Critical batch size under Muon optimizer

Tommy Chu

***Abstract*—McCandlish et al. predicted that one quantity, the gradient noise scale B_{noise}, sets the critical batch size B^* at which data parallelism stops paying off. We test whether that prediction survives Muon, whose non-linear Newton–Schulz step falls outside the original derivation. Across 124 runs of GPT-2, B^* exists under Muon, rising about $5\times$ over training and shifting down at high momentum. The noise scale matches B^* in magnitude but not in structure. It does not track how B^* evolves over training, and it ranks the momenta in the opposite order. The cause is that B_{noise} is path-dependent. Even at a fixed validation loss it varies with the training batch, so it cannot be used to set the batch size.**

I. Introduction

Doubling the batch size B eventually stops halving the number of steps to convergence. McCandlish et al. [1] called this threshold the critical batch size B^* and proposed the gradient noise scale $B_{\text{noise}} = (\operatorname{tr}\Sigma/|G|^2) \cdot \text{seq_len}$, the Hessian-free estimator, as a cheap proxy for it. They predicted that $B^* \approx B_{\text{noise}}$ across a broad family of architectures and tasks.

It remains open whether this prediction extends to modern, non-linear optimizers. Direct measurements of B^* on Adam-trained LLMs find that B_{noise} underestimates it by orders of magnitude [2]. Muon [3], a new optimizer for LLMs that has shown promising scaling properties [4], applies a non-linear Newton–Schulz orthogonalisation to its momentum buffer, a step that falls outside the assumptions of McCandlish's derivation.

We therefore ask three questions.

1) Does Muon have a critical batch size?
2) Does momentum move it?
3) Does the cheap noise-scale shortcut still predict it?

In short: (1) yes; (2) yes, downward; (3) no. We answer all three by measuring B^* from 124 training runs.

II. Experimental setup

A. Model and training

We train a 162M-parameter GPT-2 (12 layers, hidden dimension 768, sequence length 1024) on Fineweb10B, on $8\times$ NVIDIA A100 80GB per run, with the modded-nanogpt [5] training harness. Muon manages the $\sim$85M weight-matrix parameters inside the transformer blocks and AdamW manages the rest at a fixed learning rate. Only B, momentum μ and Muon's learning rate η vary across runs. The schedule is constant-then-linear-decay: hold η constant for the first 30% of steps, then decay linearly to zero. Total cost was about 500 GPU hours.

B. Sweep grid

We conduct 124 training runs (single seed):

- $B \in \{2^{15}, 2^{16}, \ldots, 2^{23}\}$ tokens, where $2^{19} = 524$K is the Track-3 baseline [5].
- $\mu \in \{0.85,\ 0.95,\ 0.99\}$.
- $\eta \in \{0.5,\ 0.75,\ 1.0,\ 1.5\} \cdot 0.035\sqrt{B/2^{19}}$ per (B, μ) cell.

We added 16 additional top-ups to expand the grid for cells where the best-performing η sits at the edge.

Token budget is 1B-tokens for $B \le 1$M and 2B-tokens for $B \ge 2$M (so step-starved large batches can reach harder loss targets). For a fair comparison across B we read numbers only at the common 1B-token mark.

C. Muon update

We use the best-performing public Muon variant at the time of this writing. On top of the original [3, Algorithm 2], it adds Nesterov lookahead, a shape-aware scaling, and decoupled weight decay (0.025). For each Muon-managed parameter at step t, with mini-batch gradient g_t:

$$m_t = (1-\mu)\, g_t + \mu\, m_{t-1} \tag{1}$$

$$\tilde{m}_t = (1-\mu)\, g_t + \mu\, m_t \quad \text{(Nesterov)} \tag{2}$$

$$u_t = \mathrm{NS}_{12}(\tilde{m}_t) \tag{3}$$

$$\theta_{t+1} = \theta_t - \eta \sqrt{\max(1,\ d_{\text{out}}/d_{\text{in}})}\, u_t \tag{4}$$

where NS_{12} is 12 iterations of the quintic $p(X) = 2X - 1.5\,X^3 + 0.5\,X^5$ that approximates the polar factor of $\tilde{m}_t$ (the closest semi-orthogonal matrix in Frobenius norm). This orthogonalisation is the non-linearity that makes Muon interesting for our purposes. Coefficients and iteration count are fixed.

D. Gradient noise probe

After each validation pass we draw two disjoint micro-batches from a held-out shard, accumulate their gradients across ranks, and compute

$$\text{dot} = G_1 \cdot G_2 \tag{5}$$

$$\text{diff_sq} = \|G_1 - G_2\|_F^2 \tag{6}$$

$$\operatorname{tr}\Sigma = (b/2) \cdot \text{diff_sq} \tag{7}$$

$$B_{\text{noise}} = (\operatorname{tr}\Sigma/\text{dot}) \cdot \text{seq_len} \quad (\text{if dot} > 0) \tag{8}$$

where $b = 128$ is the micro-batch size in sequences and $\text{seq_len} = 1024$ is the sequence length. The probe is taken on the *raw* Muon gradient, before Newton–Schulz, on Muon parameters only. Each run logs about 40 validation passes and 100 noise probes evenly spaced across steps.

