THE BEST OPTIMIZER DEPENDS ON BATCH SIZE

Anonymous authors
Paper under double-blind review

ABSTRACT

A plethora of new adaptive optimizers are designed to efficiently estimate and use minibatch gradient statistics to shape parameter updates, but they are typically benchmarked at a single batch size. Hyperparameter scaling rules promise to preserve performance as batch size and gradient noise change, suggesting that the best optimizer at one batch size should remain the best at another. We challenge this approach to developing and evaluating optimizers by showing: (1) no principled scaling rule for Muon works consistently across training settings, and (2) the best optimizer for language model pretraining changes with batch size even after extensive hyperparameter tuning. To understand these failures, we analyze a noisy quadratic model and show that the batch-size scaling rule that preserves performance is direction-dependent, shifting from square-root to linear as a direction's curvature-to-noise ratio rises. We further prove that gradient noise drives a bias–variance tradeoff in preconditioning that can reverse optimizer rankings across batch sizes, even under optimal learning rate and momentum tuning. Consistent with this, when we increase the batch size $16\times$ partway through language model pretraining, keeping small-batch updates in only the sharpest 0.001% of directions removes up to 59.5% of the resulting loss gap over the next 8% of training, while a random subspace of the same size barely helps. Taken together, our results show that principled optimizer design must account for batch size and empirical benchmarking should be performed across batch sizes.

1 INTRODUCTION

Larger batches can increase training throughput, but retaining optimization performance is challenging and often requires adjusting hyperparameters (Goyal et al., 2017; Shallue et al., 2019). Batch-size scaling rules prescribe these adjustments without requiring additional tuning (Section 2.1). These rules are derived analytically and intended to apply broadly across training settings (Malladi et al., 2022; Mlodozeniec et al., 2026; Shulgin et al., 2026), but different derivations yield conflicting prescriptions for the same optimizer (Table 1). We ask whether any of these prescriptions, or alternatives beyond them, can fulfill this promise:

Q1: *Can a hyperparameter scaling rule consistently work in different training settings?*

We study this question with Muon (Jordan et al., 2024) (Definition 1). We derive scaling rules by applying prior theoretical frameworks designed to either preserve training dynamics via SDE or minimize convergence bounds (Definitions 2 and 3), and compare them with a broad grid of candidate rules in language modeling and image classification (Section 3.2). Neither theoretical prescription consistently gives the best performance (Figure 2) and in language modeling both fall behind the trivial baseline of leaving the hyperparameters unchanged at the largest batch sizes. Even among the broader set of candidates, the best rule differs between tasks (Table 1) and between target batch sizes within a task (Figure 2).

Beyond how to adjust Muon's hyperparameters, we ask whether batch size changes its performance relative to other optimizers. Adaptive optimizers differ in how they shape updates using minibatch estimates of quantities such as gradient second moments (Kingma & Ba, 2015; Gupta et al., 2018; Morwani et al., 2025; Wang & Aitchison, 2024). Batch size controls the noise in these estimates, potentially changing which form of adaptation is most effective (Wang & Aitchison, 2024; Zhang et al., 2019). Established benchmarks like the Modded-NanoGPT optimization benchmark (Jordan, 2026), AlgoPerf (Dahl et al., 2023) and Fantastic Optimizers (Wen et al., 2026b) rank optimizers without testing their performance across batch sizes, leaving open the question:

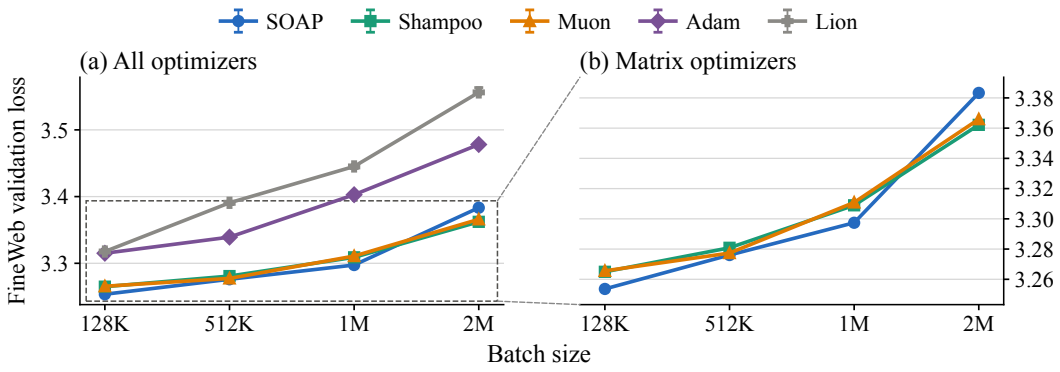

Figure 1: **The best optimizer changes with batch size.** FineWeb validation loss from the Modded-NanoGPT optimization benchmark, pretrained on a fixed budget of 1.7B tokens with decoupled weight decay; every optimizer is tuned independently at every batch size. (a) Validation loss with respect to batch size for all five optimizers. (b) The performance of three matrix optimizers on a zoomed loss axis. SOAP is the best optimizer from 128K to 1M tokens, ahead by up to 0.012 nats, yet at 2M it falls to third, 0.021 nats behind Shampoo. Rankings cross between other pairs of optimizers too: Muon beats Shampoo by 0.0035 nats at 512K, but Shampoo beats Muon by 0.0037 nats at 2M. The HyperBall (Wen et al., 2026a) counterpart is Figure 16.

Q2: *Do optimizer rankings persist across batch sizes after extensive retuning?*

We compare Adam (Kingma & Ba, 2015), Lion (Chen et al., 2023), Muon (Jordan et al., 2024), SOAP (Vyas et al., 2025), and Shampoo (Gupta et al., 2018) in the Modded-NanoGPT optimization benchmark setting, independently tuning each at every batch size using coordinate descent over hyperparameters (Wen et al., 2026b) (Section 4). We observe the *optimizer crossover* phenomenon: optimizer rankings reverse across batch sizes: SOAP performs best at smaller batches, whereas Shampoo performs best at the largest batch size tested. This crossover occurs under both decoupled weight decay and HyperBall norm control (Wen et al., 2026a) (Figures 1 and 16).

Motivated by recent evidence that quadratic models capture the local optimization dynamics of language model pretraining (Meterez et al., 2026), we study a noisy quadratic model (NQM, Zhang et al. (2019)) with direction-dependent curvature and gradient noise to understand why scaling rules do not transfer consistently and why optimizer rankings reverse (Section 5). Tuning the learning rate on SignSGD-M in one direction shows that its scaling with batch size shifts from approximately square-root toward linear as the direction's curvature-to-noise ratio increases (Section 5.1). A single scaling exponent therefore cannot match the individually tuned learning rates across directions. When we compare SGD and Newton's method, we find that Newton's preconditioning trades faster reduction of initialization bias for greater variance from gradient noise (Zhang et al., 2019) (Section 5.2). We prove that this bias–variance tradeoff can favor SGD at small batches and Newton's method at large batches, even with independently optimal learning rate and momentum (Theorem 5.1).

We test whether this direction-dependent scaling appears in language-model pretraining (Section 6). Following Merrill et al. (2025), we branch from intermediate checkpoints of a tuned MuonW run, increasing the batch size from 128K to 2M tokens with square-root learning-rate scaling. Retaining small-batch updates in the sharpest 768 of roughly 85M directions of a preconditioned Hessian proxy, while using large-batch updates elsewhere, removes 35–59.5% of the loss gap to a small-batch control over the next 8% of training (Figure 5). A random subspace of the same size changes the gap by at most 3%. Thus, a tiny set of sharp directions carries a disproportionate share of the local penalty from applying one learning-rate exponent to every direction. Because the quadratic approximation is local, we assess loss shortly after the switch rather than final training loss (Wen et al., 2025).

Altogether, our results challenge established practices in optimizer evaluation and design. Comparisons at a single batch size are not sufficient to establish the efficacy of an optimizer, and we recommend performing extensive hyperparameter tuning across several batch sizes. Optimizer design must likewise go beyond efficiently estimating and using landscape statistics to account for how batch size changes the benefits and costs of adaptation.

2 PRELIMINARIES

2.1 SCALING RULES

A batch-size scaling rule prescribes how to adjust hyperparameters tuned at a reference batch size B_0 for use at $B = \kappa B_0$. Scaling rules are thought to be specific to a given optimizer but generalize across empirical learning settings. For a hyperparameter γ at B_0, let $\gamma^{(\kappa)}$ denote its prescribed value at κB_0. Typically, $\kappa > 1$ due to the potential throughput gains from larger batches. We fix the training budget τ (tokens for language modeling or examples for image classification), giving τ/B updates. The *sample time*, $s = kB$, is the data processed after k updates. Table 1 compares prescriptions from two approaches; Section A discusses broader related work.

Invariant-preservation View. This view adjusts hyperparameters to preserve a training property, such as an approximate parameter trajectory or the amount of data remembered by a moving average. Different invariants can yield different scaling rules.

An SDE (stochastic differential equation) approximates an optimizer's random updates by continuous-time dynamics. For SGD on loss L, let $\Sigma(x)$ be the individual-gradient covariance at parameters x; averaging B independent gradients gives covariance $\Sigma(x)/B$. Under suitable conditions (Li et al., 2018; 2021), at time $t = k\eta$, an SDE approximation of SGD is $\mathrm{d}X_t = -\nabla L(X_t)\,\mathrm{d}t + \sqrt{\eta/B}\,\Sigma(X_t)^{1/2}\,\mathrm{d}\mathcal{W}_t$. Here X_t denotes the parameters and $\mathcal{W}_t$ is Brownian motion. Preserving η/B gives the *linear scaling rule*, $\eta^{(\kappa)} = \kappa\eta$ (Goyal et al., 2017; Smith et al., 2018; Li et al., 2021). Since $t = (\eta/B)s$ is also preserved, the SDE predicts the same parameter distribution and expected loss at equal sample time. Matching the coupled SDE for parameters and optimizer state (e.g. moment estimates) yields scaling rules for Adam (Malladi et al., 2022) and AdamW (Mlodozeniec et al., 2026).

However, the invariant need not be the full trajectory or even come from an SDE. Monzio Compagnoni et al. (2025) use an SDE to preserve an upper bound on long-time expected loss and the relative speeds of the parameter and moment states. Marek et al. (2025) keep Adam's second-moment memory fixed in sample time, while Bergsma et al. (2025) keep AdamW's weight-decay averaging timescale fixed relative to the training budget.

Bound-minimization View. A second approach starts from a convergence bound on a specified quantity, such as suboptimality or stationarity, expressed in terms of the optimizer's hyperparameters, batch size, and number of updates (Kovalev, 2025; Pethick et al., 2025). Minimizing this bound over the scaled hyperparameters at each batch size, with sample-time budget fixed, gives a scaling rule. Varying the sample budgets or jointly optimizing the hyperparameters can further yield different rules (Shulgin et al., 2026).

2.2 MUONW-ADAMW OPTIMIZER

Following standard practice, we use Muon with decoupled weight decay (MuonW) for matrix parameters and AdamW for embeddings, the output projection, and one-dimensional parameters. Both apply weight decay separately from the gradient update.

Definition 1 (MuonW–AdamW)**.** Let W_t, θ_t be MuonW and AdamW parameters with minibatch gradients G_t, g_t, respectively. For $j \in \{\mathrm{M}, \mathrm{A}\}$, η_j and λ_j denote learning rates and weight-decay coefficients; $\mu, \beta_{1,\mathrm{A}}, \beta_{2,\mathrm{A}}$ are momentum and moment retention coefficients. The Nesterov mixing weight ω multiplies the updated Muon momentum before orthogonalization; this implementation sets $\omega = \mu$. Starting from $M_0 = m_0 = v_0 = 0$, for $t \geq 1$,

$$
\begin{aligned}
M_t &= \mu M_{t-1} + (1-\mu)G_t, & \widetilde{M}_t &= \omega M_t + (1-\omega)G_t,\\
m_t &= \beta_{1,\mathrm{A}} m_{t-1} + (1-\beta_{1,\mathrm{A}})g_t, & v_t &= \beta_{2,\mathrm{A}} v_{t-1} + (1-\beta_{2,\mathrm{A}})g_t^{\odot 2},\\
\widehat{m}_t &= m_t/(1-\beta_{1,\mathrm{A}}^t), & \widehat{v}_t &= v_t/(1-\beta_{2,\mathrm{A}}^t).
\end{aligned}
$$

$$
\begin{aligned}
W_{t+1} &= (1-\eta_{\mathrm{M}}\lambda_{\mathrm{M}})W_t - \eta_{\mathrm{M}}\,\mathrm{Orth}(\widetilde{M}_t),\\
\theta_{t+1} &= (1-\eta_{\mathrm{A}}\lambda_{\mathrm{A}})\theta_t - \eta_{\mathrm{A}}\widehat{m}_t/(\sqrt{\widehat{v}_t}+\epsilon).
\end{aligned}
$$

Here Orth is the Newton–Schulz map with fixed matrix-shape rescaling (Algorithm 3), $\epsilon > 0$ is a fixed stabilizer, and powers, roots, and division in AdamW are elementwise.

Primes denote values at $B' = \kappa B$; p is a free learning-rate exponent in the Power Lines product constraint.

MuonW	η'	μ'	λ'
Invariant preservation			
[BDG+25]*	$\kappa^p \eta$	μ	$\kappa^{1-p}\lambda$
Ours	$\kappa\eta$	μ^κ	λ
Bound minimization			
[SRZ+26]	$\sqrt{\kappa}\,\eta$	μ	–
[BXM26]	$\sqrt{\kappa}\,\eta$	–	–
Ours	$\sqrt{\kappa}\,\eta$	μ	λ
Empirical			
LLM	η	μ	$\kappa\lambda$
CIFAR	$\sqrt{\kappa}\,\eta$	μ	λ

AdamW	η'	β_1'	β_2'	λ'
Invariant preservation				
[MLP+22]	$\sqrt{\kappa}\,\eta$	β_1^κ	β_2^κ	–
[MAB+26]	$\sqrt{\kappa}\,\eta$	β_1^κ	β_2^κ	$\sqrt{\kappa}\,\lambda$
[MLS+25]	–	β_1	β_2^κ	–
[BDG+25]	$\kappa^p\eta$	β_1	β_2	$\kappa^{1-p}\lambda$
[MLI+25]	$\sqrt{\kappa}\,\eta$	$\beta_1^{\sqrt{\kappa}}$	$\beta_2^{\sqrt{\kappa}}$	$\sqrt{\kappa}\,\lambda$
Bound minimization				
[BXM26]	$\sqrt{\kappa}\,\eta$	–	–	–
[SHT+25]	$\sqrt{\kappa}\,\eta$	–	–	–
Stability				
[GZR22]	$\sqrt{\kappa}\,\eta$	–	–	–
Empirical				
LLM	η	β_1	β_2^κ	λ
CIFAR	$\sqrt{\kappa}\,\eta$	β_1	β_2	λ

Table 1: **Batch-size scaling prescriptions at $B' = \kappa B$.** Entries give target values; "–" means unspecified. We extend Power Lines (Bergsma et al., 2025) to Muon assuming fixed momentum. The original work favors $p = 0$ for AdamW, which we also found to be effective for MuonW in our empirical sweep. Fixed Adam moments in Power Lines and fixed β_1 in Marek et al. (2025) are empirical choices. Bu et al. (2026); Schaipp et al. (2025) concern SGD on convex optimization, with empirical extensions to the optimizers shown. For a moment retention coefficient β near 1 and fixed exponent a, we use $\beta^a \approx 1 - a(1-\beta)$ to display moment transfers as powers; Appendix B.4 gives the details and original formulas. Empirical rows are from results in Section 3.2.

3 Deriving and Testing Batch-Size Scaling Rules

We apply the two approaches in Section 2.1 to derive prescriptions for Muon's learning rate, momentum, and weight decay. We test these theoretically motivated prescriptions within a search of over 200 candidate scaling rules for MuonW–AdamW, combining plausible choices for how its learning rates, weight decay, and momentum and moment-estimation coefficients change with batch size. Surprisingly, despite the setting-agnostic promise of both derivation styles, the best scaling rule depends on the precise training setting. Neither theoretical prescription identifies a rule that is best in both settings. Appendix B gives the derivations, assumptions, and detailed comparisons.

3.1 Two Scaling Rules for Muon

We apply both frameworks discussed in Section 2 to derive scaling rules for the MuonW-AdamW hybrid optimizer. We summarize our candidate scaling rules alongside ones proposed by prior works in Table 1.

