-size scaling laws can provide the appropriate theoretical description of achievable performance (see App. A).

Scaling learning rate with batch size and token transfer at a fixed batch size. Bjorck et al. [9] provide perhaps the closest empirical counterpart to our fixed-batch horizon scaling: at a fixed batch size of 0.5M tokens, and across model sizes from 50M to 2.7B parameters, they find that the optimal peak learning rate decreases $\eta^\star \propto T^{-\delta}$ (with δ varying from 0.32 to 0.71 across model sizes) as the token horizon increases. Their observation is directionally consistent with Theorem 1, which predicts $\eta^\star(T) \propto T^{-1/2}$ for fixed momentum and fixed batch size.

Malladi et al. [38] and Compagnoni et al. [14] provide theoretical evidence for the well-known square root scaling law: for adaptive methods, at a fixed token budget, scaling $b \mapsto \kappa b$ requires $\eta \mapsto \kappa^{1/2}\eta$. Recently, Mlodozeniec et al. [42] verified this law at scale, and also noted that (at a fixed batch size) the compute-budget (token-horizon) transfer requires scaling the learning rate down as $\eta \mapsto \eta\kappa^{-1/2}$ when the number of training tokens is increased by a factor κ. The same law is discussed by [11] in the context of convex problems trained with SGD. We note that *both these trends are predicted by our Theorem* 1: at a given token budget T, the optimal learning rate scales as $b^{1/2}T^{-1/2}$.

Further, note that our work complements the SDE viewpoint [14, 38] in a crucial way. SDE analyses were first motivated by intuition around sharp minima, generalization, and critical batch size [28, 31, 46, 50]. They derive scalings by matching statistical properties of the iterate distributions, and they also provide practical tools that are not captured by our bound-based proxy, such as SVAG [35] diagnostics for testing whether the SDE approximation is valid in a given training setting and predictions for when batch-size scaling should break down. Here we ask a complementary question: which batch size, learning rate, and momentum minimize a finite-horizon convergence proxy? Our comparisons, though recovering their scalings as a special case, concern the expected gradient-norm "performance" (connected to the loss) and capture a finite training horizon.

Although our contribution is theoretical, Appendix F.1 reports a small constant-learning-rate language-model sanity check. It is directionally consistent with the proxy: at fixed batch size the best learning rate decreases with the token horizon, while at a fixed horizon the best learning rate increases with batch size.

Optimal batch size scaling with token budget. Empirically, with fixed momentum, the optimal batch size scaling is found to be anywhere between $T^{0.3271}$ and $T^{0.8225}$ (Appendix Table 3, first column), which is reasonably[3] consistent with our result $T^{1/2}$. Several works instead discuss the notion of critical batch size: *the threshold at which further increasing batch size no longer gives near-linear speedup in optimization steps*. Zhang et al. [56] show empirically that the critical batch size scales primarily with data size rather than model size. In addition, they show theoretically (for SGD on linear regression) that in the bias-dominated, early-training regime, the critical batch size is $b = 1$; as training progresses and variance becomes dominant, the critical/efficient batch size

3. Note that differences in exact exponents may arise from the indirect relationship between gradient norm-based metrics and loss. See Appendix C for a simple sensitivity analysis showing how deviations from the i.i.d. bounded-variance mini-batch model and other constraints can shift the predicted exponents.

increases. A similar behavior can also be seen in Appendix Figure 3 for LMOs, where we see that the optimal, rather than critical, batch size ramps up at around 10^9 tokens[4], and is initially one (see third panel).

SGD: optimal batch size and linear learning rate scaling. The framework of Kovalev [33] covers normalized steepest descent (LMO) methods, not vanilla SGD. However, it is possible to extend our methodology to SGD and derive scaling results from well-known bounds [20]. We presented this in Appendix E: a simple argument yields that at a fixed batch size the optimal learning rate scales as $\eta^\star(b, T) = bT^{-1/2}$, in perfect agreement with the linear scaling suggestion by Jastrzebski et al. [28], Smith et al. [50]. Next, our results suggest that, in contrast to LMO methods such as Muon or SignSGD with momentum, *SGD does not have a non-trivial token-optimal batch size* (given an optimal step size). This is in agreement with recent empirical results, suggesting SGD (in contrast to Adam) always profits from an increased iteration budget at low token count [39, 52].