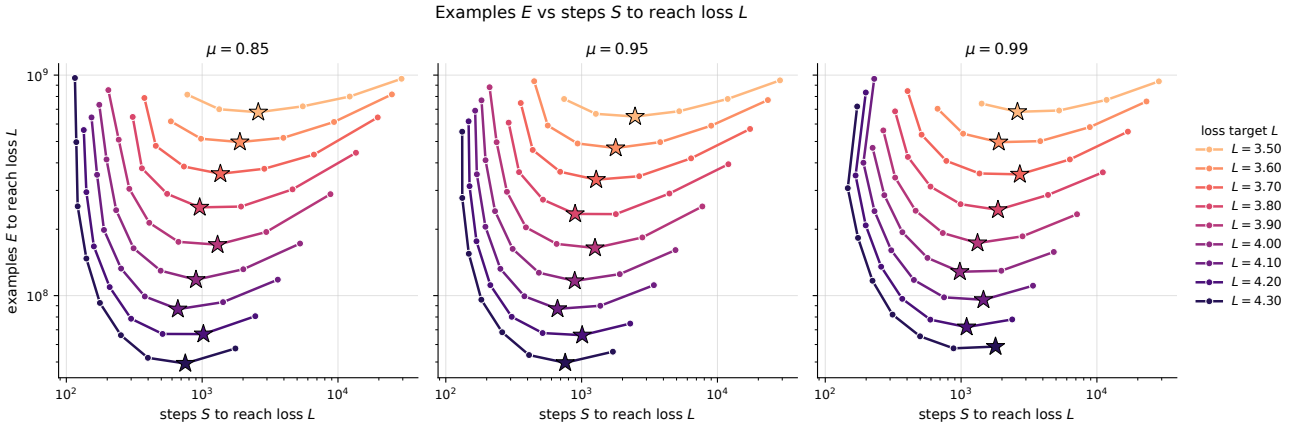


Fig. 1. Pareto fronts for each momentum. For each loss target L (dark purple early-training to light yellow late-training) we plot the LR-tuned best run at every batch size, with steps $S(L)$ on the x-axis and training samples $E(L)$ on the y-axis. The star marks the vertex elbow (our operational B^*).

E. Derived quantities

a) LR-tuned $S(L)$: For each (B, μ) cell, we take the LR run that reaches loss L in the fewest steps and linearly interpolate S between the two validation probes bracketing L.

b) Samples to L: defined as $E(L) = S(L) \cdot B$.

c) Critical batch size $B^*(L, \mu)$: We define this as the vertex of a parabola fit in $\log_{10} E$ vs $\log_2 B$, restricted to $B \leq 2\text{M}$. We use a parabola because the standard McCandlish hyperbola is degenerate on our data (appendix B).

d) Noise scale near L: This is the per-run geometric mean of B_{noise} over a tight loss window around L. We use the geometric mean because B_{noise} is log-distributed.

III. Muon has a critical batch size

A. Existence

Figure 1 shows the Pareto front for each momentum. Every curve has an interior minimum in samples processed, $E(L)$, which we take as the operational critical batch size B^*. The answer to our first question is therefore *yes*: under Muon, doubling B stops paying for itself.

The shape of that minimum, however, differs from theory. McCandlish predicts an L-shaped frontier, i.e. the right leg (small batch) should flatten, while the left leg (large-batch) should rise vertically. We instead see U-shapes.

a) Right leg (small B): The right leg curves up instead of flattening. At $\mu = 0.85$ the small- and mid-B cells are bracketed, with interior LR optima (fig. 5), so these are their true best runs. We attribute it to the Newton–Schulz step. Orthogonalisation pushes the update toward unit spectral scale, whereas McCandlish assumes a near-linear update in which small-batch gradient noise integrates away. At small B the momentum buffer is noise-dominated, so orthogonalising it emits a full-magnitude step in a partly-random direction.

b) Left leg (large B): The left leg is mildly inflated too. Large-B optima pin to the smallest, but still suboptimal, LR (fig. 5). Furthermore for $B \geq 2\text{M}$ the common 1B-token budget ends mid-decay and misses the end-of-training drop.

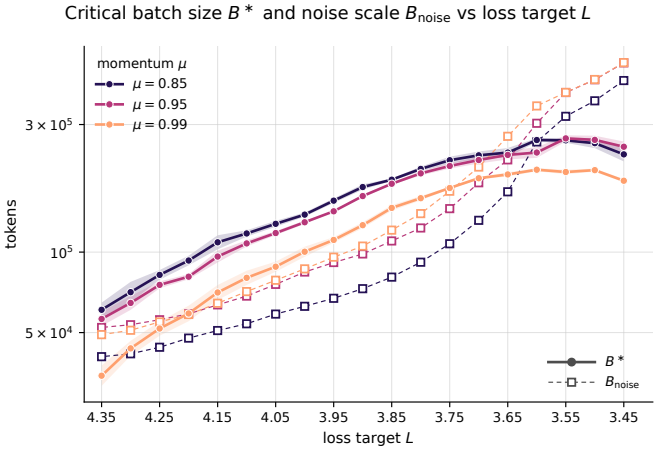


Fig. 2. $B^*(L)$ and $B_{\text{noise}}(L)$ on shared axes, coloured by momentum μ. B_{noise} is the geometric mean over the three near-elbow batch sizes within a $\pm 6\%$ loss window. Loss decreases left to right. The two almost agree in magnitude but disagree in slope, and their momentum ordering is inverted.

Fortunately the elbow falls between the two artefacts, where neither artefact dominates.

B. Momentum lowers B^*

Figure 2 plots $B^*(L)$ for each momentum. B^* rises by about $5\times$ as the loss falls, then flattens near the lowest L. This is similar to what McCandlish and Kaplan [6] report for Adam-trained language models.

Momenta $\mu \in \{0.85, 0.95\}$ nearly coincide, and $\mu = 0.99$ separates below both. We interpret this as temporal averaging (momentum smooths the gradient over past mini-batches), so higher μ has more effective samples per update and large-batch averaging saturates sooner.

IV. The noise scale does not predict B^*

B_{noise} agrees with B^* in magnitude, but not in structure, and it depends on the batch size it is meant to recommend.