Candidate 1: Matching a Muon SDE. Relative to the SGD example, our MuonW update carries momentum, orthogonalizes its update direction, and applies decoupled weight decay. We therefore model the matrix parameter W_s and momentum M_s together at sample time $s = kB$. Write $\bar\eta = \eta_{\mathrm{M}}/B$ for the learning rate per sample and $\rho = -\log\mu/B$ for the momentum decay per sample. The mean input to orthogonalization is $q_s = \omega M_s + (1-\omega)\nabla L(W_s)$. For this calculation, ω is held fixed as μ changes; at B_0, it equals μ. Using the individual-gradient covariance Σ from Section 2.1, a first-order expansion of orthogonalization, with derivative $D\,\mathrm{Orth}$, gives

$$\mathrm{d}M_s = \rho\big(\nabla L(W_s) - M_s\big)\,\mathrm{d}s + \rho\Sigma(W_s)^{1/2}\,\mathrm{d}\mathcal{W}_s, \tag{1}$$

$$\mathrm{d}W_s = -\bar\eta\big(\mathrm{Orth}(q_s) + \lambda_{\mathrm{M}} W_s\big)\,\mathrm{d}s - \bar\eta(1-\omega)D\,\mathrm{Orth}(q_s)\Sigma(W_s)^{1/2}\,\mathrm{d}\mathcal{W}_s. \tag{2}$$

At a fixed parameter–momentum state, the gradient, its covariance, and the orthogonalization map do not change with B. With ω fixed, we match the displayed SDEs by keeping ρ, $\bar\eta$, and the weight-decay coefficient $\bar\eta\lambda_{\mathrm{M}}$ fixed. At $B = \kappa B_0$, fixed $\rho = -\log\mu/B$ gives $\mu^{(\kappa)} = \mu^\kappa$, and fixed $\bar\eta = \eta_{\mathrm{M}}/B$ gives $\eta_{\mathrm{M}}^{(\kappa)} = \kappa\eta_{\mathrm{M}}$. Because $\bar\eta$ is fixed, $\bar\eta\lambda_{\mathrm{M}}$ stays fixed when λ_{M} does too.

Definition 2 (SDE scaling candidate for Muon)**.** When changing batch size from B_0 to κB_0, use

$$\eta_{\mathrm{M}}^{(\kappa)} = \kappa\eta_{\mathrm{M}}, \qquad \mu^{(\kappa)} = \mu^\kappa, \qquad \lambda_{\mathrm{M}}^{(\kappa)} = \lambda_{\mathrm{M}}. \tag{3}$$

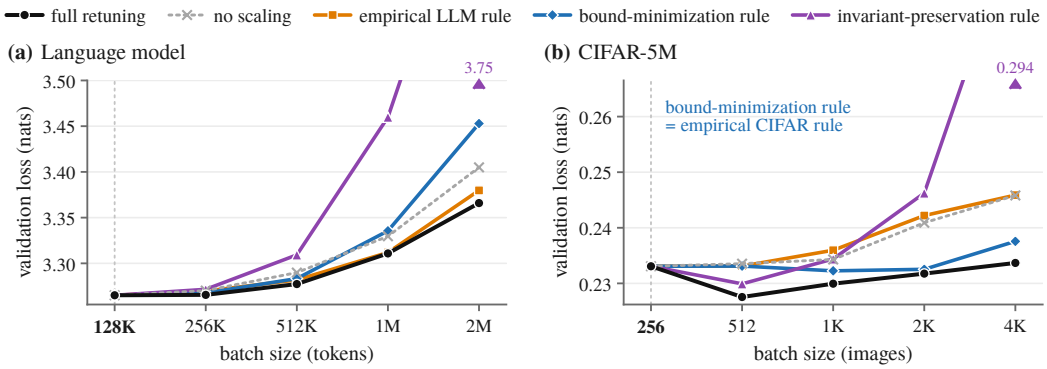

Figure 2: **No single scaling rule transfers well in both settings.** We transfer a tuned configuration from a reference batch to larger batches using the rules in Table 1: (a) MuonW–AdamW language-model pretraining on 1.7B tokens, with a reference batch of 128K tokens; (b) a ResNet with GroupNorm on CIFAR-5M, with a reference batch of 256 images. The theoretical rules follow Definitions 2 and 3 for the matrix parameters, with AdamW counterparts chosen following prior work on SDE and bound minimization. *Full retuning* retunes every hyperparameter at each batch by coordinate descent. In language modeling, the empirical LLM rule is consistent with Power Lines (Bergsma et al., 2025) and stays closest to full retuning, while both theoretical rules fall behind no scaling from 1M tokens on. On CIFAR-5M, the bound-minimization rule instead stays within 0.006 nats of full retuning, while the empirical LLM rule does no better than no scaling. CIFAR-5M points are medians over three seeds. Section E gives full results.

Our MuonW implementation instead ties ω to μ at every batch size, so this rule is a candidate for that optimizer rather than an exact invariance guarantee. Appendix B.2 proves the coefficient-matching result and gives a finite-horizon approximation bound under explicit noise and regularity assumptions.

Candidate 2: Minimizing the Muon convergence bound. For this candidate, we compare the minimizers of a bound on Muon's Frank–Wolfe gap, a measure of stationarity. With exact orthogonalization and decoupled weight decay, Muon selects the matrix in a spectral-norm ball most aligned with the negative momentum direction, then moves toward it by an effective step size $\gamma_{\mathrm{FW}} = \eta_{\mathrm{M}}\lambda_{\mathrm{M}}$ (Sfyraki & Wang, 2025). We hold momentum and weight decay fixed. Let $\mathcal{C}$ denote this spectral-norm ball and $\mathcal{G}_{\mathcal{C}}$ its Frank–Wolfe gap, measuring the largest first-order decrease within the ball (formally defined in Appendix B.3). For W_a sampled uniformly from the τ/B iterates, we prove that there exist constants $c_1, c_2 > 0$ independent of B and γ_{FW}, and a remainder $r(B)$ independent of γ_{FW}, such that $\mathbb{E}\mathcal{G}_{\mathcal{C}}(W_a) \le \frac{c_1 B}{\tau\gamma_{\mathrm{FW}}} + c_2\gamma_{\mathrm{FW}} + r(B)$.

The first term spreads the initial objective gap over only τ/B updates, so it falls as γ_{FW} grows; the second is a smoothness cost that rises with the step size. Ignoring other constraints, their balance gives $\gamma_{\mathrm{FW},\star}(B) = \sqrt{c_1 B/(\tau c_2)}$, so the bound-minimizing effective step size grows by $\sqrt{\kappa}$ when B grows by κ. With weight decay fixed, the same ratio applies to the learning rate.

Definition 3 (Bound-based scaling candidate for Muon)**.** When changing batch size from B_0 to κB_0, use

$$\eta_{\mathrm{M}}^{(\kappa)} = \sqrt{\kappa}\,\eta_{\mathrm{M}}, \qquad \mu^{(\kappa)} = \mu, \qquad \lambda_{\mathrm{M}}^{(\kappa)} = \lambda_{\mathrm{M}}. \tag{4}$$

The constrained minimizer follows square-root scaling until the effective step-size cap at 1 becomes active. Appendix B.3 gives the full bound, its constraints, and the distinction between exact orthogonalization and the finite Newton–Schulz implementation.

3.2 Sweeping Scaling Rules

We test whether a single scaling rule transfers well across training settings and target batch sizes. For MuonW–AdamW, we evaluate all combinations of fixed, square-root, or linear scaling for the two learning-rate groups and for each weight-decay coordinate, together with fixed or decay-preserving scaling for Muon momentum and Adam's two moment coefficients. This gives $3^3 \times 2^3 = 216$

complete rules in language modeling, where auxiliary weight decay is held fixed[1], and $3^4 \times 2^3 = 648$ on CIFAR-5M (Nakkiran et al., 2021), where the matrix and auxiliary weight decays are scaled separately.

Setting. We transfer a tuned reference configuration to four larger batches in language modeling and six target batches in one-pass CIFAR-5M training with a GroupNorm ResNet. Here we choose CIFAR-5M to focus on optimization effect instead of generalization effect of batch size. The reference batches are 128K tokens and 256 examples, respectively. Within each setting, all rules share the model, data budget, and schedule shape. We select one rule across target batches by its mean batch-size regret, defined as the equally weighted mean of its validation-loss gap to the best tested rule at each batch. Appendix E.1 gives the full protocol and scaling conventions.

The best rule depends on the setting. The selected language-model rule keeps both learning rates fixed and scales matrix weight decay linearly with batch size. The selected CIFAR-5M rule instead scales both learning rates with the square root of batch size and keeps both weight decays fixed. Thus, even for the same optimizer, the two settings favor different scaling prescriptions. The empirically best CIFAR rule coincides with the **bound-minimization** rule and the empirically best language modeling rule at 2M is consistent with the **Power Lines** rule (Bergsma et al., 2025). One possible explanation is that the run on CIFAR-5M is much closer to convergence compared to the language model experiments, which stay far from convergence (Wen et al., 2025). Within language modeling, the best scaling rule also changes across target batch sizes within each setting. The best rules use square-root scaling for both learning rates at 256K and 512K tokens, but fixed learning rates and linear matrix weight-decay scaling at 1M and 2M (Figure 2 and section E).

Keeping momentum coefficients fixed is generally preferable in our sweeps. The best common rules in both settings keep Adam's first-moment coefficient β_1 and Muon's momentum coefficient μ fixed as batch size changes (Section E). This suggests keeping these coefficients fixed as a default when transferring hyperparameters across batch sizes. The observation is consistent with prior theory showing that momentum offers limited benefit over SGD in the small-batch regime when small learning rates and gradient noise govern the dynamics (Wang et al., 2024), and with noisy-quadratic analyses predicting that momentum's gains emerge primarily at larger batch sizes (Zhang et al., 2019). One possible explanation is that momentum becomes useful at large batches by increasing the maximum learning rate that training can tolerate. River-valley theory supports this interpretation by showing that heavy-ball momentum stabilizes larger learning rates in sharp directions, allowing faster progress along a flat, low-loss valley (Lu et al., 2026). This mechanism motivates our interpretation of the Adam and Muon results, although our scaling sweeps do not directly measure their stability thresholds.

4 Batch Size Changes the Best Optimizer

In this section, we test how batch size affects the ranking of optimizers in the NanoGPT benchmark. Because the scaling rules are unreliable, we perform extensive hyperparameter tuning for each optimizer at each batch size. We find frequent *optimizer crossovers*, where the best optimizer at one batch size underperforms at another one. Our results provide substantive evidence that optimizer benchmarking should be performed over a range of batch sizes.

Setting. We follow the setting of the Modded-NanoGPT optimization benchmark (Jordan, 2026) and vary the batch size $B \in \{128\text{K}, 512\text{K}, 1\text{M}, 2\text{M}\}$ tokens while holding the training budget to be roughly 1.7B tokens. We use coordinate descent to tune the hyperparameters extensively (Wen et al., 2026b) and report the details in Section F. As HyperBall (Wen et al., 2026a) performs competitively on the Modded NanoGPT benchmark, we compare two weight-norm control mechanisms applied to the hidden matrices. With learning rate η and weight decay λ, **weight decay (W)** shrinks a matrix parameter W_t by $1 - \eta\lambda$ at each step. **HyperBall (H)** (Wen et al., 2026a) instead steps along the normalized update and renormalizes the result, leaving each matrix's Frobenius norm unchanged and thereby constraining optimization to a fixed-norm hypersphere.

Experiment Results. After independent retuning, SOAP leads at the smallest batch, while Shampoo leads at the largest. We observe that crossovers occur at different batch sizes and across different pairs of optimizers. Section F.5 reports the same change in winner under HyperBall. We also observe

[1] Section E shows that performance is insensitive to how auxiliary weight decay is scaled.

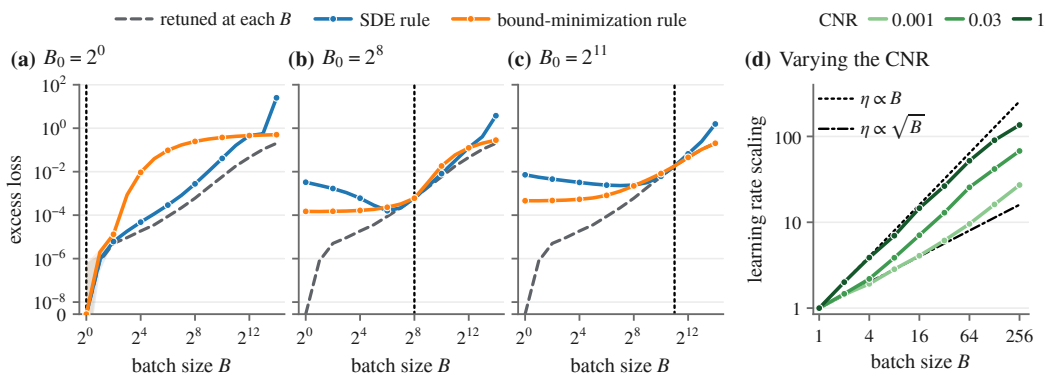

Figure 3: **Which scaling rule works depends on the reference batch size and on the curvature-to-noise ratio (CNR).** (a)–(c) SignSGD with momentum on a three-direction NQM with a fixed sample budget. Gray: hyperparameters retuned at each batch size. Blue and orange: hyperparameters tuned at B_0 (dotted line), then scaled to each batch size using the SDE and bound-minimization rules, respectively. Excess loss is terminal loss minus the retuned loss at $B = 1$. Shaded bands show one standard error across eight evaluation groups. (d) Learning rates of SignSGD with momentum tuned for terminal loss on a one-dimensional NQM, each relative to its own value at $B = 1$; we control the CNR by choosing c and h. Dotted and dash-dotted lines mark linear and square-root scaling. See Sections C.2.1 and C.3.3 for details.

the crossover from SOAP to Shampoo in nanochat setting where we perform hyperparameter transfer following the empirical LLM scaling (Section H). These results show that comparing optimizers at a single batch size provides a limited understanding of their efficacy.

5 Curvature and Noise Shape Batch-Size Scaling

We study the noisy quadratic model (NQM) to isolate how curvature and gradient noise shape batch-size effects. Prior work used this model to analyze the benefits of momentum and preconditioning across batch sizes (Zhang et al., 2019), and recent work supports quadratic approximations of local language-model training dynamics (Meterez et al., 2026).

Definition 4 ((c, h)-noisy quadratic model)**.** For $\theta \in \mathbb{R}^d$, let $h_i > 0$ and $c_i \geq 0$ denote the curvature and single-example gradient-noise variance in coordinate i. The (c, h)-NQM objective and unbiased minibatch gradient at batch size $B \in \mathbb{N}_{>0}$ are

$$L(\theta) = \frac{1}{2}\sum_{i=1}^{d} h_i \theta_i^2, \qquad \widehat{g}_{B,i}(\theta) = h_i\theta_i + \sqrt{\frac{c_i}{B}}\, z_i, \qquad z_i \overset{\text{iid}}{\sim} \mathcal{N}(0,1). \tag{5}$$

Each update uses fresh Gaussian draws, independent of initialization and all previous draws.

In the NQM setting, we note that the Muon optimizer approximately reduces to SignSGD with momentum, because orthogonalization on a 1×1 block is exactly the coordinate-wise sign operation. As such, we primarily analyze SignSGD with momentum in this section, and we ignore weight decay here for simplicity of analysis.

We first show how properties of the optimization problem can change the best scaling rule, supporting our prior finding that different training settings favor different prescriptions (Figure 2 in Section 3). Specifically, increasing the *curvature-to-noise ratio* (CNR), defined below, shifts the best scaling rule for SignSGD with momentum from approximately square-root toward linear.

Definition 5 (Curvature-to-noise ratio (CNR))**.** For a (c, h)-NQM, the curvature-to-noise ratio in coordinate i is squared curvature divided by single-example gradient-noise variance:

$$\mathrm{CNR}_i := h_i^2/c_i, \qquad \text{with } \mathrm{CNR}_i := +\infty \text{ if } c_i = 0. \tag{6}$$

We then address Section 4's finding that optimizer rankings reverse with batch size despite independent retuning. We prove that SGD and Newton's method can exhibit such a reversal on a two-dimensional NQM, because each direction exhibits a different CNR and thus a different bias–variance tradeoff in the preconditioner.

5.1 Scaling Rules Depend on the Training Setting

The efficacy of a scaling rule depends on the reference and target batch sizes. We study SignSGD with momentum on a three-dimensional noisy quadratic under a fixed sample budget. At each reference batch size, we tune the learning rate and momentum, then apply scaling rules derived under either the SDE approximation or the convex convergence analysis (Definitions 2 and 3). We compare the resulting terminal loss with that obtained by independently retuning at the target batch size; Section C.2.1 gives further empirical details. Figure 3(a)–(c) shows that the better scaling rule depends on both the reference and target batch sizes.

The curvature-to-noise ratio affects the correct scaling. On a one-dimensional noisy quadratic, we vary the noise variance while holding curvature, initialization, sample budget, and momentum fixed. We independently tune the learning rate at each batch size, without weight decay. Figure 3(d) shows that increasing CNR shifts the tuned learning rates from approximately square-root toward linear scaling. Experimental details are given in Section C.3.3.

To understand this behavior, recall that SignSGD takes a step of magnitude η in the direction determined by the sign of the momentum average. With momentum retention $\mu \in [0, 1)$, the stationary momentum average at a fixed parameter value w is Gaussian. Its expected sign, $\mathbb{E}[\mathrm{sign}(m)] = \mathrm{erf}\left(hw\sqrt{\frac{1+\mu}{1-\mu} \cdot \frac{B}{2c}}\right)$, determines the expected update. When noise dominates, this term grows as $\sqrt{B}$ and preserving displacement per processed sample motivates square-root scaling. When signal dominates, its magnitude saturates at 1 and linear scaling compensates for fewer updates per sample. Section C.3.2 gives the derivation based on this *movement ansatz*, that is, choosing learning rates that preserve expected displacement per processed sample at a fixed parameter value.