Relation between batch size and step count. Our scaling rules also predict that, under fixed momentum, achieving a certain target performance requires both a minimum batch size (Appendix Figure 10) and a minimum step count (Appendix Figure 12). Indeed, von Rütte et al. [53] report a hyperbolic relation between step count and batch size that closely resembles our theoretical results, suggesting this may be a fundamental property of LMO optimizers. We build on this and *derive this hyperbolic relationship*, clear from Appendix Figure 11. Notably, a similar relation exists for standard SGD, as empirically reported by McCandlish et al. [40], where a minimum step count and token budget are required to achieve a certain target performance, but no minimum batch size requirement exists.

Momentum scaling with batch size. A practical implementation of the idea of momentum adaptation as the batch size is scaled appears only in very recent work. Marek et al. [39] propose to scale the β_2 momentum parameter in Adam as follows: $b \mapsto \kappa b$ requires $\beta_2 \mapsto \beta_2^\kappa$. At the same time, Ferbach et al. [17], Orvieto and Gower [43] suggest that the choice $\beta_1 = \beta_2$ in Adam leads (after tuning) to near-optimal performance, drawing a direct link to SignSGD with momentum parameter β. Our scaling result in Theorem 2 shows that, at a fixed token budget T and a fixed batch size b, the optimal momentum $\beta = 1 - \alpha \approx 1 - bT^{-1/2}$. Hence, if b is scaled by a factor κ, we get $\beta \approx 1 - \kappa bT^{-1/2}$. Our result is deeply connected to the strategy by Marek et al. [39], indeed note that as $T \to \infty$ we have $\beta^\kappa \approx \left(1 - bT^{-1/2}\right)^\kappa = 1 - \kappa bT^{-1/2} + \mathcal{O}\left(T^{-1}\right)$. A decreasing optimal momentum as the batch size increases is also found to be effective by Zhang et al. [56].

Relation to norm-transfer scaling. The most direct empirical counterpart to our joint LR/batch discussion is the norm-transfer study of Filatov et al. [18]. Using the norm-based Scion optimizer, they find $b^\star(T) \propto T^{0.45\pm0.07}$ and $\eta^\star(T) \propto T^{-0.28\pm0.07}$, close to our fixed-momentum prediction $b^\star(T) \propto T^{1/2}$, $\eta^\star(T) \propto T^{-1/4}$. They also identify a low-sensitivity region where several (η, b) choices behave similarly, which is qualitatively aligned with our conclusion that, after retuning momentum and learning rate, multiple batch-growth paths can be asymptotically near-optimal and the $T^{1/6}$ law is selected only by lower-order terms.

Learning rate scaling under jointly increasing batch and token budget. To start, we recall (see first paragraph in this section) that our results align with observations that, *at a fixed batch size*, the optimal learning rate should decrease as the training horizon grows (Theorem 1; Mlodozeniec et al. [42]). Yet, how does the optimal learning rate change as the *batch size is chosen to scale* with

4. The precise value for this phase transition crucially depends on momentum, see App. F, and on the values of C_1, C_2, C_3 in equation 3.

Table 3: Empirical batch-size and learning-rate scaling across representative studies. Optimizers, model-size protocols, learning-rate schedules, and other fixed hyperparameters differ across works; the table summarizes reported exponents rather than a strictly like-for-like comparison. N denotes the number of non-embedding parameters. δ_N [9] varies from 0.32 to 0.71 across models.

	Batch Size	Learning Rate	Optimizer
DeepSeek [7]	$b^\star \propto T^{0.3271}$	$\eta^\star \propto T^{-0.1250}$	AdamW [32, 37]
Bjorck et al. [9]	fixed	$\eta^\star \propto T^{-\delta_N}$	AdamW [32, 37]
StepLaw [34]	$b^\star \propto T^{0.571}$	$\eta^\star \propto N^{-0.713}\, T^{0.307}$	AdamW [32, 37]
von Rütte et al. [53]	$b^\star \propto T^{0.8225}$	$\eta^\star \propto T^{0.2806}$	LaProp [59]
Filatov et al. [18]	$b^\star \propto T^{0.45\pm0.07}$	$\eta^\star \propto T^{-0.28\pm0.07}$	Scion [44]

the token budget? The empirical picture is mixed. Bjorck et al. [9] study the closest fixed-batch token-horizon setting and find that longer horizons require smaller learning rates, in agreement with Theorem 1. In contrast, StepLaw [34] and von Rütte et al. [53] report positive fitted learning-rate exponents[5] when batch size is also scaled with the token budget.