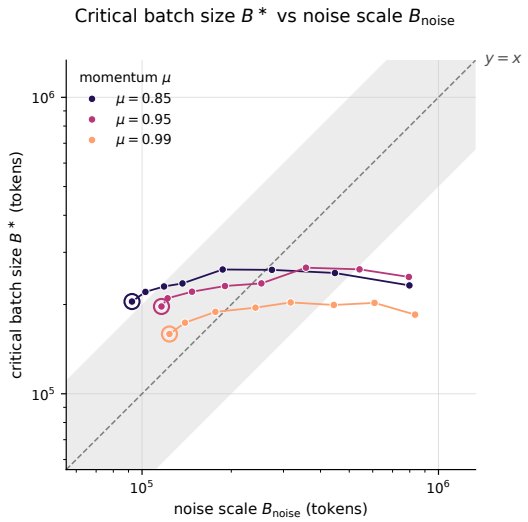


Fig. 3. B^* versus B_{noise} across loss targets, one curve per momentum (each dot an L, open ring the earliest-training end). The diagonal is McCandlish's prediction $B^* = B_{\text{noise}}$. The shaded band is $\pm 2\times$ around the diagonal.

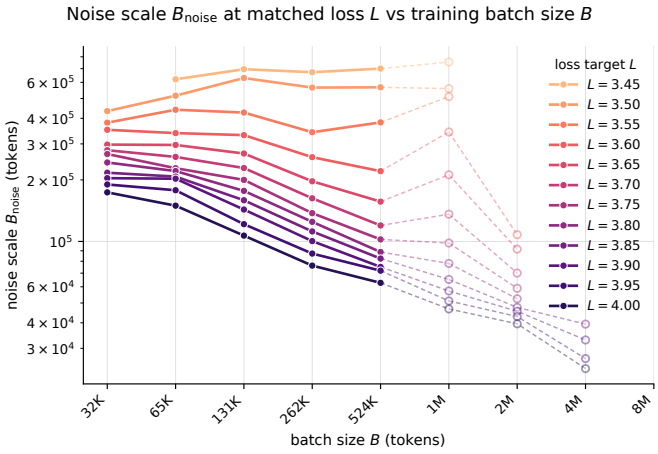


Fig. 4. B_{noise} at a matched loss target L versus training batch size B. Each point averages B_{noise} over the runs that reach L at that B. Solid markers are well-sampled cells, faded dashed markers are not.

A. Magnitude agreement, structural disagreement

Figure 3 traces B^* against B_{noise} across loss targets. On McCandlish's diagonal the two would rise together, but instead the traces are nearly flat: B_{noise} sweeps across the plot while B^* barely moves, meeting the diagonal at a single point.

The magnitudes stay within a small factor (closer than under Adam [2]) but only incidentally. A hyperbola fit would place B^* an order of magnitude higher (appendix B), and rescaling only slides the trace sideways, so no rescaling can tilt a flat trace onto the diagonal. The disagreement is structural (momentum even orders them oppositely).

B. Path-dependence of B_{noise}

B_{noise} is meant to describe the loss landscape: two runs reaching the same validation loss by different routes (different B) should report the same value. Figure 4 fixes a loss target L and plots B_{noise} against the training batch B. The curves are not flat, and not even monotone. At high L (early training) B_{noise} falls as B grows, while at low L (late training) it rises. Whatever B_{noise} measures, it is not a property of the loss alone: equal loss reached at different B leaves the optimizer in a different state. This compounds the mismatch of section IV-A: B_{noise} not only fails to track how B^* moves, but also depends on the batch size.

V. Caveats and Limitations

a) Single seed: Our claims are trend-level and rest on the density of the sweep rather than on single runs.

b) One model, one dataset: Every run trains the same 162M GPT-2 on Fineweb10B. Whether our results hold across scales and architectures is untested.

c) Simple noise scale: We probe the Hessian-free estimator. The Hessian-weighted noise would be more accurate (but far costlier).

d) LR-grid edges: At $B \geq 4\text{M}$ all three momenta bottom out at the lowest LR, so those cells are under-tuned (fig. 5).

e) Frozen AdamW: Our questions concern Muon's B^* and momentum and so we hold the AdamW LRs fixed.

VI. Conclusion

Muon has a critical batch size. It rises about $5\times$ over training and shifts down at high momentum. The gradient noise scale matches it in magnitude but tracks neither how B^* moves over training nor how momentum moves it (fig. 4). At a fixed loss, B_{noise} still depends on the training batch, so it measures the path a run took rather than the loss it reached. Using it to set the batch size is therefore circular. As a result, for optimizers such as Muon, Adam, and Shampoo, we expect direct branched-training measurement [2] to estimate batch-size scaling more reliably than the noise scale.

References

[1] S. McCandlish, J. Kaplan, D. Amodei, and O. D. Team, "An empirical model of large-batch training," 2018. [Online]. Available: https://arxiv.org/abs/1812.06162

[2] W. Merrill, S. Arora, D. Groeneveld, and H. Hajishirzi, "Critical batch size revisited: A simple empirical approach to large-batch language model training," 2025. [Online]. Available: https://arxiv.org/abs/2505.23971

[3] K. Jordan, Y. Jin, V. Boza, J. You, F. Cesista, L. Newhouse, and J. Bernstein, "Muon: An optimizer for hidden layers in neural networks," 2024. [Online]. Available: https://kellerjordan.github.io/posts/muon/

[4] J. Liu, J. Su, X. Yao, Z. Jiang, G. Lai, Y. Du, Y. Qin, W. Xu, E. Lu, J. Yan, Y. Chen, H. Zheng, Y. Liu, S. Liu, B. Yin, W. He, H. Zhu, Y. Wang, J. Wang, M. Dong, Z. Zhang, Y. Kang, H. Zhang, X. Xu, Y. Zhang, Y. Wu, X. Zhou, and Z. Yang, "Muon is scalable for llm training," 2025. [Online]. Available: https://arxiv.org/abs/2502.16982

[5] K. Jordan, J. Bernstein, B. Rappazzo, @fernbear.bsky.social, B. Vlado, Y. Jiacheng, F. Cesista, B. Koszarsky, and @Grad62304977, "modded-nanogpt: Speedrunning the nanogpt baseline," 2024. [Online]. Available: https://github.com/KellerJordan/modded-nanogpt

[6] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," 2020. [Online]. Available: https://arxiv.org/abs/2001.08361

APPENDIX A
LOSS CURVES AND LEARNING RATES

The B^* estimator reads the LR-tuned best run from each (B, μ) cell, so it is only as good as the tuning. Three diagnostics establish that the tuning is sound and that the 1B-token cap is a fair common reference.