5.2 Batch Size Can Reverse Optimizer Rankings After Retuning

The preceding section shows that directions with different CNRs can require different scaling exponents to match their individually tuned learning rates. We now show that these directional differences can also produce optimizer crossovers even after optimal hyperparameter tuning (Section 4). We compare SGD with Newton's method on a two-dimensional noisy quadratic, where the two directions have different CNRs because one is flat and noisy while the other is sharp and noiseless.

Figure 4: **Optimizer crossover on a noisy quadratic.** Tuned loss on a three-direction NQM, with each optimizer retuned at every batch size; "-M" denotes momentum.

First consider the updates without momentum. Both optimizers take the form

$$w_{t+1,i} = (1 - a_i h_i) w_{t,i} - a_i \sqrt{c_i / B}\, z_{t,i}, \qquad (7)$$

with step size $a_i = \eta$ for SGD and $a_i = \eta / h_i$ for Newton's method. At the same learning rate, SGD reduces initialization error more slowly in flatter directions, whereas Newton's method reduces it at the same rate in every direction while amplifying noise in the flat direction by $1/h_i$. Batch size changes both the noise per update and the number of updates available, changing the balance between these effects. The following theorem establishes that this bias-variance tradeoff can reverse optimizer rankings even after optimal tuning.

Theorem 5.1. *For every sufficiently large sample budget T, there exists a two-dimensional NQM on which optimally tuned SGD outperforms optimally tuned Newton at $B = 1$, but the ranking reverses at $B = T$. This holds both without momentum and with optimally tuned momentum.*

Section C.1.2 provides a complete proof of the theorem as well as empirical evidence of the bias-variance tradeoff. Beyond this construction, a three-dimensional noisy quadratic exhibits the same crossover with every optimizer independently tuned at each batch size (Figure 4), and Section D shows crossovers on three matrix regression tasks.

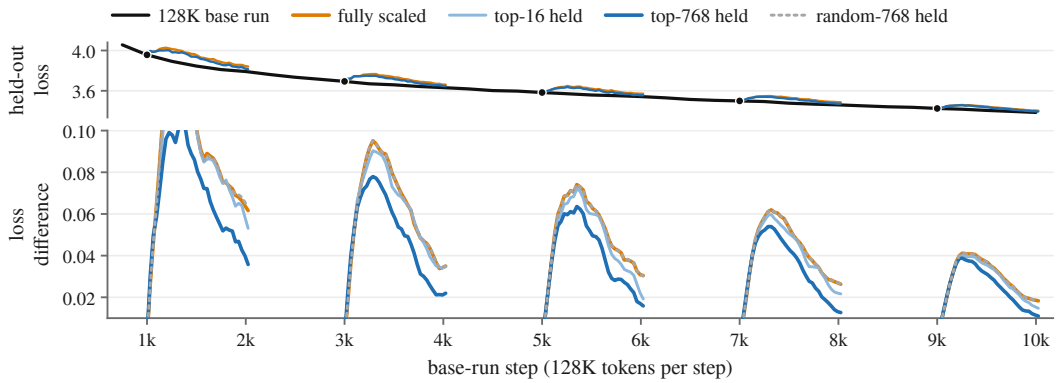

Figure 5: **Holding the sharpest** 0.001% **of directions at** 128K **removes up to** 59.5% **of the** 2M **branch penalty.** Top: held-out loss of the 128K base run (black) and of the fully scaled and top-768 held 2M branches started at steps $1{,}000$, $3{,}000$, $5{,}000$, $7{,}000$, and $9{,}000$. Bottom: held-out loss minus the control branch's loss at each evaluation for the fully scaled, top-16 held, top-768 held, and random-768 held (dotted) branches, evaluated every 32 base steps on 1M held-out tokens and smoothed over three evaluations.

6 Sharp Directions Carry Most of the Local Batch-Size Penalty in Pretraining

Based on our analysis in Section 5, we conjecture that the complex interaction between batch size and optimizers is an artifact of the heterogeneity of the CNR across directions. We validate this in a real language model in this Section. We show that when a scaling rule transfers the model of Section 4 to a larger batch, the resulting loss increase is concentrated in the sharp directions.

Directional branching. Following the critical-batch-size measurement in Merrill et al. (2025), we branch from checkpoints of a base run. The base run is the tuned MuonW run in Section 4 at $B = 128\text{K}$ tokens per step, and we call its checkpoints at every $1{,}000$ steps *anchors*. From each anchor, a branch trains for $1{,}024$ further steps (134M tokens) on the same data as the base run. Let $w_t \in \mathbb{R}^N$ denote the $N \approx 85\text{M}$ hidden-matrix parameters at step t, flattened into one vector. Each branch runs two copies of Muon, both initialized from the anchor's optimizer state and fed the same B-token gradients. The *small-batch* copy computes an update direction $u_t^{(B)}$ from each gradient. The *large-batch* copy computes an update direction $u_t^{(\kappa B)}$ from the mean of the last $\kappa = 16$ gradients once every κ steps, which emulates training at $\kappa B = 2\text{M}$ tokens; we set $u_t^{(\kappa B)} = 0$ at the other steps. Let η_t and λ_t be the base run's matrix learning rate and weight-decay coefficient. Following the bound-minimization rule (Definition 3), the large-batch copy uses the learning rate $\eta_t^{(\kappa B)}$, equal to $\sqrt{\kappa}$ times the mean of η_t over those κ steps. Given a subspace of $\mathbb{R}^N$ with orthogonal projector Q, called the *held subspace*, the branch updates

$$w_{t+1} = (1 - \eta_t \lambda_t)\, w_t - \eta_t\, Q\, u_t^{(B)} - \eta_t^{(\kappa B)}\, (I - Q)\, u_t^{(\kappa B)}.$$

The branch thus takes small-batch updates inside the held subspace and large-batch updates outside it. The *control branch* ($Q = I$) continues the base run at 128K, and the *fully scaled branch* ($Q = 0$) moves every direction to 2M. The remaining parameters keep the base run's AdamW updates.

Choosing the held subspace. We hold the r sharpest directions at the anchor, for $r \in \{16, 768\}$. To measure sharpness, we precondition the Gauss–Newton approximation G of the loss Hessian by the Jacobian P of Muon's orthogonalization map at the anchor. We then take the top-r eigenvectors of PG as the sharpest directions after preconditioning induced by Muon and let Q be the orthogonal projector onto their span (see Section G for details). We prove that a local linearization of Muon's dynamics around the anchor is a noisy quadratic model with the same curvature spectrum (Section G.1), so these are the highest-curvature directions of that model. We ignore the anisotropy in gradient noise in this experiment and leave that for future work. As a baseline, we also hold a random 768-dimensional subspace. The held subspace stays fixed throughout each branch.

Result. We score each branch by its *local branch penalty*: its validation loss at the end of the branch minus that of the control branch from the same anchor. Section G details the construction and tabulates the numbers behind Figure 5. Keeping the base batch in only the sharpest 0.001% of directions removes 35–59.5% of the 2M branch penalty for branches started between steps $1{,}000$ and $9{,}000$ (Table 55), indicating that the bulk of the branch penalty was driven by sharp directions. The recovery share grows with held-rank r and the top 16 directions remove 5–26%. A random 768-dimensional subspace, by contrast, changes the penalty by at most 3% at every starting step. The effect disappears once the learning rate decays at the end of training: for branches started at steps $11{,}000$ and $12{,}000$, no held subspace significantly changes the penalty. This is potentially due to the fact that we ignore gradient noise in the direction selection. Consistent with Section 5, the results indicate that the bound-minimization rule induces different batch-size penalties across directions. Our results suggest that future work on batch-size scaling should account for variation in CNR across directions and explore how to exploit it.

AI USE STATEMENT

In this work, we used generative AI tools to help develop theoretical models and conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, implement methods, and support qualitative and thematic data analysis. We have not used generative AI tools to generate synthetic data sets, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean or reformat datasets, or interpret results. Additionally, we used generative AI tools to create or modify scientific figures, create or edit software code, draft parts of the paper, edit the paper to improve readability, search for information and identify relevant literature, and format references. We have reviewed all AI-assisted work and checked every proof line by line. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the assumptions, derivations, and proofs for our scaling-rule and noisy-quadratic results in Sections B and C, and for the local noisy-quadratic model of Muon in Section G.1. The synthetic experimental settings and tuning procedures are described in Sections C and D. For the CIFAR-5M and language-model experiments, Sections E, F and H document the training settings, hyperparameter searches, evaluation protocols, and detailed results. Section G specifies the directional-branching procedure and reports the measurements underlying the corresponding figure. We also provide an anonymized code archive as supplementary material. It contains the language-model trainer for all five optimizers with the tuned configuration for every optimizer and batch size, generators for the 216 language-model and 648 CIFAR-5M scaling rules, the CIFAR-5M trainer, the coordinate-descent tuner used for full retuning, the directional-branching implementation, and the noisy-quadratic-model and matrix-model simulations.

REFERENCES

Shun-ichi Amari, Jimmy Ba, Roger Grosse, Xuechen Li, Atsushi Nitanda, Taiji Suzuki, Denny Wu, and Ji Xu. When does preconditioning help or hurt generalization? In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=S724o4_WB3.

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. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=bFXbLQzRoZ.

Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal LR across token horizons. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2409.19913.

Zhiqi Bu, Shiyun Xu, and Jialin Mao. Convex dominance in deep learning I: A scaling law of loss and learning rate. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dSdLqg02tx.

Franz Louis Cesista and Kaiyue Wen. Critical batch size for steepest descent under arbitrary norms, November 2025. URL https://leloykun.github.io/ponder/steepest-descent-crit-bz/. Blog post, November 22.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49205–49233, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/9a39b4925e35cf447ccba8757137d84f-Abstract-Conference.html.

Dami Choi, Christopher J. Shallue, Zachary Nado, Jaehoon Lee, Chris J. Maddison, and George E. Dahl. On empirical comparisons of optimizers for deep learning, 2019. URL https://arxiv.org/abs/1910.05446.

George E. Dahl, Frank Schneider, Zachary Nado, Naman Agarwal, Chandramouli Shama Sastry, Philipp Hennig, Sourabh Medapati, Runa Eschenhagen, Priya Kasimbeg, Daniel Suo, Juhan Bae, Justin Gilmer, Abel L. Peirson, Bilal Khan, Rohan Anil, Mike Rabbat, Shankar Krishnan, Daniel Snider, Ehsan Amid, Kongtao Chen, Chris J. Maddison, Rakshith Vasudev, Michal Badura, Ankush Garg, and Peter Mattson. Benchmarking neural network training algorithms, 2023. URL https://arxiv.org/abs/2306.07179.

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

Katie Everett, Lechao Xiao, Mitchell Wortsman, Alexander A. Alemi, Roman Novak, Peter J. Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, and Jeffrey Pennington. Scaling exponents across parameterizations and optimizers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, 2024. URL https://arxiv.org/abs/2407.05872.

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

Diego Granziol, Stefan Zohren, and Stephen Roberts. Learning rates as a function of batch size: A random matrix theory approach to neural network training. *Journal of Machine Learning Research*, 23(173):1–65, 2022. URL https://jmlr.org/papers/v23/20-1258.html.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1842–1850, 2018. URL https://proceedings.mlr.press/v80/gupta18a.html.

Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17086–17098, 2022. URL https://arxiv.org/abs/2110.00641.

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, 2026. URL https://arxiv.org/abs/2603.21191.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in SGD, 2017. URL https://arxiv.org/abs/1711.04623.

Tyler B. Johnson, Pulkit Agrawal, Haijie Gu, and Carlos Guestrin. AdaScale SGD: A user-friendly algorithm for distributed training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 4911–4920, 2020. URL https://proceedings.mlr.press/v119/johnson20a.html.

Keller Jordan. Modded-NanoGPT Optimization Benchmark, 2026. URL https://github.com/KellerJordan/modded-nanogpt/tree/master/records/track_3_optimization.

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

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

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

Frederik Kunstner, Lukas Balles, and Philipp Hennig. Limitations of the empirical fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/46a558d97954d0692411c861cf78ef79-Abstract.html.

Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between SGD and Adam on transformers, but sign descent might be. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=a65YK0cqH8g.

Kiwon Lee, Andrew N. Cheng, Elliot Paquette, and Courtney Paquette. Trajectory of mini-batch momentum: Batch size saturation and convergence in high dimensions. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/efcb76ac1df9231a24893a957fcb9001-Abstract-Conference.html.

Qianxiao Li, Cheng Tai, and Weinan E. Stochastic modified equations and dynamics of stochastic gradient algorithms I: mathematical foundations. *CoRR*, abs/1811.01558, 2018. URL http://arxiv.org/abs/1811.01558.

Shuaipeng Li, Penghao Zhao, Hailin Zhang, Xingwu Sun, Hao Wu, Dian Jiao, Weiyan Wang, Chengjun Liu, Zheng Fang, Jinbao Xue, Yangyu Tao, Bin Cui, and Di Wang. Surge phenomenon in optimal learning rate and batch size scaling. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/ef74413c7bf1d915c3e45c72e19a5d32-Abstract-Conference.html.

Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1454–1466, 2018. doi: 10.1109/TNNLS.2017.2672978. URL https://arxiv.org/abs/1512.04202.

Zhiyuan Li, Sadhika Malladi, and Sanjeev Arora. On the validity of modeling SGD with stochastic differential equations (SDEs). In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.nips.cc/paper/2021/hash/69f629564298659092 1fa916d61c1f8-Abstract.html.

Wu Lin, Scott C. Lowe, Felix Dangel, Runa Eschenhagen, Zikun Xu, and Roger B. Grosse. Understanding and improving Shampoo and SOAP via Kullback–Leibler minimization. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=pQQuC1nIQq.

Miao Lu, Zeyu Bian, Kaiyue Wen, Beining Wu, Siyu Chen, Tianhao Wang, and Zhiyuan Li. Towards understanding momentum acceleration in river-valley loss landscape, 2026. URL https://arxiv.org/abs/2609.30957.

Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the SDEs and scaling rules for adaptive gradient algorithms. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/

2022/hash/32ac710102f0620d0f28d5d05a44fe08-Abstract-Conference.html.

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 *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2507.07101.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training, 2018. URL https://arxiv.org/abs/1812.06162.

William Merrill, Shane Arora, Dirk Groeneveld, and Hannaneh Hajishirzi. Critical batch size revisited: A simple empirical approach to large-batch language model training. In *Advances in Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=XUKUx7Xu89.

Alexandru Meterez, Pranav Ajit Nair, Depen Morwani, Cengiz Pehlevan, Sham Kakade, and Alex Damian. A defense of the quadratic model, 2026. URL https://arxiv.org/abs/2607.21716.

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

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. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2411.15958.

Depen Morwani, Itai Shapira, Nikhil Vyas, Eran Malach, Sham M. Kakade, and Lucas Janson. A new perspective on Shampoo's preconditioner. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.17748.

Preetum Nakkiran, Behnam Neyshabur, and Hanie Sedghi. The deep bootstrap framework: Good online learners are good offline generalizers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=guetrIHLFGI.

Guilherme Penedo et al. The FineWeb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

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

Shikai Qiu, Zixi Chen, Hoang Phan, Qi Lei, and Andrew Gordon Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales. In *Advances in Neural Information Processing Systems*, 2025. URL https://arxiv.org/abs/2512.05620.

Naoki Sato, Hiroki Naganuma, and Hideaki Iiduka. Convergence bound and critical batch size of Muon optimizer, 2025. URL https://arxiv.org/abs/2507.01598.

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. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 53267–53294, 2025. URL https://proceedings.mlr.press/v267/schaipp25a.html.

Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pp. 343–351, 2013. URL https://proceedings.mlr.press/v28/schaul13.html.

Robin M. Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley: Benchmarking deep learning optimizers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 9367–9376, 2021. URL https://proceedings.mlr.press/v139/schmidt21a.html.

Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. Benchmarking optimizers for large language model pretraining, 2025. URL https://arxiv.org/abs/2509.01440.

Maria-Eleni Sfyraki and Jun-Kun Wang. Lions and muons: Optimization via stochastic frank-wolfe, 2025. URL https://arxiv.org/abs/2506.04192.

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. URL https://jmlr.org/papers/v20/18-789.html.

Egor Shulgin, Dimitri von Rütte, Tianyue H. Zhang, Niccolò Ajroldi, Bernhard Schölkopf, and Antonio Orvieto. Deriving hyperparameter scaling laws via modern optimization theory, 2026. URL https://arxiv.org/abs/2603.15958.

Samuel L. Smith and Quoc V. Le. A Bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1710.06451.

Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1Yy1BxCZ.

Teodora Srećković, Jonas Geiping, and Antonio Orvieto. Is your batch size the problem? Revisiting the Adam-SGD gap in language modeling, 2025. URL https://arxiv.org/abs/2506.12543.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf.