Meanwhile, Theorem 1 predicts that if $b \propto T^{1/2}$, then $\eta^\star = T^{-1/4}$. Appendix C.3 shows that a positive fitted exponent can arise without contradicting Theorem 1 when one measures $\eta^\star$ along a batch-scaling path $b(T)$ (hence $K(T) = T/b(T)$), e.g. when the budget is increased mainly via larger batches (fewer additional steps) or when hyperparameters are transferred across budgets. Further, note that the bound in equation 2 rules out $\eta^\star(T) \to \infty$ for the constant-step proxy because of the $O(\eta)$ term, independent of batch size and momentum.

Appendix C additionally discusses how deviations from the idealized assumptions behind the proxy (e.g., non-$b^{-1/2}$ noise scaling, heavy tails, or norm-mismatched variance for LMO methods) can change the predicted exponents. Overall, this points to a clear direction for future investigation: determine which part of the remaining mismatch is due to (i) protocol constraints vs. (ii) assumption mismatch vs. (iii) looseness of the bound in equation 2 for modern large model training regimes.

5. Note that the comparison with empirical learning-rate scaling is subtle and influenced by factors such as model size. DeepSeek [7] reports a decrease in optimal learning rate as the token budget increases. However, since DeepSeek also scales model size with training horizon, this effect may largely reflect the lower learning rates required by larger models. In contrast, Li et al. [34] explicitly models the role of model size, and von Rütte et al. [53] uses μP initialization to decouple size from learning rate. Both decoupled analyses find a positive correlation between the token horizon and the learning rate.

Appendix E. Comparison with SGD

In contrast to LMO-based optimizers, vanilla SGD does not exhibit a non-trivial token-optimal batch size. Under a fixed token budget $T = bK$, the classical bounds balance the optimization and variance terms so that, after tuning, the resulting performance depends only on T and becomes independent of b. Consequently, batch size mainly trades off variance reduction against the number of steps, unlike LMO methods, where additional momentum- and normalization-induced terms break this cancellation and induce a genuine optimal batch-size scaling.

To see this, with the same notation as equation 2 consider the classical non-convex guarantee [20]:

$$\min_{1\le k\le K} \mathbb{E}\|\nabla f(x^k)\|_2^2 \;\lesssim\; \frac{\Delta_0}{\eta K} \;+\; \frac{L\eta\sigma^2}{b}. \tag{48}$$

Here we consider SGD without momentum. Adding or tuning momentum does not improve the asymptotic rate for SGD: see [21] (Theorem 3) and [55] (Theorem 3). The novelty in modern analysis of adaptive methods after Cutkosky and Mehta [15] was to show that indeed momentum has instead a crucial role in adaptive methods. Optimizing equation 48 over η yields

$$\eta^\star \propto \sqrt{b/K}, \qquad \min_{1\le k\le K} \mathbb{E}\|\nabla f(x^k)\|_2^2 \;\lesssim\; \sqrt{\frac{\Delta_0 L\sigma^2}{bK}}.$$

Thus, at a fixed iteration budget K, increasing b improves the bound (as in many stochastic methods). Instead, substituting $K = T/b$ into equation 48 gives, optimizing over η

$$\min_{1\le k\le K} \mathbb{E}\|\nabla f(x^k)\|_2^2 \;\lesssim\; \min_{\eta}\left\{\frac{\Delta_0\, b}{\eta T} + \frac{L\eta\sigma^2}{b}\right\} \;\lesssim\; \sqrt{\frac{\Delta_0 L\sigma^2}{T}}, \qquad \eta^\star \propto bT^{-1/2}$$

which is *independent of* b at the level of this proxy bound. In other words, under a fixed token budget, the *classical SGD analysis does not predict a non-trivial token-optimal batch size*: after optimizing η, b largely trades off iterations vs. variance reduction in a way that cancels.