Figure 5 sweeps the learning rate within each cell. Small and mid-B cells trace clean U-shapes with interior minima at every momentum, so their best runs are genuine optima. Large-B cells instead pin to the smallest rung on the ladder, and we flag these LR-saturated cells and hold them out of the B^* fit.

Figure 6 plots the tuned η^* against B. It follows the $\sqrt{B}$ rule through mid-B and overshoots the guide, while at $\mu = 0.99$ it dips below the guide around 524K to 1M. At $B \geq 4\text{M}$ all three momenta collapse onto the bottom rung of their ladders, which sits on the $\sqrt{B}$ line.

Figure 7 shows the loss trajectories against steps, tokens, and wall-clock seconds. No run misbehaves. All batch sizes reach comparable loss at 1B tokens, so the cap is a fair common cut. Under the cap the largest batches stay highest in wall-clock, so large B buys no speed advantage here.

APPENDIX B
B^* FITS AND DIAGNOSTICS

Our B^* values are parabola-vertex elbows fit on $\log_{10} E$ versus $\log_2 B$ for $B \leq 2\text{M}$. We use a parabola because the textbook McCandlish hyperbola is degenerate on our data. This section shows the raw substrate, the per-cell fits, and the comparison that drives that choice.

Figure 8 is the raw $S(B)$ behind the Pareto fronts, one curve per loss target. The curves drop steeply at small B and flatten at large B for $\mu \in \{0.85, 0.95\}$, while at $\mu = 0.99$ the largest-B points turn back up because the LR-saturated cells need more steps. The hardest targets cover only a mid-range of B, since the smallest batches never reach them and the largest run out of budget first.

Figure 9 overlays the parabola and hyperbola fits on the (B, E) data for four loss targets and three momenta. At lower L the U is fully bracketed and the vertex sits in the interior with a tight band. At higher L the vertex B^* falls to the small-batch edge, the left arm is truncated, and the band widens. The hyperbola stays monotone and never bends up the left arm, so it tracks the easy high-L targets but misses the bracketed U at low L.

Figure 10 is the detailed comparison at $L = 3.60$. The right-leg hyperbola fit ($B \geq 0.6\, B^*_{\text{elbow}}$) is clean inside its window, and the residuals stay near zero across it. They climb to 0.5 to 0.8 at the smallest B, which confirms that the rising left leg is real and not a fit artefact. The hyperbola fits the leg it is handed yet places B^* roughly an order of magnitude above the elbow, so it does not give a usable critical batch size.

Figure 11 isolates the momentum effect at a single loss target ($L = 3.60$), overlaying all three momenta. The elbow of the samples E moves left as μ grows, so higher momentum lowers the critical batch.

APPENDIX C
B_{noise} DIAGNOSTICS

Figure 12 supports the noise-scale claims of sections IV-A and IV-B by splitting B_{noise} into its two ingredients. The gradient-covariance trace $\operatorname{tr}\Sigma$ decays slowly and stays within about a factor of 3 over most of training. The squared gradient norm $|G|^2$ collapses by two to three orders of magnitude. The climb of B_{noise} therefore happens because the denominator falls, not because gradient variance explodes. The sharp end-of-run rise in B_{noise} is $|G|^2 \to 0$ during the cooldown, not a real noise blow-up.