Neha Wadia, Daniel Duckworth, Samuel S. Schoenholz, Ethan Dyer, and Jascha Sohl-Dickstein. Whitening and second order optimization both make information in the dataset unusable during training, and can reduce or prevent generalization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 10617–10629, 2021. URL https://proceedings.mlr.press/v139/wadia21a.html.

Runzhe Wang, Sadhika Malladi, Tianhao Wang, Kaifeng Lyu, and Zhiyuan Li. The marginal value of momentum for small learning rate SGD. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3JjJezzVkT.

Xi Wang and Laurence Aitchison. Batch size invariant Adam, 2024. URL https://arxiv.org/abs/2402.18824.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape view. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.05192.

Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them II: Hyperball optimization, 2026a. URL https://arxiv.org/abs/2606.16899.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=2J51qUZ0iG.

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1MczcgR-.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2203.03466.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George E. Dahl, Christopher J. Shallue, and Roger Grosse. Which algorithmic choices matter at which batch sizes? Insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/e0eacd983971634327ae1819ea8b6214-Abstract.html.

Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean P. Foster, and Sham M. Kakade. How does critical batch size scale in pre-training? In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.21676.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham M. Kakade. Deconstructing what makes a good optimizer for autoregressive language models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.07972.

APPENDIX

Appendix roadmap.

- Section A discusses related work on batch-size scaling rules, the critical batch size, optimizer comparisons, batch-size-invariant optimizers, noisy quadratic models, and preconditioning under gradient noise.
- Section B fixes the optimizer notation and scaling coordinates, states the invariant-preservation and bound-minimization views of a scaling rule, and proves the results used in Section 3.
- Section C interleaves the noisy quadratic analysis and experiments: finite-budget risks and optimizer comparisons, finite-budget SignSGD batch-size transfer, and learning-rate scaling for momentum-SignSGD and RMSProp.
- Section D defines the matrix model and reports the additional results obtained with it.
- Section E gives the shared scaling-rule design, both empirical sweeps, the auxiliary weight-decay experiment, the comparison across settings, and the complete loss tables.
- Section F collects the supporting language-model optimizer-crossover results, optimizer definitions, the tuning protocol, final hyperparameters, and complete one-coordinate sweep tables.
- Section G gives the directional-branching protocol and numerical results, and establishes a local noisy-quadratic correspondence for Muon with a finite-horizon approximation bound.
- Section H measures how batch size changes the compute efficiency of each optimizer, including transfer across the token horizon and across batch size.

A RELATED WORK

Batch-size scaling rules. A scaling rule prescribes how hyperparameters tuned at one batch size should change at another. Goyal et al. (2017) popularized the linear learning-rate rule for SGD with an informal argument, which was later justified as preserving an SDE approximation of SGD (Smith & Le, 2018; Jastrzębski et al., 2017). Malladi et al. (2022) use SDE approximations to derive a square-root learning-rate rule for RMSprop and Adam that also rescales the moment coefficients and ϵ. Li et al. (2021) give a testable necessary condition for the SDE approximation of SGD, and hence its justification of the linear rule, to hold. Later work preserves narrower invariants: an asymptotic expected-loss bound together with the relative speeds of AdamW's moments and iterates (Monzio Compagnoni et al., 2025), the token half-life of Adam's second moment (Marek et al., 2025), or AdamW's weight-decay averaging timescale at fixed model size and training-data budget (Bergsma et al., 2025); the last two are identified empirically. A second line takes convergence bounds for LMO-based methods such as Muon (Kovalev, 2025; Pethick et al., 2025) and minimizes them over the learning rate and momentum separately at each batch size (Shulgin et al., 2026). Sato et al. (2025) instead minimize the sample complexity implied by a Muon convergence bound over the batch size and obtain a lower bound on the critical batch size that depends on momentum and weight decay. Their experiments set learning rates at other batch sizes by square-root scaling without testing whether this transfer is optimal. In a blog post, Cesista & Wen (2025) extend this critical-batch-size analysis to steepest descent under arbitrary norms with Nesterov momentum and decoupled weight decay. They also give a heuristic argument for square-root learning-rate scaling at a fixed number of steps, assuming the map from momentum to update direction is locally Lipschitz. Their language-model experiments, run at a fixed token budget rather than a fixed number of steps, find that the optimal learning rate grows roughly as the square root of batch size for AdamW and Muon. Related transfer results rescale hyperparameters with width through a reparameterization (Yang et al., 2021), compare width-scaling prescriptions under several parameterizations and optimizers (Everett et al., 2024), and rescale AdamW's learning rate, weight decay, ϵ, and moment coefficients with batch size and training duration (Mlodozeniec et al., 2026). Everett et al. (2024) compare a grid of width-scaling prescriptions at a fixed batch size. We derive batch-size scaling prescriptions from both SDE approximations and convergence bounds for the same optimizer, include them in grids of 216 and 648 complete rules, and compare the grids across two training settings. No tested rule is best in both settings, and the theoretically derived prescriptions disagree with each other.

Critical batch size. The critical batch size marks where increasing the batch size stops reducing the number of updates to a target loss proportionally: below it, doubling the batch roughly halves

the number of updates, and above it, returns diminish. McCandlish et al. (2018) predict it from a scalar gradient noise scale that weights gradient variance and the mean gradient by curvature; their simpler proxy is the total gradient variance divided by the squared gradient norm. Shallue et al. (2019) measure steps to a target across batch sizes and find that the end of the perfect-scaling region depends on the model, the optimizer, and to a lesser extent the dataset. Zhang et al. (2019) reproduce the optimizer dependence in experiments and in a noisy quadratic model: momentum and preconditioning speed up noise-free convergence but amplify the noise floor, so they extend perfect scaling to larger batches. Recent work fits how the critical batch size grows with training-data size in language-model pretraining, finding little dependence on model size (Zhang et al., 2025; Bergsma et al., 2025). For Adam-style updates, the optimal learning rate can first rise and then fall as batch size grows (Li et al., 2024). Merrill et al. (2025) measure it along a run by branching from checkpoints at larger batch sizes and warm the batch size up accordingly; Section 6 reuses that branch construction direction by direction. The critical-batch-size estimates in these works summarize training by a single scalar, even when the underlying model resolves individual directions. In our noisy quadratic analysis, directions with different curvature-to-noise ratios can favor different batch-size scaling rules. A single prescription can then preserve behavior in some directions while changing it in others, a distinction that a total-loss measurement alone does not resolve.

Comparing optimizers. Optimizer benchmarks establish rankings under a fixed protocol. Choi et al. (2019) show that a reported ranking can reverse when the hyperparameter search space changes, and Schmidt et al. (2021) find that no optimizer dominates across tasks, even at the largest of their tuning budgets. AlgoPerf (Dahl et al., 2023) standardizes the tuning budget and workload set but leaves the batch size to each submission. The Modded-NanoGPT optimization benchmark (Jordan, 2026) fixes the architecture, data, and batch size and ranks optimizers by the number of steps to a target loss. For language-model pretraining, Zhao et al. (2025) report that Adam, Adafactor, Lion, and Signum perform comparably once tuned while SGD lags, and Wen et al. (2026b) find that measured speedups over AdamW shrink with model size and depend on the tuning protocol. Qiu et al. (2025) report that Muon, SOAP, and Shampoo each keep a roughly $1.4\times$ gain in compute efficiency over AdamW from 190M to 1.4B parameters when the learning rate follows μP and weight decay scales inversely with width, at a fixed batch size. Their comparison measures forward and backward FLOPs needed to reach the same loss and excludes optimizer-update costs. Several studies vary batch size directly. Tuned SGD approaches Adam at small batch sizes (Marek et al., 2025; Srećković et al., 2025), and for 124M-parameter models, sign-based methods, MARS, and Prodigy catch up with AdamW as the batch grows from 16K to 262K tokens (Semenov et al., 2025). Muon needs fewer tokens than AdamW to reach a target loss, including beyond the critical batch size (Essential AI et al., 2025), and SOAP's advantage over both AdamW and Shampoo grows from 256K to 2M tokens per batch (Vyas et al., 2025). These studies show that optimizer comparisons depend on the tuning protocol, model scale, and batch size. We hold the model, data distribution, and token budget fixed and retune five optimizers independently at every batch size. The best optimizer changes: KL-SOAP (Lin et al., 2026), a variant of SOAP, leads at 128K tokens per batch and Shampoo (Gupta et al., 2018) leads at 2M, under both weight decay and HyperBall. This large-batch shift toward Shampoo runs opposite to the SOAP–Shampoo comparison of Vyas et al. (2025), which uses SOAP rather than KL-SOAP and a different model.

Batch-size-invariant optimizer design. Another response to batch size is to modify the optimizer. Wang & Aitchison (2024) propose Batch Size Invariant Adam, which modifies the second-moment update to average squared fixed-microbatch gradients instead of squaring the averaged minibatch gradient. In their small-update limit, linear scaling of the learning rate and of the moment refresh rates $1-\beta_1$ and $1-\beta_2$ yields the same first-order updates of the parameters and both moment estimates after processing the same microbatches. Johnson et al. (2020) instead adapt the SGD learning rate online from estimates of the gradient variance and norm. Hilton et al. (2022) make PPO-style policy optimization batch-size invariant by decoupling the proximal policy, which limits the update size, from the behavior policy used for importance sampling. These approaches modify the training algorithm. Our scaling analysis keeps the MuonW–AdamW hybrid fixed and rescales its hyperparameters across batch sizes, so Batch Size Invariant Adam is excluded from Table 1.

Noisy quadratic models. The noisy quadratic model represents training as a separable quadratic with a curvature and a gradient-noise variance per direction. Schaul et al. (2013) use it to derive a greedy per-coordinate optimal learning rate: the inverse curvature scaled by the fraction of the expected squared gradient that comes from its mean. Wu et al. (2018) use it to show that when

the problem is both noisy and ill-conditioned, greedily optimal learning-rate schedules decay too quickly and are suboptimal over longer horizons. Zhang et al. (2019) use it to predict how momentum, preconditioning, and iterate averaging change the perfect-scaling region, comparing optimizers across batch sizes by the number of steps to a target loss. For high-dimensional least-squares regression, Lee et al. (2022) identify a batch fraction above which minibatch heavy-ball momentum attains the full-batch accelerated rate and larger batches no longer help. Meterez et al. (2026) argue that quadratic models capture local optimization dynamics during language-model training. We fix the total sample budget, so batch size trades gradient noise against the number of remaining updates. We prove that this tradeoff reverses the ordering of two optimizers under independently optimal learning rate and momentum (Theorem 5.1).

Preconditioning and gradient noise. Adaptive optimizers such as Adam and SOAP build their preconditioners from second-moment statistics of minibatch gradients. Adam maintains an exponential moving average of elementwise squared gradients, while SOAP also averages $GG^\top$ and $G^\top G$ for matrix-valued gradients G. At a fixed parameter value, averaging B independent per-example gradient vectors with mean μ and covariance Σ gives a minibatch gradient g_B satisfying $\mathbb{E}[g_B g_B^\top] = \mu\mu^\top + \Sigma/B$. These second moments therefore contain both the mean-gradient contribution and a noise contribution that decreases with batch size; averaging the statistics over time does not remove that dependence. Morwani et al. (2025) study Shampoo's approximation to the Adagrad and Gauss–Newton matrices. When Shampoo's factors are built from real-label minibatch gradients, they find that its approximation to the Gauss–Newton matrix degrades as the batch size used to estimate the matrix increases. These statistics need not measure curvature: Kunstner et al. (2019) show that the empirical Fisher does not generally approximate the Fisher, and discuss gradient-noise adaptation as an alternative interpretation of empirical-Fisher preconditioning. Preconditioning can also hurt generalization. Wadia et al. (2021) show that whitening and certain unregularized, exact second-order methods can discard information useful for generalization in models with a fully connected, isotropically initialized first layer. Amari et al. (2021) decompose the generalization error of preconditioned gradient descent in overparameterized ridgeless regression into bias and variance. Natural gradient minimizes the variance term within their class of preconditioners, while the bias comparison depends on how the signal aligns with the features; the overall comparison therefore depends on label noise, misspecification, and signal alignment. Kunstner et al. (2023) vary batch size up to the full batch and conclude that minibatch noise is not the main cause of the gap between SGD and Adam on transformers.

B Derivations and comparison of batch-size scaling rules

This appendix derives the two Muon scaling candidates in Section 3 and compares them with previously published prescriptions. The first preserves a specified continuous-time model at matched data consumption. The second minimizes a stationarity bound at fixed data budget. Their different objectives lead to different learning-rate rules.

B.1 Setup and notation

We compare a reference batch size B with $B' = \kappa B$, where $\kappa > 0$, holding the model and data budget τ fixed. The corresponding number of updates is $n = \tau/B$. The budget and batch use the same unit: tokens for language modeling and examples for image classification. When comparing learning-rate schedules, we hold their shape fixed as a function of the fraction of the budget consumed.

We use the optimizer notation of Definition 1, omitting the parameter-group subscripts M and A when unambiguous.

Let θ denote a generic parameter vector, let ξ be a sample from the data distribution, and let $\ell(\theta;\xi)$ be its loss. Define $L(\theta) = \mathbb{E}_\xi[\ell(\theta;\xi)]$ and $g(\theta) = \nabla L(\theta)$. Index updates by $k \geq 1$, with initial momentum $m_1 = 0$. Conditional on the optimizer history $\mathcal{F}_k$ before step k, let z_k denote the scaled, zero-mean minibatch-gradient noise and $\Sigma(\theta_k)$ its covariance. Write the minibatch gradient as

$$\widehat{g}_B(\theta_k) = g(\theta_k) + B^{-1/2} z_k, \qquad \mathbb{E}[z_k \mid \mathcal{F}_k] = 0, \qquad \mathrm{Cov}(z_k \mid \mathcal{F}_k) = \Sigma(\theta_k). \tag{8}$$

For matrix parameters, covariance is defined after vectorization. Let $\mu \in (0,1)$ be the momentum retention coefficient, so its refresh rate is $1-\mu$. With Nesterov mixing weight $\omega \in [0,1]$ on the

updated momentum, direction map Ψ, learning rate $\eta > 0$, and weight decay $\lambda \geq 0$, the momentum state m_k, corrected input $\widetilde{m}_{k+1}$, and parameter state θ_k evolve as

$$m_{k+1} = \mu m_k + (1-\mu)\widehat{g}_B(\theta_k), \tag{9}$$
$$\widetilde{m}_{k+1} = \omega m_{k+1} + (1-\omega)\widehat{g}_B(\theta_k), \tag{10}$$
$$\theta_{k+1} = \theta_k - \eta\Psi(\widetilde{m}_{k+1}) - \eta\lambda\theta_k. \tag{11}$$

In this theoretical model, ω and μ are independent hyperparameters. Setting $\Psi = \mathrm{Orth}$ gives the Muon variant studied here. The experimental MuonW update in Definition 1 is the restricted case $\omega = \mu$. That restriction is not imposed on the theoretical batch-size transport. The generic recursion is also used in the bound calculation in Section B.3.

B.2 Matching the sample-time Muon diffusion

We state a sample-time diffusion and prove that the scaling rule below preserves its law. One step processes B samples, so the sample time before update k is $s_k = (k-1)B$. Define the parameter step per sample and exact exponential-moving-average (EMA) decay rate per sample by

$$\gamma_{\mathrm{samp}} = \frac{\eta}{B}, \qquad \rho = -\frac{\log\mu}{B}. \tag{12}$$

We will assume Ψ is a twice continuously differentiable direction map in real arithmetic,[2] with Jacobian $J(q) = D\Psi(q)$. Let $\sigma(\theta) = \Sigma(\theta)^{1/2}$ be a covariance square root and let W_s be standard Brownian motion in the vectorized parameter space. At the continuous-time state (θ_s, m_s), denote the mixture of momentum and the mean gradient by q_s. Consider the sample-time diffusion

$$q_s = \omega m_s + (1-\omega)g(\theta_s),$$
$$\mathrm{d}m_s = \rho(g(\theta_s) - m_s)\,\mathrm{d}s + \rho\sigma(\theta_s)\,\mathrm{d}W_s, \tag{13}$$
$$\mathrm{d}\theta_s = -\gamma_{\mathrm{samp}}(\Psi(q_s) + \lambda\theta_s)\,\mathrm{d}s - \gamma_{\mathrm{samp}}(1-\omega)J(q_s)\sigma(\theta_s)\,\mathrm{d}W_s. \tag{14}$$

Both equations use the same Brownian motion, reflecting the shared minibatch noise in the parameter and momentum updates. All coefficients are evaluated at the current state, so the model uses the Itô convention. Proposition B.1 proves exact invariance of this SDE, while Proposition B.2 bounds its error relative to the discrete recursion under additional assumptions.

Proposition B.1 (Coefficient matching with independent momentum coefficients)**.** *Fix the objective, noise covariance, direction map, and initial joint law of parameters and momentum. Suppose the coefficients of* (13)–(14) *are globally Lipschitz and have at most linear growth. For $B' = \kappa B$, the transformation*

$$\eta' = \kappa\eta, \qquad \mu' = \mu^\kappa, \qquad \omega' = \omega, \qquad \lambda' = \lambda \tag{15}$$

preserves the law of this diffusion at matched sample time.