Summary. The LMO bound used in the paper (equation 2) differs structurally: (i) it controls $\|\nabla f(\cdot)\|_\star$ (not squared), and (ii) it contains additional terms coupling (η, α) and mini-batching, including a noise-floor term $\propto \sigma\sqrt{\alpha/b}$ and a burn-in / initialization term that can scale as $\propto \sigma/(\alpha\sqrt{b}K)$ under matched stochastic initialization (Appendix C.1). As a consequence,

- with *fixed* α, the proxy retains a decreasing-in-b contribution under $T = bK$ (e.g. $\propto 1/\sqrt{b}$), leading to a genuinely non-trivial token-optimal batch size (Theorem 1: $b_T^\star \propto \sqrt{T}$);
- with *tuned* α, the leading-order cancellation in b under $T = bK$ becomes more similar in spirit to SGD, and $b^\star(T)$ is pinned only by lower-order terms (Theorem 3; see also App. C.2).

Appendix F. Additional Plots

F.1. Empirical sanity check

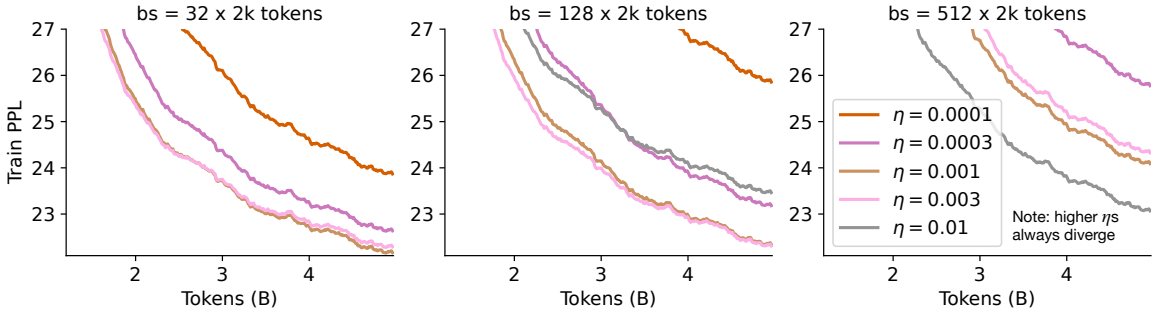

Figure 2: Sanity-check experiment for our analysis. We perform a constant-η training experiment on a 160M transformer (PlainLM implementation [1]), trained with a language modeling objective for up to $5B$ tokens from SlimPajama [51]. The setup is the same as in Theorem 1 and Appendix Figure 3. With 8% warmup + constant learning rate (Adam betas fixed to $(0.9, 0.95)$), grid-searching η across batch sizes shows clear structure: (i) at fixed batch size, the optimal η decreases over training time (e.g., $b = 32$ shifts from 0.003 to 0.001 after $2.5B$ tokens); (ii) this trend appears similarly for $b = 128$, albeit the switching point is not yet reached at $5B$ tokens; and (iii) the optimal η increases with batch size (at $T = 5B$ tokens, $\eta = 0.001$ at $b = 32$, $\eta = 0.003$ at $b = 128$, $\eta = 0.01$ at $b = 512$). Constant η allows finite-time comparisons that reveal both token-dependent and batch-dependent optimal ηs.