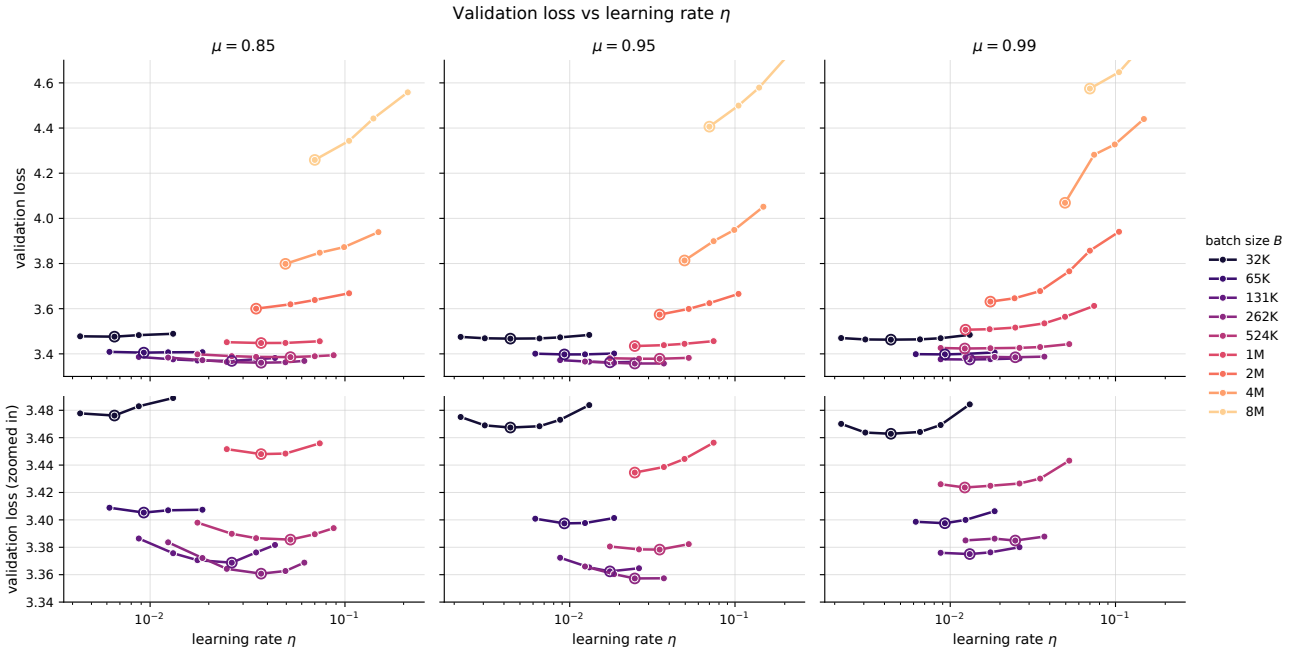


Fig. 5. Per-cell LR sensitivity. Validation loss at the common 1B-token mark versus Muon's learning rate η, one curve per batch size, one column per momentum μ. The lower row re-plots the same data on a zoomed y-axis to separate the overlapping mid-B cluster. Open rings mark each curve's minimum, i.e. the LR-tuned best run that feeds our Pareto fronts. Small- and mid-B cells trace clean U-shapes with bracketed (interior) minima at every μ, so their ringed runs are true LR optima. Large-B cells ($B \geq 2\text{M}$) instead pin to the smallest η on the ladder and rise monotonically across the remaining rungs. The optimum is therefore certainly below the bottom of the swept range and these are the LR-saturated edge picks that inflate the left leg of the frontier (fig. 1).

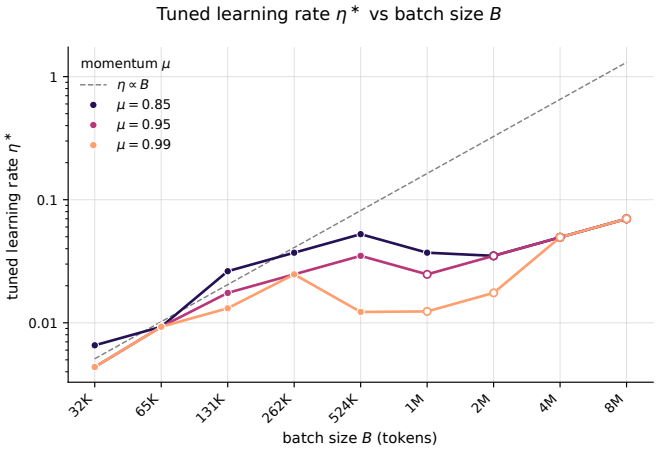


Fig. 6. Tuned Muon learning rates η^* for each (B, μ) cell. The dashed line is the $\eta \propto \sqrt{B}$ guide. Filled markers are bracketed optima (interior LR minima). Open markers are the LR-saturated edge picks of fig. 5. Through mid-B the tuned LR rises faster than $\sqrt{B}$ at $\mu \in \{0.85, 0.95\}$ and overshoots the guide, while $\mu = 0.99$ dips below it between 524K and 1M. At $B \geq 4\text{M}$ all three momenta collapse to the bottom rung of their swept ladder. That rung sits on the $\sqrt{B}$ line.