Proof. At a joint state $x = (m, \theta)$, let $q = \omega m + (1-\omega)g(\theta)$. The joint drift b and diffusion coefficient A of (13)–(14) are

$$b(x) = \begin{pmatrix} \rho(g(\theta) - m) \\ -\gamma_{\mathrm{samp}}(\Psi(q) + \lambda\theta) \end{pmatrix}, \qquad A(x) = \begin{pmatrix} \rho\sigma(\theta) \\ -\gamma_{\mathrm{samp}}(1-\omega)J(q)\sigma(\theta) \end{pmatrix}.$$

Thus the joint process $X_s = (m_s, \theta_s)$ satisfies $\mathrm{d}X_s = b(X_s)\,\mathrm{d}s + A(X_s)\,\mathrm{d}W_s$. Write primes for the coefficients and process under the transformed hyperparameters. Since $B > 0$, $\kappa > 0$, and $\mu \in (0,1)$,

$$\gamma'_{\mathrm{samp}} = \frac{\eta'}{B'} = \frac{\kappa\eta}{\kappa B} = \gamma_{\mathrm{samp}}, \qquad \rho' = -\frac{\log\mu'}{B'} = -\frac{\log(\mu^\kappa)}{\kappa B} = -\frac{\log\mu}{B} = \rho.$$

Moreover, $\omega' = \omega$ and $\lambda' = \lambda$. The objective, direction map, and covariance square root are fixed, so at every state x we have $q' = q$, $b'(x) = b(x)$, and $A'(x) = A(x)$. In particular, the full joint covariance $A'(x)A'(x)^\top = A(x)A(x)^\top$ is unchanged.

For each deterministic initial state, the global Lipschitz and linear-growth assumptions give a unique global strong solution. Couple the reference and transformed equations using that same initial state

[2] Note that the idealized Muon instantiation doesn't have this property but the matrix sign operation is empirically simulated by a smooth matrix polynomial.

and the same Brownian motion. They then have identical coefficients and driving paths, so pathwise uniqueness gives $X'_s = X_s$ for all $s \geq 0$ almost surely. Their path laws therefore agree for every deterministic initial state. Integrating these laws against the common initial joint distribution proves equality of the path laws for the prescribed initialization, and hence at matched sample time. □

Approximation of the discrete updates. Let $U_k = (m_{k+1}, \theta_{k+1})$ be the joint discrete state after k updates of (9)–(11), and let $X_s = (m_s, \theta_s)$ solve (13)–(14). Write b, A for the joint drift and diffusion coefficients displayed in the preceding proof. For a sample horizon $\tau = nB$, define the coefficients in normalized time $t = s/\tau$ by $b_\tau = \tau b$ and $A_\tau = \sqrt{\tau} A$. For a random vector V, write $\|V\|_{L^2} = (\mathbb{E}\|V\|^2)^{1/2}$. Vector norms are Euclidean after vectorization, and $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm.

Proposition B.2 (Finite-horizon diffusion approximation)**.** *Fix $B > 0$, $\eta > 0$, $\mu \in (0, 1)$, $\omega \in [0, 1]$, $\lambda \geq 0$, and an integer $n \geq 1$. Let ξ_k be independent standard Gaussian vectors, independent of U_0, and assume the gradient noises satisfy $z_{k+1} = \sigma(\theta_{k+1})\xi_k$. Suppose $\|\sigma(\theta)\|_{\mathrm{F}} \leq S$ for all θ, and the operator norms of $D\Psi$ and $D^2\Psi$ are bounded by K and H on the entire parameter space. Assume that for constants $L, C \geq 0$ and all joint states x, y,*

$$\|b_\tau(x) - b_\tau(y)\|^2 + \|A_\tau(x) - A_\tau(y)\|_{\mathrm{F}}^2 \leq L^2\|x - y\|^2,$$
$$\|b_\tau(x)\|^2 + \|A_\tau(x)\|_{\mathrm{F}}^2 \leq C(1 + \|x\|^2).$$

Suppose $\mathbb{E}\|U_0\|^2 < \infty$, and let $Q < \infty$ bound the momentum lag over this horizon:

$$\max_{0 \leq k < n} \|g(\theta_{k+1}) - m_{k+1}\|_{L^2} \leq Q.$$

Define the momentum-coefficient discrepancy δ_B, the direct noise coefficient c, and the direction-error bound $\mathcal{R}_B$ by

$$\delta_B = \frac{1 - \mu}{B} - \rho, \qquad c = 1 - \omega\mu,$$
$$\mathcal{R}_B = K\omega(1 - \mu)\left(Q + \frac{S}{\sqrt{B}}\right) + \frac{\sqrt{3}Hc^2S^2}{2B}.$$

There is a finite constant $C_{\mathrm{E}} = C_{\mathrm{E}}(L, C, \mathbb{E}\|U_0\|^2)$ and a coupling with $X_0 = U_0$ such that

$$\max_{0 \leq k \leq n} \|U_k - X_{kB}\|_{L^2} \leq C_{\mathrm{E}}\sqrt{\frac{B}{\tau}} + e^{1+L+L^2}\sqrt{4\delta_B^2(\tau^2Q^2 + \tau S^2) + 2\tau^2\gamma_{\mathrm{samp}}^2\mathcal{R}_B^2}, \tag{16}$$

Moreover, $|\delta_B| \leq \rho^2 B/2$.

Proof. Set $h = B/\tau = 1/n$, and couple the Gaussian noises to Brownian increments by $\xi_k = (W_{(k+1)B} - W_{kB})/\sqrt{B}$. Then $\widetilde{W}_t = W_{\tau t}/\sqrt{\tau}$ is standard Brownian motion. Write $\Delta\widetilde{W}_k = \widetilde{W}_{(k+1)h} - \widetilde{W}_{kh}$, whose conditional covariance is hI.

At step $k + 1$, define the momentum lag and mean corrected input by $r_{k+1} = g(\theta_{k+1}) - m_{k+1}$ and $q_{k+1} = \omega m_{k+1} + (1 - \omega)g(\theta_{k+1})$. The momentum recursion gives exactly

$$\widetilde{m}_{k+2} = q_{k+1} + \omega(1 - \mu)r_{k+1} + cz_{k+1}/\sqrt{B}.$$

Define the direction remainder R_{k+1} by

$$\Psi(\widetilde{m}_{k+2}) = \Psi(q_{k+1}) + (1 - \omega)J(q_{k+1})z_{k+1}/\sqrt{B} + R_{k+1}.$$

Applying the derivative bound to the momentum shift, Taylor's theorem to the noise increment, and $c - (1 - \omega) = \omega(1 - \mu)$ yields

$$\|R_{k+1}\| \leq K\omega(1 - \mu)\left(\|r_{k+1}\| + \frac{\|z_{k+1}\|}{\sqrt{B}}\right) + \frac{Hc^2\|z_{k+1}\|^2}{2B}.$$

Conditional Gaussianity and $\|\sigma\|_{\mathrm{F}} \leq S$ give $\mathbb{E}[\|z_{k+1}\|^2 \mid \mathcal{F}_{k+1}] \leq S^2$ and $\mathbb{E}[\|z_{k+1}\|^4 \mid \mathcal{F}_{k+1}] \leq 3S^4$. Minkowski's inequality therefore gives $\|R_{k+1}\|_{L^2} \leq \mathcal{R}_B$.

Let f_k, D_k, and u_k denote the drift, diffusion, and direction defects relative to an Euler–Maruyama step:

$$f_k = \begin{pmatrix} \tau\delta_B r_{k+1} \\ 0 \end{pmatrix}, \quad D_k = \begin{pmatrix} \sqrt{\tau}\delta_B \sigma(\theta_{k+1}) \\ 0 \end{pmatrix}, \quad u_k = \begin{pmatrix} 0 \\ -\tau\gamma_{\text{samp}} R_{k+1} \end{pmatrix}.$$

Direct substitution into the discrete recursion gives

$$U_{k+1} = U_k + hb_\tau(U_k) + A_\tau(U_k)\Delta\widetilde{W}_k + hf_k + D_k\Delta\widetilde{W}_k + hu_k.$$

In particular, $\mathbb{E}\|f_k\|^2 \le \tau^2\delta_B^2 Q^2$, $\mathbb{E}\|D_k\|_{\text{F}}^2 \le \tau\delta_B^2 S^2$, and $\mathbb{E}\|u_k\|^2 \le \tau^2\gamma_{\text{samp}}^2\mathcal{R}_B^2$.

Let $Y_0 = U_0$ and $Y_{k+1} = Y_k + hb_\tau(Y_k) + A_\tau(Y_k)\Delta\widetilde{W}_k$. Define $e_k = U_k - Y_k$, $\Delta b_k = b_\tau(U_k) - b_\tau(Y_k)$, and $\Delta A_k = A_\tau(U_k) - A_\tau(Y_k)$. The increment before adding the direction defect is $F_k = e_k + h\Delta b_k + hf_k + (\Delta A_k + D_k)\Delta\widetilde{W}_k$. All its coefficients are measurable before the Brownian increment. Conditional expectation, the Lipschitz bound, and $h \le 1$ give

$$\begin{aligned}
\mathbb{E}\|F_k\|^2 &\le \big[(1+h)(1+2Lh) + 2L^2h\big]\mathbb{E}\|e_k\|^2 + h(1+h)\mathbb{E}\|f_k\|^2 + 2h\mathbb{E}\|D_k\|_{\text{F}}^2, \\
\mathbb{E}\|e_{k+1}\|^2 &\le (1+h)\mathbb{E}\|F_k\|^2 + h(1+h)\mathbb{E}\|u_k\|^2 \\
&\le e^{(2+2L+2L^2)h}\mathbb{E}\|e_k\|^2 + h\big(4\mathbb{E}\|f_k\|^2 + 4\mathbb{E}\|D_k\|_{\text{F}}^2 + 2\mathbb{E}\|u_k\|^2\big).
\end{aligned}$$

The second inequality is algebraic and does not require u_k to be measurable before the increment. Iterating from $e_0 = 0$ and using $nh = 1$ yields

$$\max_{k\le n}\mathbb{E}\|U_k - Y_k\|^2 \le e^{2+2L+2L^2}\big[4\delta_B^2(\tau^2Q^2 + \tau S^2) + 2\tau^2\gamma_{\text{samp}}^2\mathcal{R}_B^2\big].$$

For completeness, define the continuous Euler interpolation by

$$\overline{Y}_t = U_0 + \int_0^t b_\tau(Y_{\lfloor v/h\rfloor})\,\mathrm{d}v + \int_0^t A_\tau(Y_{\lfloor v/h\rfloor})\,\mathrm{d}\widetilde{W}_v, \qquad 0 \le t \le 1.$$

It satisfies $\overline{Y}_{kh} = Y_k$. The growth bound, Itô isometry, and Gronwall's inequality imply

$$\sup_{t\le 1}\mathbb{E}\|\overline{Y}_t\|^2 \le M, \qquad M = 3\big(\mathbb{E}\|U_0\|^2 + C\big)e^{3C},$$

and hence $\mathbb{E}\|\overline{Y}_t - Y_{\lfloor t/h\rfloor}\|^2 \le 2Ch(1+M)$. For $Z_t = X_{\tau t}$, the integral equations and the Lipschitz bound give

$$\mathbb{E}\|Z_t - \overline{Y}_t\|^2 \le 4L^2\int_0^t \mathbb{E}\|Z_v - \overline{Y}_v\|^2\,\mathrm{d}v + 8L^2C(1+M)h.$$

A further application of Gronwall's inequality gives $\sup_{t\le1}\mathbb{E}\|Z_t - \overline{Y}_t\|^2 \le C_{\text{E}}^2 h$, with $C_{\text{E}}^2 = 8L^2C(1+M)e^{4L^2}$. The triangle inequality at the grid points proves (16). Finally, $\mu = e^{-\rho B}$ gives

$$0 \le -\delta_B = \frac{1}{B}\int_0^{\rho B}(1 - e^{-u})\,\mathrm{d}u \le \frac{1}{B}\int_0^{\rho B} u\,\mathrm{d}u = \frac{\rho^2 B}{2}.$$

The proof is then compete. □

B.3 Minimizing a stochastic Frank–Wolfe bound

A second prescription follows from minimizing an upper bound on the Frank-Wolfe gap after a fixed data budget. We minimize this bound over the learning rate while holding the weight-decay coefficient $\lambda > 0$ fixed as batch size varies. We adapt the stochastic Frank–Wolfe analysis of Sfyraki & Wang (2025) to the recursion in Section B.1.

Let $\mathcal{C}$ be a compact convex set with diameter $D > 0$.

Assumption B.3 (Exact linear minimization)**.** Fix $\lambda > 0$. For every input z, the direction map Ψ satisfies

$$u(z) := -\frac{\Psi(z)}{\lambda} \in \arg\min_{v\in\mathcal{C}}\langle z, v\rangle.$$

When $z = g(\theta)$, the point $u(z)$ minimizes the first-order model $L(\theta) + \langle z, v - \theta\rangle$ over $v \in \mathcal{C}$. It therefore gives the best destination in $\mathcal{C}$ according to the local linear approximation of the loss. For a gradient or momentum estimate z, the same interpretation holds for the linear model defined by that estimate. Writing $\gamma = \eta\lambda$, the update with weight decay becomes

$$\theta_{k+1} = (1-\gamma)\theta_k + \gamma u(\widetilde{m}_{k+1}).$$

Thus the existing optimizer update moves a fraction γ of the way toward this destination, which is a stochastic Frank–Wolfe step. Initialization in $\mathcal{C}$ and $0 < \gamma \le 1$ keep all iterates in $\mathcal{C}$.

Interpretation for Muon. Consider one matrix parameter block and a fixed shape factor $a_{\text{shape}} > 0$. For a matrix input Z of rank r, write its compact singular value decomposition as $Z = U\,\mathrm{diag}(s_1, \dots, s_r)V^\top$, with $s_i > 0$. Replacing each positive singular value by one gives the polar factor $UV^\top$. The idealized Muon direction is $\Psi(Z) = a_{\text{shape}}UV^\top$; take $\Psi(0) = 0$. Let $\|\cdot\|_{\text{op}}$ denote the spectral norm and use the Frobenius inner product $\langle Z, X\rangle = \mathrm{tr}(Z^\top X)$. Choose

$$\mathcal{C} = \{X : \|X\|_{\text{op}} \le a_{\text{shape}}/\lambda\}, \qquad u(Z) = -\frac{a_{\text{shape}}}{\lambda}UV^\top.$$

The point $u(Z)$ belongs to $\mathcal{C}$, and every $X \in \mathcal{C}$ satisfies

$$\langle Z, X\rangle \ge -\frac{a_{\text{shape}}}{\lambda}\sum_{i=1}^{r} s_i = \langle Z, u(Z)\rangle,$$

so this direction satisfies Assumption B.3. Geometrically, the negative polar direction minimizes the linear model over a spectral-norm ball, and weight decay sets the ball's radius. For several matrix blocks, the same argument applies to the Cartesian product of their respective balls. Finite Newton–Schulz iterations and rounding only approximate this exact direction; the theorem below ignores that implementation error. For fixed matrix shapes and shape factors, the diameter is fixed when weight decay is fixed, but changes if weight decay is optimized. Define the Frank–Wolfe gap by

$$\mathcal{G}_{\mathcal{C}}(\theta) = \max_{u\in\mathcal{C}}\langle u - \theta, -g(\theta)\rangle. \tag{17}$$

For a convex objective this quantity bounds the constrained suboptimality. For a nonconvex objective it is the stationarity measure controlled below.

Theorem B.4 (Stochastic Frank–Wolfe bound)**.** *Suppose Assumption B.3 holds and L has $L_{\text{sm}} > 0$ Lipschitz gradient on $\mathcal{C}$. Conditional on the history, assume the minibatch gradient is unbiased and, for a scalar noise bound $\sigma \ge 0$, satisfies $\mathbb{E}[\|\widehat{g}_B(\theta_k) - g(\theta_k)\|^2 \mid \mathcal{F}_k] \le \sigma^2/B$. Run (9)–(11) with $\Psi(q) = -\lambda u(q)$, $\theta_1 \in \mathcal{C}$, $m_1 = 0$, $\lambda > 0$, fixed $\mu \in (0,1)$ and $\omega \in [0,1]$, and $0 < \gamma = \eta\lambda \le 1$. Let τ be the sample budget, assume $n = \tau/B$ is an integer, and choose a independently and uniformly from $\{1, \dots, n\}$. For $\Delta_1 = L(\theta_1) - \inf_{\theta\in\mathcal{C}} L(\theta)$,*

$$\begin{aligned}\mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_a) \le{}& \frac{B\Delta_1}{\tau\gamma} + L_{\text{sm}}D^2\gamma\left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right) + \frac{D\sigma}{\sqrt{B}} \\ &+ \frac{DB\omega}{(1-\mu)\tau}\left((1-\mu)\frac{\sigma}{\sqrt{B}} + \mu\|g(\theta_1)\|\right).\end{aligned} \tag{18}$$

Proof. The pre-update initialization $m_1 = 0$ corresponds to $M_0 = 0$ in Definition 1. The exact linear minimization oracle keeps every iterate in $\mathcal{C}$. The proof couples objective descent to the momentum tracking error, cancelling that error in a single potential.