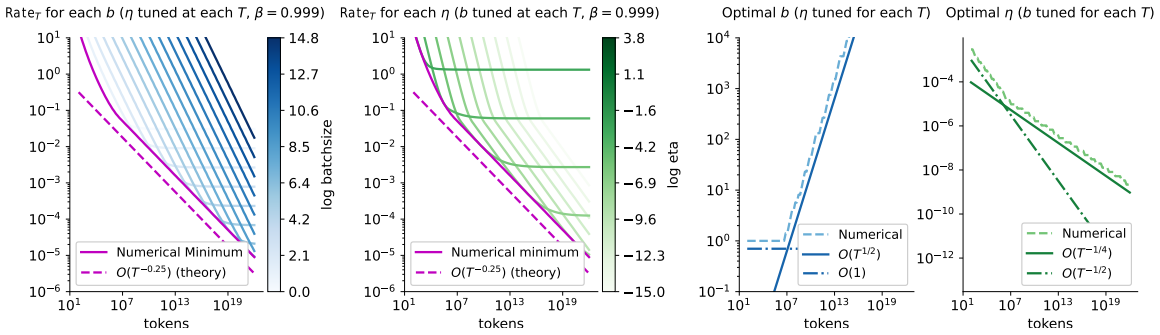

Figure 3: Verification of Theorem 1. Shown are the trends of equation 3 ($C_1 = C_2 = C_3 = 1$) under the choice $\beta = 1 - \alpha = 0.999$. Additional αs can be found in Appendix F. In the two plots on the left, we show in magenta performance at the best value of (η, b) for each token budget, following $\mathcal{O}(T^{-1/4})$. Plotted in blue are also performances for a fixed batch size, minimizing over η at each token budget, and in green performances for a fixed learning rate, minimizing b at each token budget. In the two plots on the right, we show how the optimal batch size and learning rate scale with tokens. The trends predicted by Theorem 1 hold after a burn-in phase, where the optimal batch size is $b = 1$.

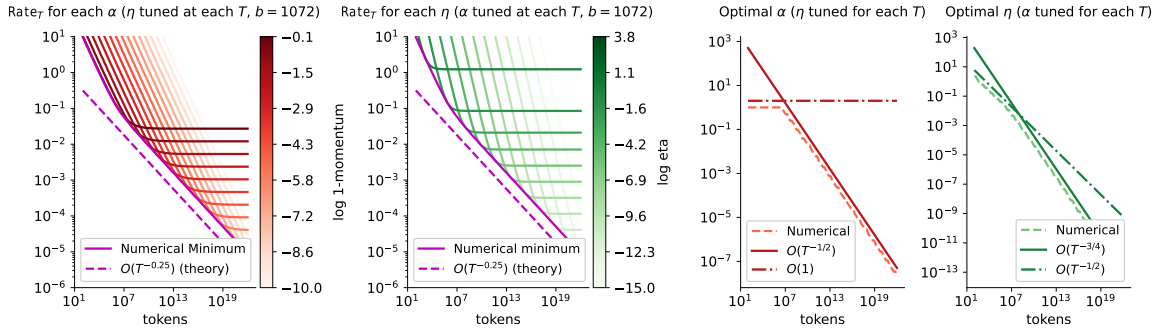

Figure 4: Numerical verification of Theorem 2 for $b = 1072$. The setting is the same as in Appendix Figure 3.

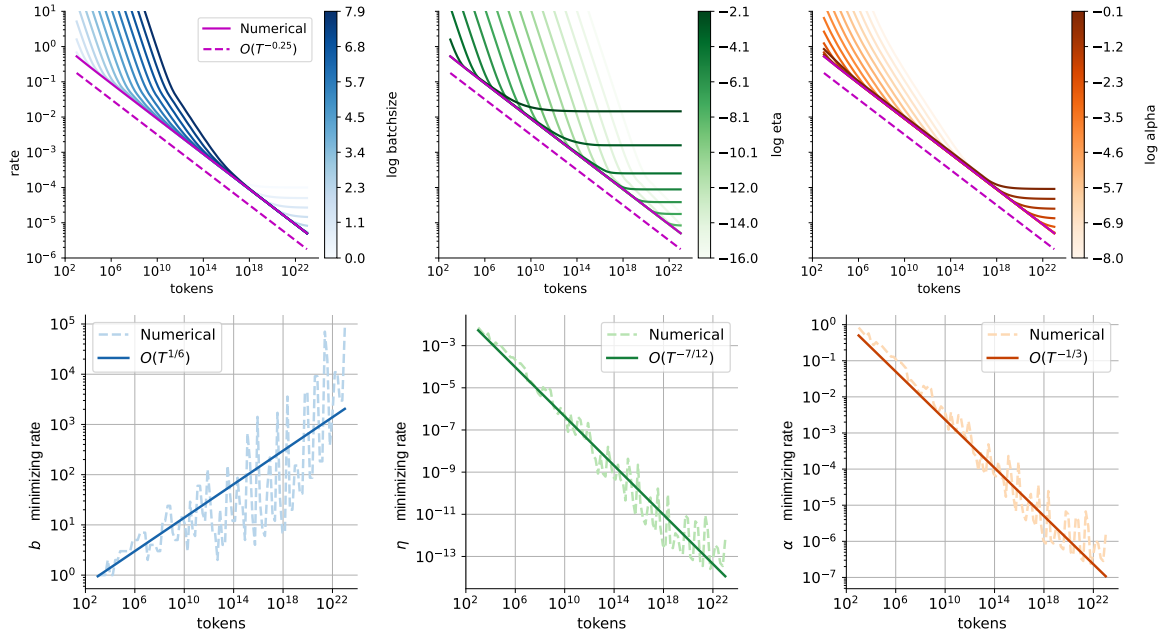

Figure 5: Numerical verification of Theorem 3. Oscillations are due to the numerical issues caused by extremely similar performance among hyperparameter choices – see Corollary 4. See Appendix C.2 for further comments.