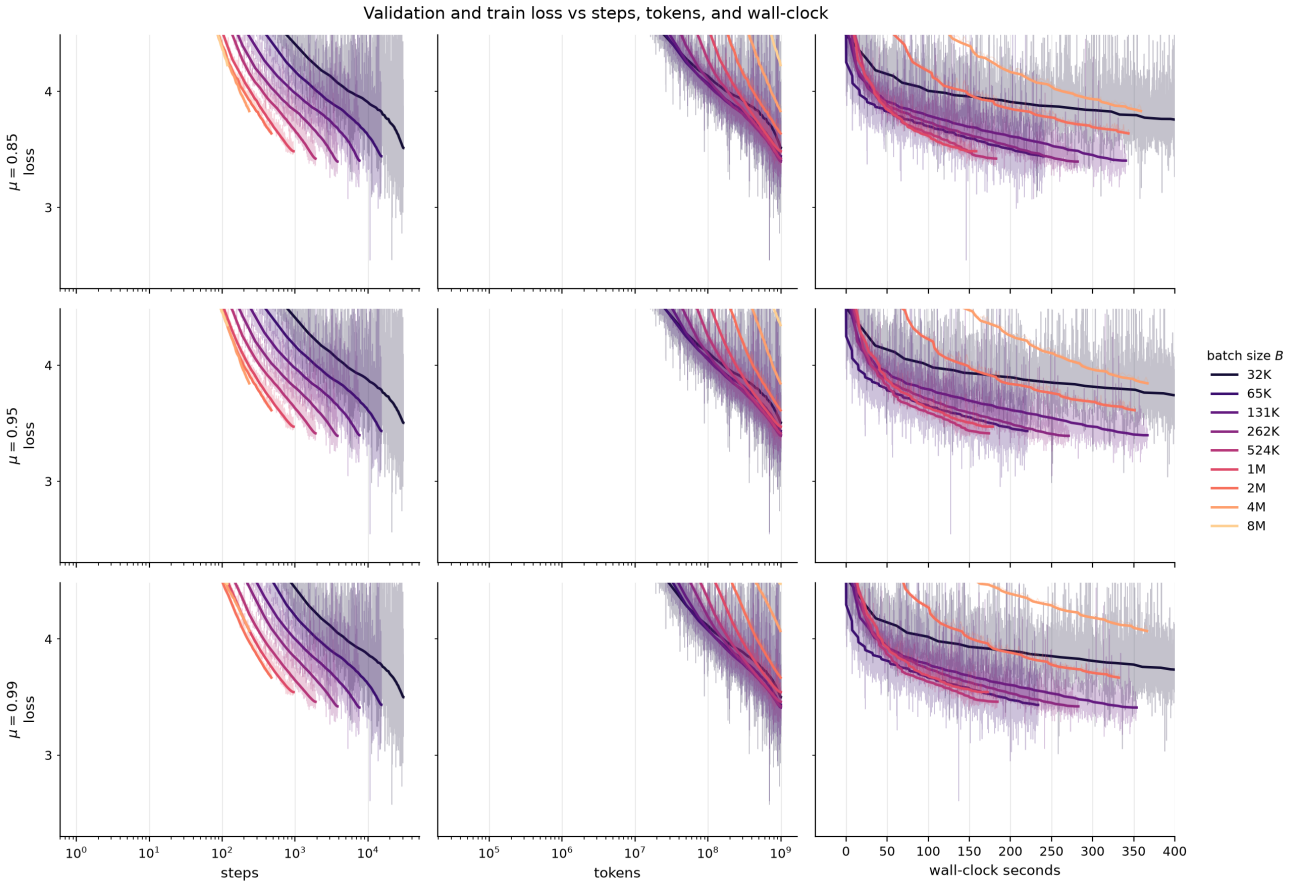


Fig. 7. Validation loss (solid) and per-step train loss (faded) for the LR-tuned best run at each batch size, plotted against optimizer steps, tokens, and wall-clock seconds across columns, with one row per momentum. In the tokens column most batch sizes funnel to comparable loss at 1B tokens. In the wall-clock column the largest batches stay highest and the mid-range batches reach the lowest loss soonest, so large B buys no wall-clock advantage under the cap. Everything is capped at 1B tokens.

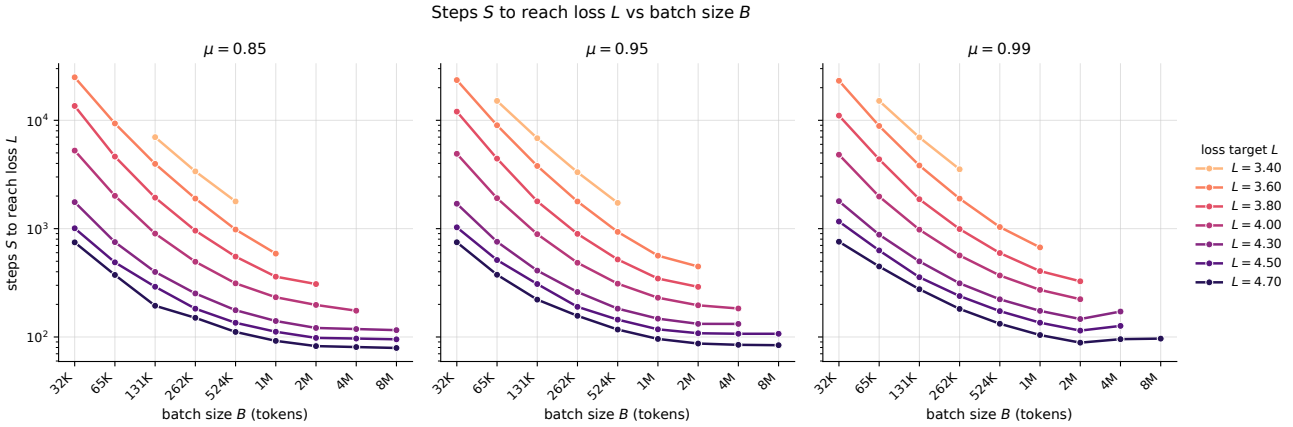


Fig. 8. Raw $S(B)$ data behind fig. 1. Each line gives steps to reach a loss target L for the LR-tuned best run, coloured by L, with one panel per momentum. Lines drop steeply at small B and flatten at large B for $\mu \in \{0.85, 0.95\}$, while at $\mu = 0.99$ the largest-B points turn back up because those LR-saturated cells need more steps. The hardest targets (low L) cover only a mid-range of B, since the smallest batches never reach them and the largest batches run out of budget first.