Let $v_k = u(\widetilde{m}_{k+1})$ be the point selected by the optimizer, and let $v_k^\star = u(g(\theta_k))$ minimize the linear model defined by the true gradient. Both points belong to $\mathcal{C}$, and $\theta_{k+1} = \theta_k + \gamma(v_k - \theta_k)$. Convexity of $\mathcal{C}$ and $0 < \gamma \le 1$ therefore give

$$\theta_{k+1} \in \mathcal{C}, \qquad \|\theta_{k+1} - \theta_k\| = \gamma\|v_k - \theta_k\| \le \gamma D.$$

By Assumption B.3 and the definition of the Frank–Wolfe gap,

$$\langle \widetilde{m}_{k+1}, v_k - v_k^\star\rangle \le 0, \qquad \langle g(\theta_k), v_k^\star - \theta_k\rangle = -\mathcal{G}_{\mathcal{C}}(\theta_k).$$

Adding and subtracting the true-gradient minimizer gives

$$\begin{aligned}\langle g(\theta_k), v_k - \theta_k\rangle &= \langle g(\theta_k), v_k^\star - \theta_k\rangle + \langle g(\theta_k), v_k - v_k^\star\rangle \\ &= -\mathcal{G}_{\mathcal{C}}(\theta_k) + \langle \widetilde{m}_{k+1}, v_k - v_k^\star\rangle + \langle g(\theta_k) - \widetilde{m}_{k+1}, v_k - v_k^\star\rangle \\ &\le -\mathcal{G}_{\mathcal{C}}(\theta_k) + D\|\widetilde{m}_{k+1} - g(\theta_k)\|.\end{aligned}$$

Smoothness then yields

$$L(\theta_{k+1}) \le L(\theta_k) + \langle g(\theta_k), \theta_{k+1} - \theta_k \rangle + \frac{L_{\mathrm{sm}}}{2}\|\theta_{k+1} - \theta_k\|^2$$
$$\le L(\theta_k) - \gamma \mathcal{G}_{\mathcal{C}}(\theta_k) + \gamma D \|\widetilde{m}_{k+1} - g(\theta_k)\| + \frac{L_{\mathrm{sm}} D^2 \gamma^2}{2}.$$

Define the minibatch error $\varepsilon_k = \widehat{g}_B(\theta_k) - g(\theta_k)$ and the expected momentum error $e_k = \mathbb{E}\|m_{k+1} - g(\theta_k)\|$. The conditional second-moment bound and Cauchy–Schwarz imply

$$\mathbb{E}\|\varepsilon_k\| \le \mathbb{E}\left[\sqrt{\mathbb{E}[\|\varepsilon_k\|^2 \mid \mathcal{F}_k]}\right] \le \frac{\sigma}{\sqrt{B}}.$$

Using (10), we obtain

$$\widetilde{m}_{k+1} - g(\theta_k) = \omega(m_{k+1} - g(\theta_k)) + (1-\omega)\varepsilon_k,$$
$$\mathbb{E}\|\widetilde{m}_{k+1} - g(\theta_k)\| \le \omega e_k + (1-\omega)\frac{\sigma}{\sqrt{B}}.$$

Taking expectations in the descent inequality therefore gives

$$\mathbb{E}L(\theta_{k+1}) \le \mathbb{E}L(\theta_k) - \gamma \mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_k) + \gamma D \omega e_k + \gamma D(1-\omega)\frac{\sigma}{\sqrt{B}} + \frac{L_{\mathrm{sm}} D^2 \gamma^2}{2}. \tag{19}$$

For $k < n$, the momentum recursion also gives the exact decomposition

$$m_{k+2} - g(\theta_{k+1}) = \mu(m_{k+1} - g(\theta_k)) + \mu(g(\theta_k) - g(\theta_{k+1})) + (1-\mu)\varepsilon_{k+1}.$$

The triangle inequality and Lipschitz continuity of the gradient imply

$$e_{k+1} \le \mu e_k + \mu \mathbb{E}\|g(\theta_k) - g(\theta_{k+1})\| + (1-\mu)\mathbb{E}\|\varepsilon_{k+1}\|$$
$$\le \mu e_k + \mu L_{\mathrm{sm}} \mathbb{E}\|\theta_{k+1} - \theta_k\| + (1-\mu)\frac{\sigma}{\sqrt{B}}$$
$$\le \mu e_k + \mu\gamma L_{\mathrm{sm}} D + (1-\mu)\frac{\sigma}{\sqrt{B}}.$$

Define the potential $V_k = \mathbb{E}L(\theta_k) + \gamma D \omega e_k/(1-\mu)$. For $k < n$, adding the weighted momentum-error bound to (19) yields

$$V_{k+1} \le \mathbb{E}L(\theta_k) - \gamma \mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_k) + \gamma D\omega e_k + \gamma D(1-\omega)\frac{\sigma}{\sqrt{B}} + \frac{L_{\mathrm{sm}} D^2 \gamma^2}{2}$$
$$+ \frac{\gamma D \omega}{1-\mu}\left(\mu e_k + \mu\gamma L_{\mathrm{sm}} D + (1-\mu)\frac{\sigma}{\sqrt{B}}\right)$$
$$= V_k - \gamma \mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_k) + L_{\mathrm{sm}} D^2 \gamma^2 \left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right) + \frac{\gamma D \sigma}{\sqrt{B}}.$$

Here the coefficients of e_k and the noise combine according to

$$\gamma D\omega + \frac{\gamma D \omega \mu}{1-\mu} = \frac{\gamma D \omega}{1-\mu}, \qquad \gamma D(1-\omega) + \frac{\gamma D\omega}{1-\mu}(1-\mu) = \gamma D.$$

Rearranging and summing over $k = 1, \ldots, n-1$ gives

$$\gamma \sum_{k=1}^{n-1} \mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_k) \le \sum_{k=1}^{n-1}(V_k - V_{k+1}) + (n-1)L_{\mathrm{sm}} D^2 \gamma^2 \left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right) + \frac{(n-1)\gamma D \sigma}{\sqrt{B}}$$
$$= V_1 - V_n + (n-1)L_{\mathrm{sm}} D^2 \gamma^2 \left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right) + \frac{(n-1)\gamma D \sigma}{\sqrt{B}}.$$

For the final step, nonnegativity of e_n gives

$$\gamma D \omega e_n \le \frac{\gamma D \omega}{1-\mu} e_n = V_n - \mathbb{E}L(\theta_n).$$

Applying (19) at $k = n$ therefore yields

$$\begin{aligned}\mathbb{E}L(\theta_{n+1}) &\le V_n - \gamma\mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_n) + \gamma D(1-\omega)\frac{\sigma}{\sqrt{B}} + \frac{L_{\mathrm{sm}}D^2\gamma^2}{2} \\ &\le V_n - \gamma\mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_n) + \frac{\gamma D\sigma}{\sqrt{B}} + L_{\mathrm{sm}}D^2\gamma^2\left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right).\end{aligned}$$

Combining this with the preceding sum gives

$$\begin{aligned}\gamma\sum_{k=1}^{n}\mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_k) &\le V_1 - \mathbb{E}L(\theta_{n+1}) + nL_{\mathrm{sm}}D^2\gamma^2\left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right) + \frac{n\gamma D\sigma}{\sqrt{B}} \\ &\le \Delta_1 + \frac{\gamma D\omega}{1-\mu}e_1 + nL_{\mathrm{sm}}D^2\gamma^2\left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right) + \frac{n\gamma D\sigma}{\sqrt{B}},\end{aligned}$$

where the second inequality uses $\mathbb{E}L(\theta_{n+1}) \ge \inf_{\theta\in\mathcal{C}} L(\theta)$. The same argument applies when $n = 1$, with the preceding sum empty. Since $m_1 = 0$, the initial momentum error satisfies

$$\begin{aligned}e_1 &= \mathbb{E}\|(1-\mu)\widehat{g}_B(\theta_1) - g(\theta_1)\| \\ &= \mathbb{E}\|(1-\mu)\varepsilon_1 - \mu g(\theta_1)\| \\ &\le (1-\mu)\mathbb{E}\|\varepsilon_1\| + \mu\|g(\theta_1)\| \\ &\le (1-\mu)\frac{\sigma}{\sqrt{B}} + \mu\|g(\theta_1)\|.\end{aligned}$$

Finally, uniform independent sampling of a, division by $n\gamma$, and $n = \tau/B$ give

$$\begin{aligned}\mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_a) &= \frac{1}{n}\sum_{k=1}^{n}\mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_k) \\ &\le \frac{\Delta_1}{n\gamma} + \frac{D\omega e_1}{n(1-\mu)} + L_{\mathrm{sm}}D^2\gamma\left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right) + \frac{D\sigma}{\sqrt{B}} \\ &\le \frac{B\Delta_1}{\tau\gamma} + L_{\mathrm{sm}}D^2\gamma\left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right) + \frac{D\sigma}{\sqrt{B}} \\ &\quad + \frac{DB\omega}{(1-\mu)\tau}\left((1-\mu)\frac{\sigma}{\sqrt{B}} + \mu\|g(\theta_1)\|\right),\end{aligned}$$

which is (18). □

Only the first two terms in (18) depend on γ. To make the tradeoff described in Section 3.1 explicit, write $c_1 = \Delta_1$, $c_2 = L_{\mathrm{sm}}D^2[\frac{1}{2} + \omega\mu/(1-\mu)]$, and let r collect the noise and initialization terms. At batch size κB_0, the bound is then

$$\mathbb{E}\mathcal{G}_{\mathcal{C}}(\theta_a) \le \frac{c_1\kappa B_0}{\tau\gamma} + c_2\gamma + r(\kappa B_0). \tag{20}$$

For $c_1, c_2 > 0$, differentiation before imposing the step-size constraints gives $c_2\gamma^2 = c_1\kappa B_0/\tau$, hence

$$\gamma_\star(\kappa B_0) = \sqrt{\frac{c_1\kappa B_0}{\tau c_2}} = \sqrt{\kappa}\,\gamma_\star(B_0).$$

At fixed weight decay, $\gamma = \eta\lambda$ is proportional to learning rate, so the same scaling holds for the learning-rate minimizer. For $\Delta_1 > 0$ and $L_{\mathrm{sm}} > 0$, minimizing them subject to the theorem gives

$$\gamma_{\mathrm{cvx}}(B) = \min\left\{\sqrt{\frac{B\Delta_1}{\tau L_{\mathrm{sm}}D^2\left(\frac{1}{2} + \frac{\omega\mu}{1-\mu}\right)}}, 1\right\}. \tag{21}$$

Thus $\eta_{\mathrm{cvx}} = \gamma_{\mathrm{cvx}}/\lambda$ grows as $\sqrt{B}$ until the cap $\gamma \le 1$ becomes active. If $\Delta_1 = 0$, the bound is minimized in the limit $\gamma \downarrow 0$; the nontrivial square-root rule concerns $\Delta_1 > 0$. This calculation keeps the momentum parameters fixed. It gives a learning rate rule and does not imply that an optimizer state statistic is preserved and transferring a tuned reference learning rate by this rule does not assert that the reference rate minimizes the bound.

Method	MuonW			Adam / AdamW			
	η'	μ'	λ'	η'	β_1'	β_2'	λ'
Invariant preservation							
Adam SDE (Malladi et al., 2022)	–	–	–	$\sqrt{\kappa}\,\eta$	$1-\kappa(1-\beta_1)$	$1-\kappa(1-\beta_2)$	–
Completed transfer (Mlodozeniec et al., 2026)	–	–	–	$\sqrt{\kappa}\,\eta$	$1-\kappa(1-\beta_1)$	$1-\kappa(1-\beta_2)$	$\sqrt{\kappa}\,\lambda$
Second-moment memory (Marek et al., 2025)	–	–	–	–	β_1	β_2^κ	–
Power Lines (Bergsma et al., 2025)*	$\kappa^p\eta$	μ	$\kappa^{1-p}\lambda$	$\kappa^p\eta$	β_1	β_2	$\kappa^{1-p}\lambda$
Loss and relative speeds (Monzio Compagnoni et al., 2025)	–	–	–	$\sqrt{\kappa}\,\eta$	$1-\sqrt{\kappa}(1-\beta_1)$	$1-\sqrt{\kappa}(1-\beta_2)$	$\sqrt{\kappa}\,\lambda$
Heuristic noise preservation (Cesista & Wen, 2025)†	$\kappa\eta$	–	–	$\kappa\eta$	–	–	–
Our Muon diffusion (Proposition B.1)	$\kappa\eta$	μ^κ	λ	–	–	–	–
Bound minimization							
Shulgin et al. (2026)	$\sqrt{\kappa}\,\eta$	μ	–	–	–	–	–
Bu et al. (2026)	$\sqrt{\kappa}\,\eta$	–	–	$\sqrt{\kappa}\,\eta$	–	–	–
Schaipp et al. (2025)	–	–	–	$\sqrt{\kappa}\,\eta$	–	–	–
Our Frank–Wolfe bound (Section B.3)	$\sqrt{\kappa}\,\eta$	μ	λ	–	–	–	–
Stability							
Granziol et al. (2022)	–	–	–	$\sqrt{\kappa}\,\eta$	–	–	–
Empirical selections							
LLM: 216-rule selection	η	μ	$\kappa\lambda$	η	β_1	β_2^κ	λ
CIFAR-5M: 648-rule selection	$\sqrt{\kappa}\,\eta$	μ	λ	$\sqrt{\kappa}\,\eta$	β_1	β_2	λ

Table 2: **Finite-change batch-size prescriptions at $B' = \kappa B$.** Each cell gives the target value in terms of the reference value for its parameter group; dashes mean unspecified. Adam SDE and Completed transfer also scale $\epsilon' = \epsilon/\sqrt{\kappa}$. Our experimental SDE candidate instead uses $\beta_i' = \beta_i^\kappa$ and fixed ϵ. $*$: the Power Lines product constraint leaves p free; its favored choice is $p = 0$. Its MuonW entries are our extension with fixed momentum, while fixed Adam moments are an empirical choice adopted from the paper. Fixed β_1 in the second-moment row is also an empirical choice adopted from the paper. †: our fixed-token-budget conversion of the heuristic in Cesista & Wen (2025), assuming per-step update-noise variance is proportional to η^2/B with a fixed proportionality factor; the source's empirical square-root rule is distinct from this conversion. Our Muon diffusion theorem holds ω fixed independently of μ; using it for tied MuonW is an additional approximation. Our bound rule assumes an exact oracle and fixed ω, before the effective step-size cap is active. Shulgin et al. (2026) use a large-horizon bound at fixed momentum for an exact oracle without Nesterov or weight decay. Section B.4 details these conditions.

B.4 Comparison with the published prescriptions

The MuonW and Adam/W tables in the main text (Table 1) report finite-change formulas at $B' = \kappa B$, distinguishing published prescriptions from our conversions and empirical selections. A dash means that the cited criterion does not specify the coordinate. For a generic retention coefficient $q \in (0, 1)$, exact sample-memory preservation gives $q' = q^\kappa$, whereas linear refresh scaling gives $q' = 1 - \kappa(1 - q)$. The former preserves the half-life measured in processed samples, $B \log 2/(-\log q)$. The two formulas agree only to first order in $1 - q$, for fixed κ. For a fixed exponent r, the expansion $q^r = 1 - r(1 - q) + \frac{1}{2}r(r - 1)(1 - q)^2 + O((1 - q)^3)$ as $q \to 1$ gives the power forms displayed in Table 1, with $r = \kappa$ or $r = \sqrt{\kappa}$. More generally, for an exponent a, affine refresh scaling $q' = 1 - \kappa^a(1 - q)$ requires $0 < \kappa^a(1 - q) < 1$ to keep the target retention in $(0, 1)$. The Power Lines entries use a free learning-rate exponent p, with the associated product constraint derived below in (24). Table 2 uses the source formulas rather than replacing affine refresh rules by exact powers. We will discuss the related work in more detail below.

The Adam SDE prescription of Malladi et al. (2022) scales $\eta' = \sqrt{\kappa}\,\eta$, $\beta_i' = 1-\kappa(1-\beta_i)$ for $i = 1, 2$, and the denominator stabilizer $\epsilon' = \epsilon/\sqrt{\kappa}$. Completed Hyperparameter Transfer adds the AdamW rule $\lambda' = \sqrt{\kappa}\,\lambda$ at fixed model size and token budget (Mlodozeniec et al., 2026). These prescriptions match a continuous-time model; their use for discrete trajectories depends on the respective SDE approximation conditions. Our experimental SDE candidate instead uses the exact-decay variant $\beta_i' = \beta_i^\kappa$ and keeps ϵ fixed. Its moment transport agrees with the source prescription up to first order in the refresh rates and keeps the β_i positive, and it does not apply the prescribed ϵ transport. The tables omit ϵ.

Monzio Compagnoni et al. (2025) preserve an asymptotic-loss upper bound and the relative speeds of the parameter and moment states when the SDE is run for infinite continuous time. Their prescription is $\eta' = \sqrt{\kappa}\,\eta$, $\lambda' = \sqrt{\kappa}\,\lambda$, and $\beta_i' = 1 - \sqrt{\kappa}(1-\beta_i)$. The supporting bound assumes a strongly convex, smooth objective with optimum at zero and isotropic noise in a noise-dominated regime where squared gradient coordinates are $O(\eta)$. It therefore supplies a different, conditional invariant from the trajectory criterion above.