Scaling plots. We provide here numerical scaling plots.

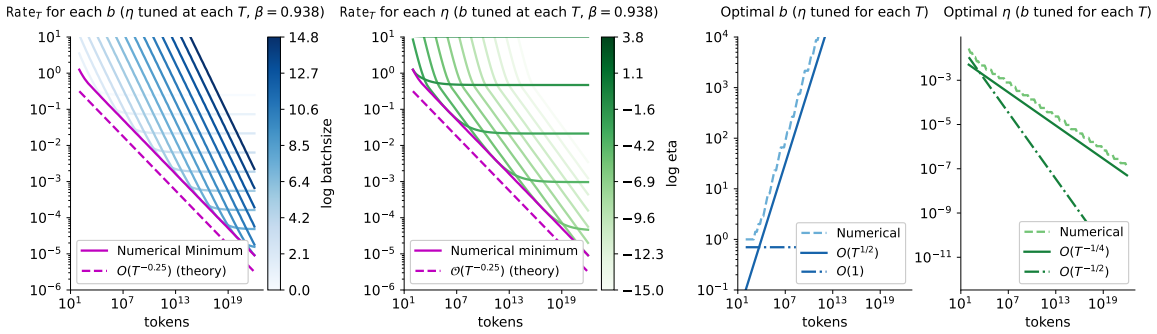

Figure 6: Numerical verification of Theorem 1 for $\beta = 0.934$.

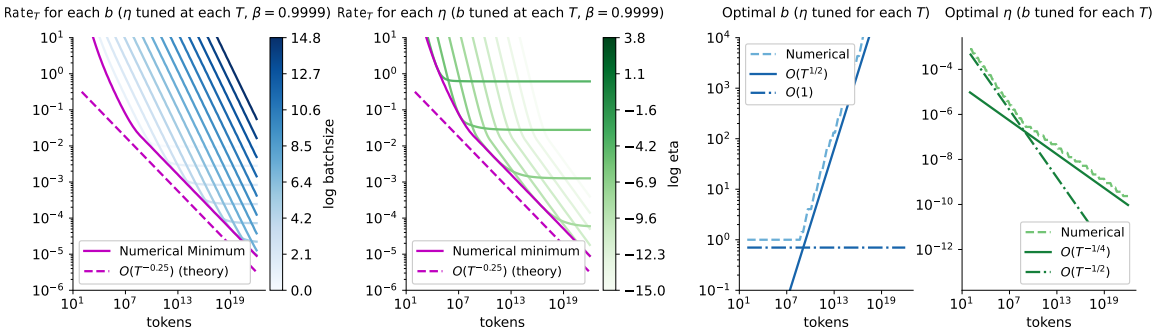

Figure 7: Numerical verification of Theorem 1 for $\beta = 0.9999$.

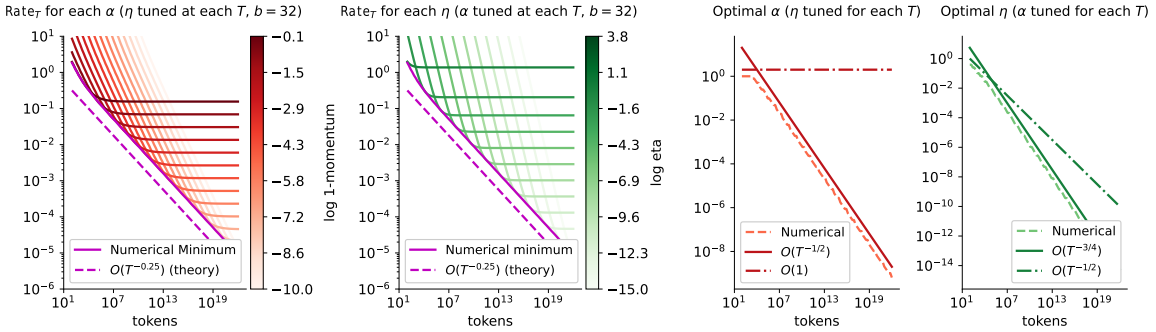

Figure 8: Numerical verification of Theorem 2 for $b = 32$.