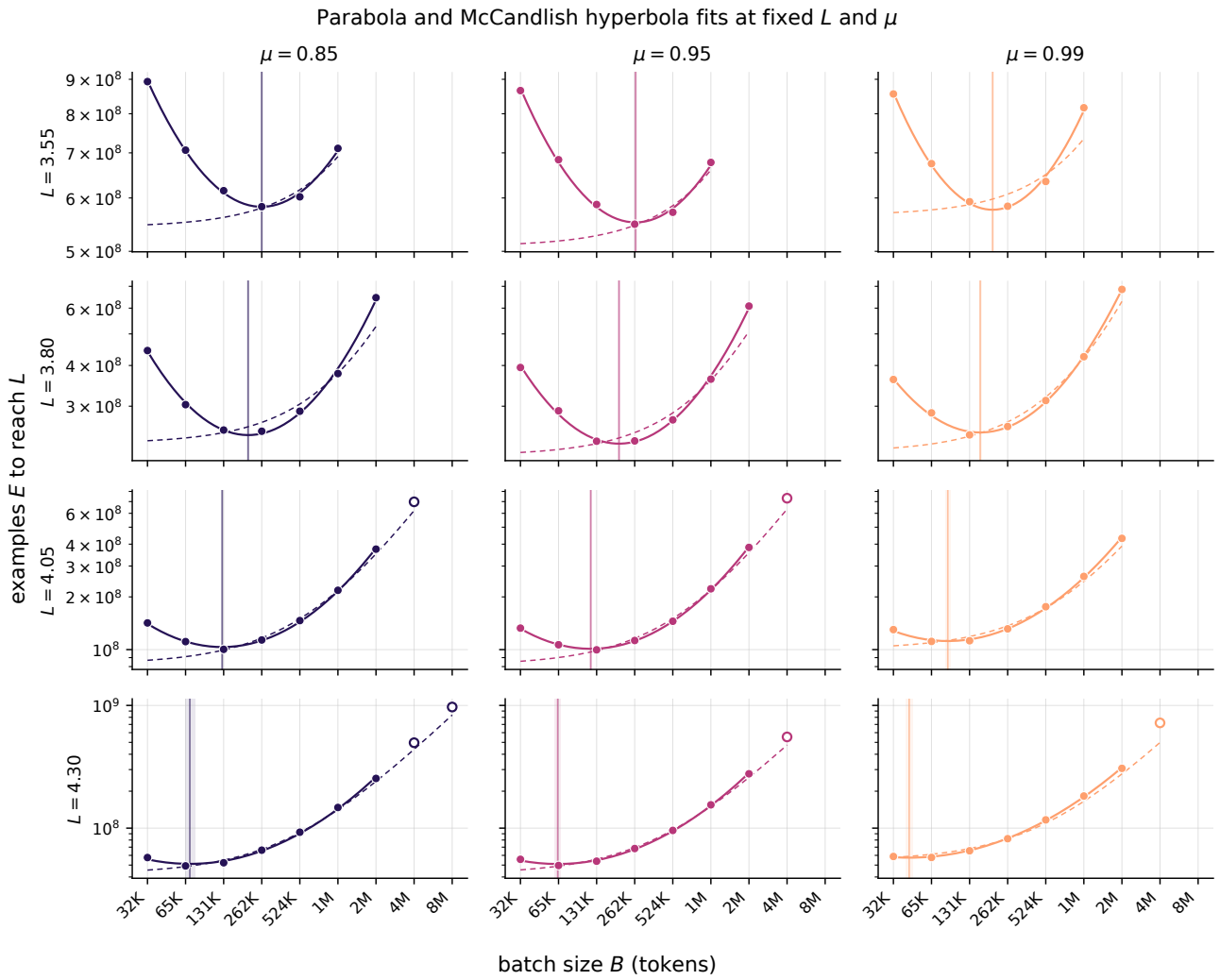


Fig. 9. Per-(L, μ) fit diagnostic. Rows are four representative loss targets and columns are momenta. Each cell shows the (B, E) data, the parabola fit (solid) over the fitted cells, a vertical line at its vertex with a shaded uncertainty band, and the McCandlish hyperbola (dashed) for comparison. Open markers are the LR-saturated cells above 2M held out of the fit. At lower L the U is fully bracketed and the vertex sits in the interior with a tight band. At higher L the vertex B^* falls to the small-batch edge, so the left arm is truncated and the band widens. The hyperbola stays monotone and never bends up the left arm, so it tracks the easy high-L targets but misses the bracketed U at low L.

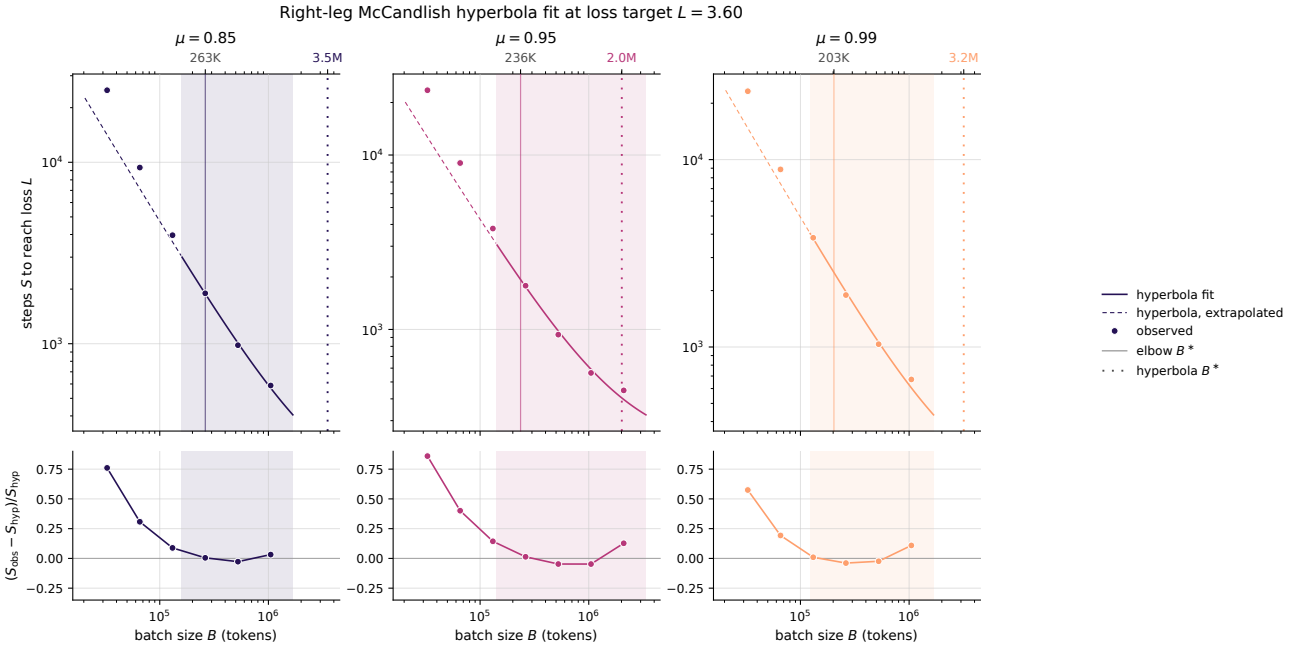


Fig. 10. Right-leg McCandlish hyperbola fit at $L = 3.60$, one column per momentum. The top row plots steps S to reach L against B for the LR-tuned best run. The hyperbola $S = S_{\min}(1 + B^*/B)$ is fit on the right leg alone (shaded, $B \geq 0.6\, B^*_{\text{elbow}}$) and is drawn solid over that leg and dashed where extrapolated to small B. The right-leg points land on the fit, while the small-B points sit well above the extrapolation, which is the left-leg inflation the hyperbola cannot represent. The bottom row gives the relative residual $(S_{\text{obs}} - S_{\text{hyp}})/S_{\text{hyp}}$. It stays near zero across the shaded leg and reaches 0.5 to 0.8 at the smallest B. The solid line marks the elbow B^* (about 200K to 260K) and the dotted line the hyperbola B^* (about 2M to 3.5M). The hyperbola fits the leg it is handed yet places B^* roughly an order of magnitude above the elbow, so it does not give a usable critical batch size.