Cesista & Wen (2025, Section 5) propose preserving the accumulated update-noise variance for steepest descent under arbitrary norms. Their argument assumes a locally Lipschitz map from momentum to update direction, a heuristic second-moment noise estimate, and additive noise variance across steps. The assumed per-step update-noise variance is proportional to η^2/B, with the proportionality factor held fixed. For T optimizer steps, preserving the accumulated variance therefore holds $T\eta^2/B$ fixed, giving

$$\frac{\eta'}{\eta} = \sqrt{\frac{B'/B}{T'/T}}.$$

At fixed step count, this gives $\eta' = \sqrt{\kappa}\,\eta$. At fixed token budget $\tau = BT$, however, $B' = \kappa B$ implies $T' = T/\kappa$, so the same heuristic gives $\eta' = \kappa\eta$. The corresponding row in Table 2 is our fixed-budget conversion, not a separate theorem for AdamW or a joint prescription for momentum, moment coefficients, and weight decay. The blog's experiments instead support square-root learning-rate scaling for AdamW and Muon at fixed token budgets (Cesista & Wen, 2025, Section 6); that empirical prescription should be distinguished from the fixed-budget consequence of its heuristic invariant.

Marek et al. (2025) studies the batch size transfer in the prescribe $\beta_2' = \beta_2^{\kappa}$ to preserve second-moment memory in processed tokens. When batch size counts sequences, this conversion assumes fixed sequence length. They keep β_1 fixed in their experiments; that choice does not follow from the second-moment invariant. The rule does not specify a universal learning-rate or weight-decay scaling.

Power Lines constrains the learning-rate–weight-decay product through parameter averaging (Bergsma et al., 2025). Writing AdamW's adaptive direction as $u_t = \widehat{m}_t/(\sqrt{\widehat{v}_t} + \epsilon)$, with bias-corrected first and second moments $\widehat{m}_t, \widehat{v}_t$, its update for $0 < \eta\lambda < 1$ is

$$\theta_{t+1} = (1 - \eta\lambda)\theta_t + \eta\lambda(-u_t/\lambda). \tag{22}$$

This is an exponential moving average of rescaled updates with nominal timescale $1/(\eta\lambda)$ steps. Relative to the τ/B training steps, the timescale is

$$\tau_{\text{Power Lines}} = \frac{B}{\eta\lambda\tau}. \tag{23}$$

Preserving it at fixed model size and budget requires $\eta'\lambda' = \kappa\eta\lambda$. We express this product constraint as the family

$$\eta' = \kappa^p\eta, \qquad \lambda' = \kappa^{1-p}\lambda, \tag{24}$$

where p is a free exponent; Power Lines favors $p = 0$, or fixed learning rate and linearly scaled weight decay. The prescription uses peak learning rate with a common schedule shape. Its nominal timescale approximates discrete decay when $\eta\lambda$ is small; exact retention involves products of $1 - \eta_t\lambda$. The preferred timescale also drifts empirically at large batches and with tokens per model parameter. The same averaging identity applies to MuonW after substituting its direction for u_t. The MuonW table entries are our extension of this identity with fixed μ.

Optimization-bound prescriptions depend on the optimizer and budget assumptions used to derive them. At fixed momentum, the large-horizon analysis with an exact linear minimization oracle by Shulgin et al. (2026) gives $\eta \propto \sqrt{B}$ at fixed data budget. Its analyzed recursion has neither Nesterov mixing nor decoupled weight decay; holding momentum fixed is a condition of this comparison, not a derived momentum optimum. Kovalev (2025); Pethick et al. (2025) provide related normalized-gradient and linear-minimization-oracle analyses. For Bu et al. (2026); Schaipp et al. (2025), the table's dagger marks our conversion of an asymptotic peak-learning-rate rule $\eta_{\text{peak}} \propto n^{-1/2}$ using $n = \tau/B$. This yields $\eta_{\text{peak}} \propto \sqrt{B}$ only when the schedule shape and the coefficients in the bound or fitted loss model are independent of batch size. Both works prove bounds for convex SGD and test predictions with AdamW; Bu et al. (2026) also study Muon with normalized SGD on non-matrix

parameters. In contrast, Islamov et al. (2026) study joint batch-size, sequence-length, and token-budget scaling, including fixed momentum along a path where the token budget grows. That result does not supply an unconditional fixed-budget batch transport and is therefore discussed separately from the table.

The stability analysis of Granziol et al. (2022) bounds a different learning-rate quantity: the maximal stable learning rate. For Adam, small damping means a small additive denominator stabilizer ϵ. The square-root rule applies when this damping is small and directions near the edge of the Hessian spectral bulk dominate the stability restriction (Granziol et al., 2022, Section 11). Writing N_{data} for the dataset size, the batch dependence is through $B/(1 - B/N_{\mathrm{data}})$. The table's $\sqrt{\kappa}$ approximation additionally assumes $B, B' \ll N_{\mathrm{data}}$. This result neither specifies AdamW weight-decay transport nor minimizes the stationarity bound in Section B.3.

C Noisy Quadratic Model: Analysis and Experiments

This appendix pairs the noisy quadratic analyses with their numerical protocols and supporting figures. We begin with the exact finite-budget risks and the proof of Theorem 5.1, followed by finite-budget SignSGD batch-size transfer, then learning-rate scaling for momentum-SignSGD and RMSProp.

C.1 Finite-budget risks and optimizer comparisons

We now turn from transporting a fixed optimizer to choosing an optimizer at a given batch size, focusing on the noisy quadratic model of Zhang et al. (2019). A batch size determines the signal-to-noise ratio of the minibatch gradient in each direction of the landscape. At a fixed training horizon T, it also determines the number of optimizer updates through T/B. The natural comparison is therefore not whether one scaling rule preserves a metric, but which optimizer family achieves the smallest tuned finite-budget risk at that batch size. The diagonal noisy quadratic is deliberately minimal, but it already separates these two questions because each coordinate has its own curvature, noise level, and signal-to-noise ratio.

C.1.1 Setup, noise ratios, and tuned finite-budget risk

Noisy quadratic model. Let f be a diagonal quadratic objective with Hessian H, and let $h_i > 0$ be the curvature in coordinate i:

$$f(\theta) = \frac{1}{2}\theta^\top H\theta = \frac{1}{2}\sum_{i=1}^{d} h_i\theta_i^2, \tag{25}$$

Write $\widehat{g}_B(\theta_k)$ for the minibatch gradient query at batch size B, and write Σ for the single-example gradient-noise covariance. With $\sigma_B := B^{-1/2}$, the query has the form

$$\widehat{g}_B(\theta_k) = H\theta_k + \sigma_B z_k, \qquad \mathbb{E}[z_k] = 0, \qquad \mathrm{Cov}(z_k) = \Sigma. \tag{26}$$

Assume that H and Σ are simultaneously diagonalizable. In their common eigenbasis, write h_i and c_i for the diagonal entries:

$$H = \mathrm{diag}(h_1, \ldots, h_d), \qquad \Sigma = \mathrm{diag}(c_1, \ldots, c_d),$$

Writing $z_{k,i} = \sqrt{c_i}\zeta_{k,i}$, where $\zeta_{k,i}$ is standardized, the i-th minibatch gradient coordinate is therefore

$$[\widehat{g}_B(\theta_k)]_i = h_i\theta_{k,i} + \sqrt{\frac{c_i}{B}}\,\zeta_{k,i}, \qquad \mathbb{E}[\zeta_{k,i}] = 0, \qquad \mathrm{Var}(\zeta_{k,i}) = 1. \tag{27}$$

The coordinate minibatch-noise variance is c_i/B. Define the curvature-normalized noise ratio by

$$\chi_i := \frac{c_i}{h_i^2}. \tag{28}$$

For an initial point θ_0, the initial coordinate signal-to-noise ratio is

$$\mathrm{SNR}_{i,0}(B) := \frac{Bh_i^2\theta_{0,i}^2}{c_i}. \tag{29}$$

Increasing the batch size raises this initial signal-to-noise ratio, but it does so unevenly across coordinates when the ratios c_i/h_i^2 differ.

Tuned finite-budget risk.

Let T denote the fixed training horizon, measured in examples or tokens. At batch size B, define the update count as $S := T/B$. We derive the finite-step formulas for integer S and ignore floor/ceiling effects when evaluating them in batch-size sweeps. Let A be an optimizer family, such as SGD, SignSGD, RMSProp, or a stochastic Newton method, and let $\nu \in \Lambda_A$ denote its tunable hyperparameters. Write $\theta_S^{A,\nu}(B)$ for the iterate produced by running A at batch size B for S minibatch-gradient queries. The batch-B tuned risk of optimizer family A at horizon T is

$$\mathcal{R}_A^\star(B;T) := \inf_{\nu\in\Lambda_A} \mathbb{E}\Big[f\Big(\theta_S^{A,\nu}(B)\Big)\Big] = \inf_{\nu\in\Lambda_A} \frac{1}{2}\sum_{i=1}^{d} h_i \mathbb{E}\left[\Big(\theta_{S,i}^{A,\nu}(B)\Big)^2\right]. \tag{30}$$

Whenever the minimum is attained, write an optimal hyperparameter choice as

$$\nu_A^*(B;T) \in \arg\min_{\nu\in\Lambda_A} \mathbb{E}\Big[f\Big(\theta_S^{A,\nu}(B)\Big)\Big], \tag{31}$$

so that $\mathcal{R}_A^\star(B;T) = \mathbb{E}[f(\theta_S^{A,\nu_A^*(B;T)}(B))]$. The optimizer-choice problem is to compare $\mathcal{R}_A^\star(B;T)$ across optimizer families at the same (B,T). This criterion deliberately allows the hyperparameters to be retuned at each batch size.

C.1.2 Risk decomposition and crossover proof

We prove Theorem 5.1 for the initialization $\theta_0 \sim \mathcal{N}(0,\mathcal{I}_2)$ and independent Gaussian oracle draws specified in the main text. The two-dimensional construction is $H = \mathrm{diag}(1/(T+1),1)$ and $\Sigma = \mathrm{diag}(1/(4T),0)$. For a positive diagonal preconditioner $P = \mathrm{diag}(p_i)$, use

$$m_{t+1} = \mu m_t + P\widehat{g}_B(\theta_t), \qquad \theta_{t+1} = \theta_t - \eta m_{t+1}, \qquad m_0 = 0. \tag{32}$$

Here m_t is an unnormalized heavy-ball buffer, with unit coefficient on the preconditioned gradient. SGD uses $P = I$ and Newton uses $P = H^{-1}$.

To express this preconditioning as a change of coordinates, let $\widetilde{\theta} = P^{-1/2}\theta$ and $\widetilde{m} = P^{-1/2}m$. The transformed gradient oracle is $\widetilde{g}_B(\widetilde{\theta}) = P^{1/2}\widehat{g}_B(P^{1/2}\widetilde{\theta})$, so the transformed curvature and single-example noise variance in direction i are $\widetilde{h}_i = p_i h_i$ and $\widetilde{c}_i = p_i c_i$. For $c_i > 0$, the transformed curvature-to-noise ratio is therefore

$$\widetilde{\mathrm{CNR}}_i = \frac{\widetilde{h}_i^2}{\widetilde{c}_i} = p_i \frac{h_i^2}{c_i} = p_i\,\mathrm{CNR}_i. \tag{33}$$

Thus each preconditioned optimizer can be represented as ordinary momentum SGD on its corresponding transformed quadratic. The original-coordinate ratios remain unchanged. The initialization also transforms, with $\mathbb{E}[\widetilde{\theta}_{0,i}^2] = 1/p_i$ for the initialization above.

We will first derive the risk without momentum. We then bound Newton's noise response uniformly over learning rate and momentum, compare it with one feasible SGD configuration at $B = 1$, and show the comparison reverse at the one-update batch $B = T$.

We overload the optimizer-family subscript with P to denote preconditioned stochastic gradient descent (PSGD) (Li, 2018) with fixed preconditioner P and no momentum ($\mu = 0$):

$$\theta_{t+1} = \theta_t - \eta P\widehat{g}_B(\theta_t), \qquad \eta > 0.$$

Write $\mathcal{R}_P(B,S,\eta)$ for its expected loss after S updates at learning rate η. Tuning only the learning rate at fixed budget T gives

$$\mathcal{R}_P^\star(B;T) := \inf_{\eta>0} \mathcal{R}_P(B,S,\eta),$$

Lemma C.1 (Risk of diagonal preconditioning)**.** *For $P = \mathrm{diag}(p_i)$, $p_i > 0$, and $\mu = 0$, define $q_i = p_i h_i$, $x_i = \eta q_i$, $a_S(x) = (1-x)^{2S}$, and $v_S(x) = x^2\sum_{j=0}^{S-1}(1-x)^{2j}$. The expected loss after S updates is, for every $\eta > 0$,*

$$\mathcal{R}_P(B,S,\eta) = \frac{1}{2}\sum_i \left[h_i a_S(x_i) + \frac{c_i}{Bh_i} v_S(x_i)\right]. \tag{34}$$

In particular, at fixed budget T,

$$\mathcal{R}_P^\star(B;T) = \inf_{\eta>0} \frac{1}{2} \sum_i \left[h_i a_{T/B}(\eta q_i) + \frac{c_i}{B h_i} v_{T/B}(\eta q_i) \right]. \tag{35}$$

Proof. The coordinate recursion and its second moment are

$$\theta_{t+1,i} = (1-x_i)\theta_{t,i} - \eta p_i \sqrt{c_i/B}\, \zeta_{t,i},$$
$$\mathbb{E}[\theta_{S,i}^2] = (1-x_i)^{2S} + \eta^2 p_i^2 \frac{c_i}{B} \sum_{j=0}^{S-1} (1-x_i)^{2j}.$$

Independence removes the cross terms, and $\mathbb{E}[\theta_{0,i}^2] = 1$. Multiplying by $h_i/2$, using $\eta^2 p_i^2 = x_i^2/h_i^2$, and summing proves the risk formula. Taking the infimum over the stated tuning domain gives (35). □

The variance term has a useful geometric interpretation. Define the noise-response vector $\kappa_S(x) = x(1, 1-x, \dots, (1-x)^{S-1})^\top$. Its squared norm is $v_S(x)$, and its entries sum to $1-(1-x)^S$. Cauchy–Schwarz therefore gives

$$v_S(x) \geq \frac{[1-(1-x)^S]^2}{S}. \tag{36}$$

Reducing the initial bias requires a nonzero total response, which forces nonzero noise variance. The following proof extends this argument to runs with momentum.

Proof of Theorem 5.1. Set $h = 1/(T+1)$ and $c = 1/(4T)$, so the two coordinates have curvatures $(h, 1)$ and noise variances $(c, 0)$.

Small batch: $B = 1$**.** For Newton, every coordinate has effective curvature one. For any $\eta > 0$ and $0 \leq \mu < 1$, let b_j describe the response to a single noise perturbation in the preconditioned gradient: an additive error ε at update t contributes $-b_j\varepsilon$ to that parameter coordinate at time $t+1+j$. These noise-response coefficients are computed through the auxiliary recursion

$$r_{-1} = 0, \quad r_0 = 1, \quad r_{j+1} = (1+\mu-\eta) r_j - \mu r_{j-1}, \qquad b_j = \eta r_j.$$

In particular, $b_0 = \eta$ is the immediate response. Without gradient noise, the parameter after j updates is $\theta_{j,i} = p_j \theta_{0,i}$, where $p_j = r_j - \mu r_{j-1}$. Substitution into the recursion gives $p_j - p_{j+1} = b_j$, hence $\sum_{j=0}^{T-1} b_j = 1 - p_T$. Unrolling the Newton update gives

$$\theta_{T,i} = p_T \theta_{0,i} - \frac{\sqrt{c_i}}{h_i} \sum_{j=0}^{T-1} b_j \zeta_{T-1-j,i}.$$

The same coefficients apply to both coordinates. The independent noise draws contribute variance with weights b_j^2. Cauchy–Schwarz therefore gives

$$\begin{aligned} \mathcal{R}_{\text{newton+mom}}(1, T; \eta, \mu) &= \frac{1+h}{2} p_T^2 + \frac{c}{2h} \sum_{j=0}^{T-1} b_j^2 \\ &\geq \frac{1+h}{2} p_T^2 + \frac{c}{2hT} (1-p_T)^2 \\ &\geq \frac{(1+h)c/(hT)}{2(1+h+c/(hT))} = \frac{1}{8T} + o(T^{-1}). \end{aligned} \tag{37}$$

The last inequality minimizes a quadratic over all real p_T. The bound is uniform in (η, μ), so it applies to the tuned risk and to the $\mu = 0$ subfamily.