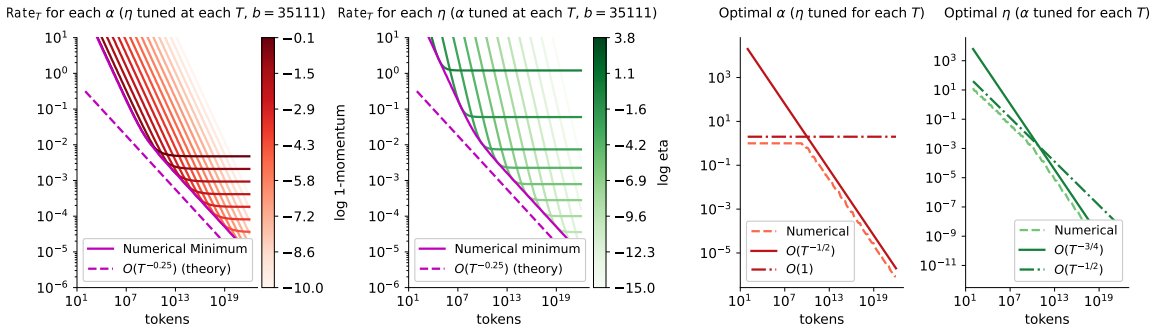

Figure 9: Numerical verification of Theorem 2 for $b = 35111$.

Iso-loss curves. As shown in Figure 10, under fixed momentum, achieving a certain target performance requires a minimum batch size. We identify a trend of $b_T^\star \propto T^{1/2}$ as predicted by our theory. The saturation of the near-optimal line in the left panel indicates that, at large token budgets, performance becomes less sensitive to batch size scaling when the learning rate η is lower bounded, since smaller momentum requires proportionally smaller η.

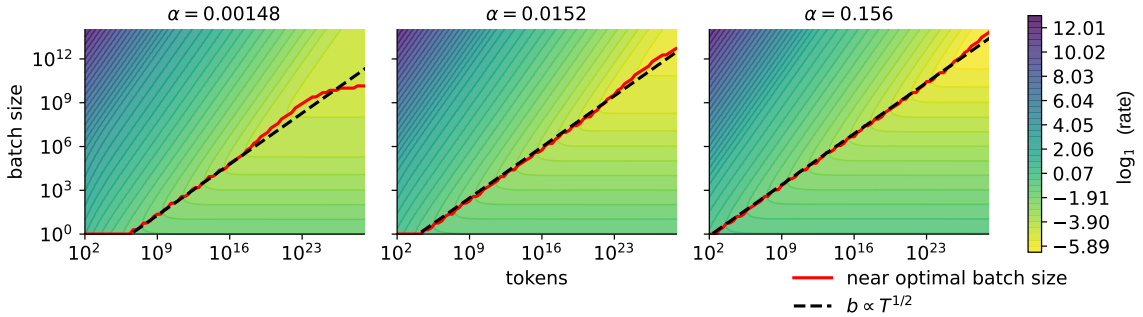

Figure 10: Contours of best achievable performance versus batch size and number of tokens. Fixed α at different values, tuned $\eta \in [1.7e-6, 1]$. The red curve denotes the near-optimal (99% performance) batch size as a function of the number of tokens.

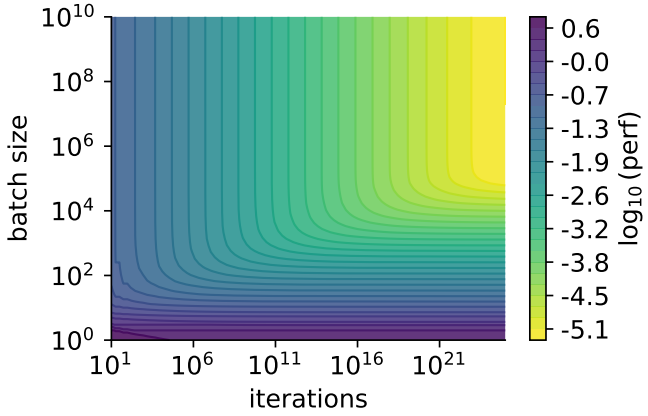

Figure 11: Contours of best achievable performance versus batch size and training iterations for a fixed α and tuned η.