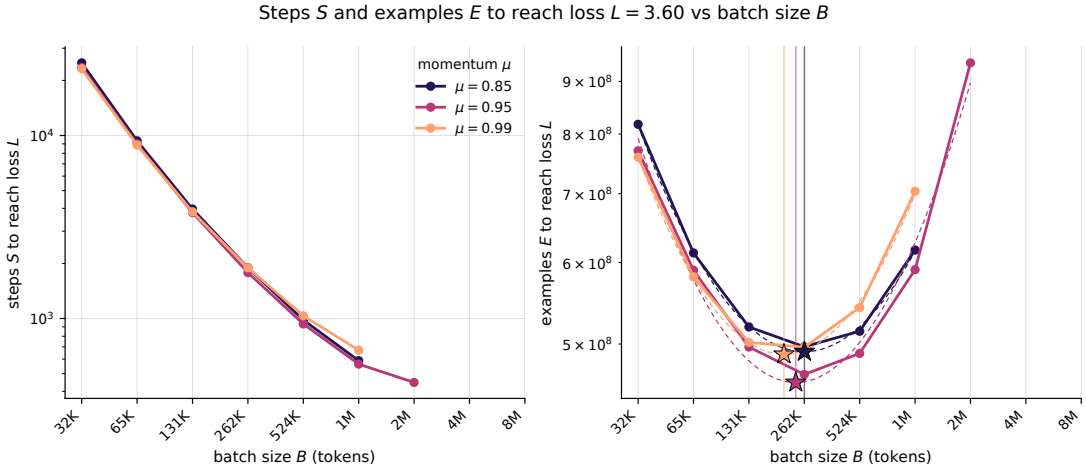


Fig. 11. Steps S (left) and samples E (right) to reach $L = 3.60$ against batch size B, with all three momenta overlaid. The three $S(B)$ curves on the left nearly coincide, so momentum barely moves S at a fixed loss. On the right $E = S \cdot B$ turns those small differences into a visible shift of the elbow. The points are the observed $E(B)$ and the dashed lines are the parabola fits. A star sits at each fit's vertex, the operational B^*. The elbow moves left from $\mu = 0.85$ to $\mu = 0.99$, so higher momentum lowers the critical batch.

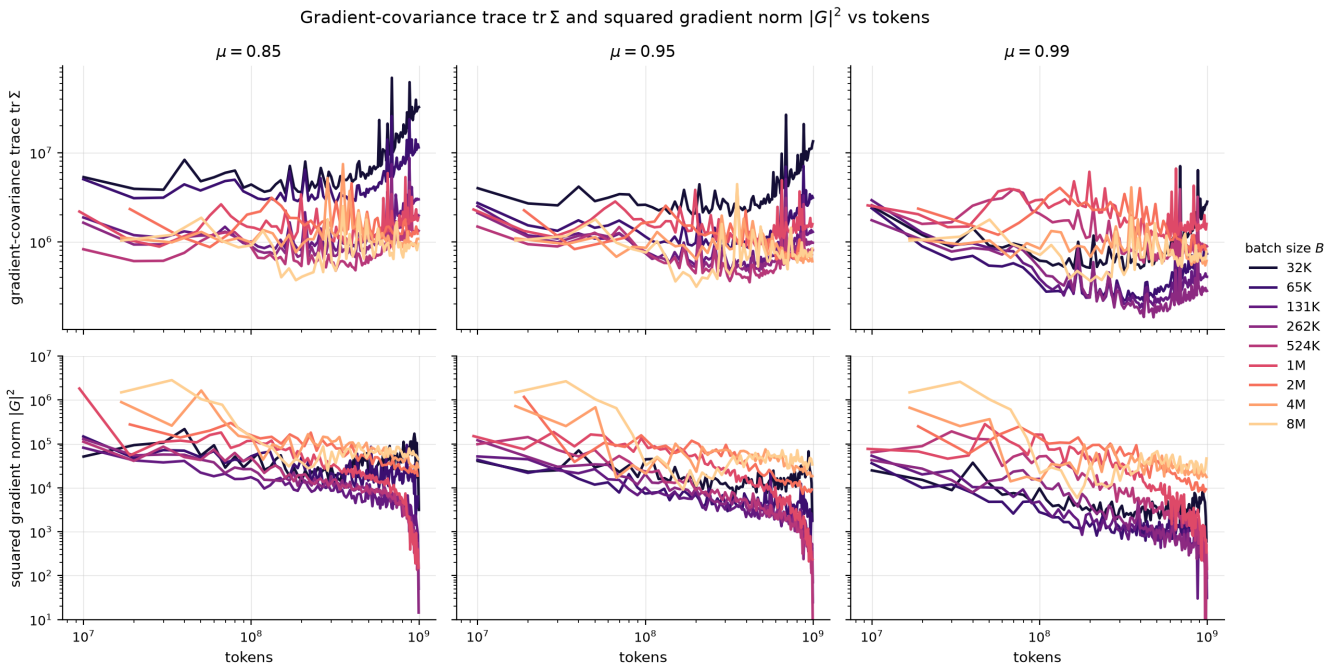


Fig. 12. The two ingredients of $B_{\text{noise}} = (\operatorname{tr}\Sigma/|G|^2)\cdot\text{seq_len}$, plotted on their own. The top row is the gradient-covariance trace $\operatorname{tr}\Sigma$, which decays slowly and stays somewhat stable over most of training. The bottom row is the squared gradient norm $|G|^2$, which collapses by the end of cooldown. The climb of B_{noise} across training therefore happens because the denominator falls, not because gradient variance explodes. The sharp end-of-run rise is $|G|^2 \to 0$ during the cooldown, not a real noise blow-up. Large batches have delayed cooldowns, so their $|G|^2$ stays higher, but eventually collapses too by the end of training on 2B tokens.