Three regimes and a hyperbolic $(K, \sqrt{b})$ tradeoff (iso-performance curves). Fix a target level ε and let $c := \varepsilon - \text{const} > 0$ absorb terms independent of (b, K). Assuming the learning rate is well tuned for each (b, K) (i.e., η minimizes the $\Delta_0/(\eta K) + L\eta(\frac{7}{2} + \frac{2}{\alpha})$ part of the bound), the remaining dependence on (b, K) can be summarized as

$$u_\eta(b, K) \approx \frac{C_{\text{det}}}{\sqrt{K}} + \frac{C_{\text{burn}}}{K\sqrt{b}} + \frac{C_{\text{floor}}}{\sqrt{b}},$$

where $C_{\text{det}} := 2\sqrt{\Delta_0 L(\frac{7}{2} + \frac{2}{\alpha})}$, $C_{\text{burn}} := \frac{2\rho\sigma}{\alpha}$, $C_{\text{floor}} := 2\rho\sigma\sqrt{\alpha}$.

Setting $u_\eta(b, K) = c$ reveals three regimes:

1. Iteration-limited ($b \to \infty$) with $K_{\min} = (C_{\text{det}}/c)^2$;
2. Batch-limited ($K \to \infty$) with $b_{\min} = (C_{\text{floor}}/c)^2$;
3. An intermediate tradeoff region where the burn-in term dominates, yielding $K\sqrt{b} \approx C_{\text{burn}}/c$, i.e. $\sqrt{b} \propto 1/K$.

We can also rearrange the equation into:

$$\left(c\sqrt{K} - C_{\text{det}}\right)\left(c\sqrt{b} - \left(C_{\text{floor}} + \frac{C_{\text{burn}}}{K}\right)\right) = C_{\text{det}}\left(C_{\text{floor}} + \frac{C_{\text{burn}}}{K}\right) \approx C_{\text{det}}\left(C_{\text{floor}} + \frac{C_{\text{burn}}}{K_0}\right)$$

where K_0 denotes a representative iteration scale used to approximate the slowly varying $1/K$ term. This mirrors the shifted-hyperbola iso-loss law reported by von Rütte et al. [53] (see their equation (7)), up to our use of $\sqrt{b}$ induced by the $1/\sqrt{b}$ noise scaling.

When momentum is fixed, and learning rate is tuned, the hyperbolic relationship between batch size and iterations is shown in the middle panel of Figure 12. Note that on the left panel, if α and η can be unrestrictedly tuned, the final performance achieved is higher than that of the middle and right panels. This is because a larger batch size requires a much smaller learning rate, which is often not realistic. If the η grid is lower bounded, then there is no significant difference in the achievable performance between fixed α (middle) and jointly tuned α (right).

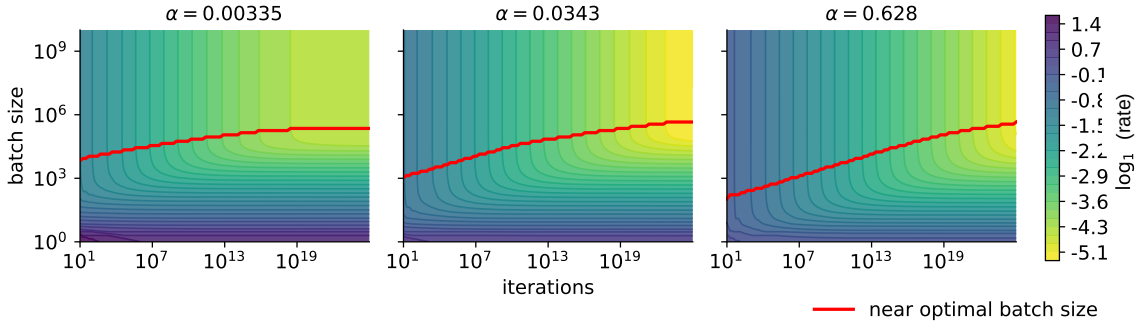

Figure 12: Contours of best achievable performance versus batch size and training iterations. Fixed α at various values, and tuned $\eta \in [1.07e-7, 1]$. The red curve denotes the near-optimal (99% performance) batch size as a function of the number of iterations.