For SGD, choose $\eta = 1$, $\mu = 0$. The sharp-coordinate bias vanishes in one step, while the flat coordinate has effective step h. By (34),

$$\begin{aligned} \mathcal{R}_{\text{sgd}}(1, T; 1) &= \frac{h}{2}(1-h)^{2T} + \frac{c}{2(2-h)}[1-(1-h)^{2T}] \\ &= \frac{1+7e^{-2}}{16T} + o(T^{-1}). \end{aligned}$$

Since $(1 + 7e^{-2})/16 < 1/8$, this feasible SGD risk is strictly below the Newton lower bound for all sufficiently large T. Both SGD families contain this configuration, proving both small-batch inequalities.

Large batch: $B = T$. Only one update is available. Because $m_0 = 0$, the momentum coefficient has no effect on either method's iterate. Newton with $\eta = 1$ eliminates the initial bias, giving

$$\mathcal{R}^\star_{\text{newton}}(T;T) = \mathcal{R}^\star_{\text{newton+mom}}(T;T) \le \frac{c}{2hT} = \frac{1}{8T} + o(T^{-1}).$$

Dropping SGD's nonnegative noise term and optimizing its bias over all real learning rates gives the lower bound

$$\begin{aligned}\mathcal{R}^\star_{\text{sgd}}(T;T) &= \mathcal{R}^\star_{\text{sgd+mom}}(T;T)\\ &\ge \frac{1}{2}\min_{\eta\in\mathbb{R}}\{h(1-\eta h)^2 + (1-\eta)^2\}\\ &= \frac{h(1-h)^2}{2(1+h^3)} = \frac{1}{2T} + o(T^{-1}).\end{aligned}$$

The quadratic is minimized at $\eta = (1 + h^2)/(1 + h^3)$. Its leading coefficient exceeds Newton's, proving both large-batch inequalities for all sufficiently large T. □

For a dyadic budget $T = 2^k$ in the theorem's range, at least one adjacent pair of dyadic batches satisfies $\mathcal{R}^\star_{\text{sgd+mom}}(B;T) \le \mathcal{R}^\star_{\text{newton+mom}}(B;T)$ and $\mathcal{R}^\star_{\text{sgd+mom}}(2B;T) > \mathcal{R}^\star_{\text{newton+mom}}(2B;T)$. Choose $2B$ as the first dyadic batch where the difference is positive. The preceding batch has nonpositive difference, allowing an intermediate tie.

C.1.3 Exact risks for general initializations

The SGD and Newton learning-rate tuning experiments in Section C.1.4 use a fixed initialization θ_0. We give the corresponding exact expected losses below, using the functions a_S and v_S defined above. For SGD with learning rate η, coordinate i satisfies

$$\mathbb{E}[\theta^2_{S,i}] = a_S(\eta h_i)\theta^2_{0,i} + \frac{c_i}{Bh_i^2} v_S(\eta h_i). \tag{38}$$

For Newton, the same calculation gives

$$\mathbb{E}[\theta^2_{S,i}] = a_S(\eta)\theta^2_{0,i} + \frac{c_i}{Bh_i^2} v_S(\eta). \tag{39}$$

Consequently, if

$$A_0 = \sum_{i=1}^d h_i\theta^2_{0,i}, \qquad V_N(B) = \frac{1}{B}\sum_{i=1}^d \frac{c_i}{h_i},$$

then the exact Newton risk is

$$\mathcal{R}_{\text{newton}}(B,S;\eta) = \frac{1}{2}\left[A_0 a_S(\eta) + V_N(B) v_S(\eta)\right]. \tag{40}$$

C.1.4 Learning-rate scaling under a finite budget

Figure 6 shows how the tuned learning rate changes with batch size for SGD, Newton, SignSGD, and RMSProp at a fixed training budget T. We compare these tuned rates with the linear scaling rule anchored at $B = 2$.

For SGD and Newton, we use the exact finite-S expressions in (38) and (40), respectively. The SignSGD panel uses no momentum and no weight decay. In coordinate i, its scalar update is

$$\theta_{k+1,i} = \theta_{k,i} - \eta\,\text{sign}\Big(h_i\theta_{k,i} + \sqrt{c_i/B}\,\zeta_{k,i}\Big),$$

where the $\zeta_{k,i}$ are independent standard Gaussian draws. A deterministic finite-volume recursion evaluates the terminal risk $\frac{1}{2}\sum_i h_i\mathbb{E}[\theta^2_{S,i}]$, with $S = T/B$. For the RMSProp panel, we use the deterministic mean-Fisher closure

$$\bar r_{k+1,i} = \beta_2\bar r_{k,i} + (1-\beta_2)\left(h_i^2 M_{k,i} + \frac{c_i}{B}\right), \qquad M_{k,i} := \mathbb{E}[\theta^2_{k,i}],$$

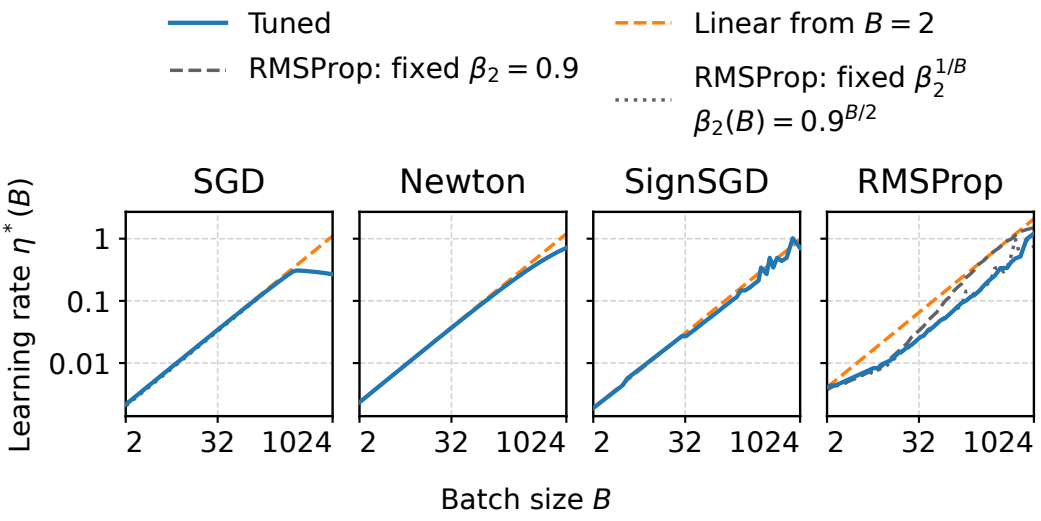

Figure 6: Tuned learning rates for SGD, Newton, SignSGD, and RMSProp (left to right) on the noisy quadratic with $h = (0.2, 1, 5)$, $c = (10, 1, 0.1)$, $\theta_0 = (0.3, 1, 3)$, and $T = 4096$. Blue curves show per-batch tuned learning rates; orange dashed curves show linear scaling anchored at $B = 2$. The dotted gray SGD curve is the large-T approximation. For RMSProp, the blue curve jointly tunes (η, β_2); the dashed gray curve fixes $\beta_2 = .9$, and the dotted gray curve keeps $\beta_2^{1/B}$ fixed using $\beta_2(B) = .9^{B/2}$. The RMSProp linear reference is anchored to the jointly tuned rate.

together with the corresponding linear second-moment update under the preconditioner $\bar{r}_{k+1,i}^{-1/2}$. We then tune RMSProp by a two-dimensional grid sweep,

$$\inf_{\substack{\eta>0 \\ 0\le\beta_2<1}} \mathcal{R}_{\mathrm{rms}}\left(B, \frac{T}{B}; \eta, \beta_2\right),$$

and plot two one-dimensional smoothing prescriptions as baselines: $\beta_2(B) \equiv .9$ and $\beta_2(B)^{1/B} = .9^{1/2}$, equivalently $\beta_2(B) = .9^{B/2}$. Even in these simplified diagnostics, the tuned steps do not remain on the linear reference anchored at $B = 2$: Newton bends smoothly through the horizon T/B, while SignSGD and RMSProp show additional bends from sign reliability and the empirical-Fisher feedback.

C.1.5 Optimizer crossover in the noisy quadratic model

We now compare optimizer families by their tuned terminal risk at a fixed training budget T. Their relative ranking changes with batch size, exhibiting optimizer crossover even after each family is retuned.

Figure 7 compares tuned SGD-M, Newton-M, SignSGD-M, and Adam at the baseline NQM noise level; Figure 4 gives the corresponding four-times-lower-noise variant. In both experiments, each family is retuned at every batch size over η and first-moment retention (μ for SGD, Newton, and SignSGD; β_1 for Adam), with Adam additionally tuning β_2. Momentum improves the small-batch SignSGD behavior.

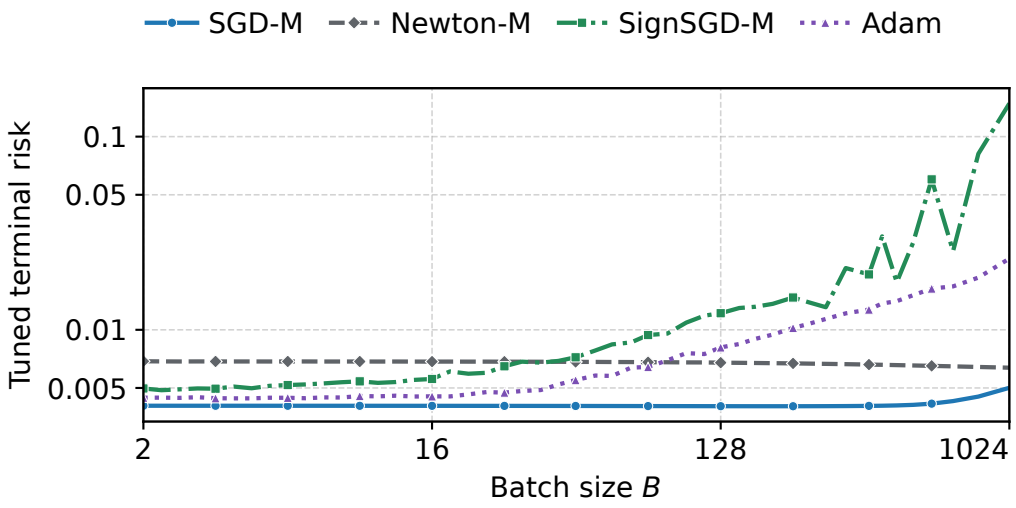

Figure 7: Tuned finite-budget risk versus batch size for momentum variants on the same three-coordinate noisy quadratic as Figure 6, with $T = 4096$. SGD-M, Newton-M, and SignSGD-M tune both η and μ at each batch size; Adam tunes η, β_1, and β_2. The plot uses a dense local refinement around the broad-sweep winners; the linear methods are evaluated by the exact second-moment recursion, and the SignSGD/Adam candidates are independently checked by high-chain Monte Carlo validation.

C.2 Finite-budget SignSGD batch-size transfer

A fixed batch-size scaling rule transports one optimizer family along a low-dimensional hyperparameter path. This is useful when the goal is to preserve a chosen invariant, but it is not the same as solving (30). In the noisy quadratic model, the target risk is a sum over coordinates whose effective tuples $(h_i, c_i, \theta_{0,i}, B, T/B)$ can demand different behavior. The SGD and Newton formulas (38)–(39) already show two different bias-variance tradeoffs; SignSGD depends on sign reliability; and RMSProp estimates an empirical Fisher whose mean depends on the stationary law of θ_i, not just on the explicit noise variance.

Equivalently, a scaling rule R from B_0 to B compares

$$\mathbb{E} f\left(\theta_{T/B}^{A,R_{B_0\to B}(\nu_0)}(B)\right)$$

against the tuned quantity $\mathcal{R}_A^\star(B;T)$. It is ideal only if the transported hyperparameters stay close to the optimizer-family optimum for every relevant noise ratio $\chi_i = c_i/h_i^2$ and signal-to-noise ratio $\mathrm{SNR}_{i,0}(B)$. The SDE and convex prescriptions can both miss this target.

C.2.1 Transfer from every reference batch against the per-batch optimum

Figure 3(a)–(c) uses a three-directional NQM with $H = \mathrm{diag}(1, 0.1, 0.01)$ and $\Sigma = \mathrm{diag}(0.01, 0.01, 0.01)$. Each trajectory starts from $\theta_0 \sim \mathcal{N}(0, I_3)$ and $m_0 = 0$. Momentum SignSGD uses $m_{t+1} = \mu m_t + (1-\mu)\widehat{g}_B(\theta_t)$ and $\theta_{t+1} = \theta_t - \eta\,\mathrm{sign}(m_{t+1})$, with no weight decay. Every run processes $T = 16384$ samples. We evaluate batches $B \in \{2^j : j = 0, \dots, 14\}$, giving exactly T/B updates. The largest batch is $B = T = 16384$, for which there is one update.

At each batch we tune shared hyperparameters by minimizing the sum of the three directional terminal losses. The coarse search uses 33 logarithmically spaced learning rates in $[10^{-5}, 2]$ and 17 logarithmically spaced refresh coefficients $1-\mu$ in $[10^{-5}, 1]$. An additional candidate is $(\eta, 1-\mu) = (0.01, 0.01)$, for 562 candidates in total. Each candidate is evaluated with 192 trajectories. A local search around six promising regions uses 768 fresh trajectories per candidate, followed by an independent sample of 4096 trajectories to select among the finalists. All candidates within a stage share initialization and oracle draws to reduce noise in their comparison; search seeds differ across batch sizes. Each selected recipe is the best configuration found by this search. At $B_0 = 1$, the

local search evaluates 318 distinct candidates and the final selection compares 29 finalists, returning $\eta_0 = 0.0004574766$ and $\mu_0 = 0.90503702$.

From each reference batch B_0 we transport its recipe to every target batch B by $\eta = (B/B_0)\eta_0$, $\mu = \mu_0^{B/B_0}$ for the SDE prescription and by $\eta = \sqrt{B/B_0}\,\eta_0$, $\mu = \mu_0$ for the convex one. The latter uses the square-root prescription motivated by the convex analysis. All transported recipes for one target batch, together with that batch's own selected recipe, are evaluated on shared initialization and oracle draws, using eight independent groups of 8192 trajectories with seeds separate from tuning and selection. Transporting a recipe to its own batch is the identity, so both rules reproduce the selected recipe exactly at $B = B_0$. The plotted quantity is the excess loss, the total terminal loss minus its selected value at $B = 1$, and the bands give one standard error across the eight groups.

C.3 Learning-rate scaling from the movement ansatz

C.3.1 Shared movement ansatz

For scalar parameter w, curvature $h > 0$, noise variance $c > 0$, and independent standard-normal z_t, the minibatch gradient at a fixed state w is $g_t = hw + \sqrt{c/B}\,z_t$. We evaluate both optimizer responses below at $w = 1$, keeping their decay coefficients fixed as batch size varies. Write an optimizer step as $w_{t+1} = w_t - \eta q_t$, where q_t is the update direction computed from the gradient and optimizer state. Holding w fixed while the optimizer state evolves, let $A(B) = \mathbb{E}[q_t]$ under its stationary law. The expected parameter displacement per processed sample is then $-\eta(B)A(B)/B$. For $A(B) \neq 0$, matching this displacement to its value at reference batch one gives the movement ansatz

$$\frac{\eta(B)}{\eta(1)} = \frac{BA(1)}{A(B)}. \tag{41}$$

This criterion matches mean movement at the frozen state, without imposing equality of update variance or terminal training loss. The two optimizers differ in their response $A(B)$, which we compute below.

C.3.2 Momentum-SignSGD response

With retention $\mu \in [0, 1)$ the moving average $m_t = \mu m_{t-1} + (1 - \mu)g_t$ is a linear function of Gaussian variables, so its stationary law is Gaussian with

$$\mathbb{E}[m] = hw, \qquad \mathrm{Var}(m) = \frac{1-\mu}{1+\mu} \cdot \frac{c}{B}. \tag{42}$$

For momentum SignSGD, $q_t = \mathrm{sign}(m_t)$, so the response is

$$A(B) = \mathbb{E}[\mathrm{sign}(m)] = \mathrm{erf}\left(\frac{hw}{\sqrt{2\mathrm{Var}(m)}}\right) = \mathrm{erf}\left(hw\sqrt{\frac{1+\mu}{1-\mu} \cdot \frac{B}{2c}}\right). \tag{43}$$

The Gaussian sign expectation gives this response in closed form.

Since $\mathrm{erf}(x) \approx 2x/\sqrt{\pi}$ for small x and $\mathrm{erf}(x) \to 1$ for large x, the response grows as $\sqrt{B}$ when noise dominates and saturates when signal dominates, so (41) interpolates between $\eta \propto \sqrt{B}$ and $\eta \propto B$. The noise-dominated limit requires $\mathrm{Var}(m) \gg h^2w^2$ at the largest batch, that is $\mathrm{CNR} \ll \frac{1-\mu}{1+\mu}B^{-1}$; at $B = 256$ this is $\mathrm{CNR} \ll 3.9 \times 10^{-3}$ for $\mu = 0$ but $\mathrm{CNR} \ll 2.1 \times 10^{-4}$ for $\mu = 0.9$. Momentum therefore does not change the two limits, only how much noise is needed to reach the square-root one.

Fitting (41) over $B = 1, \ldots, 256$ at $\mu = 0.9$ predicts exponents 0.591, 0.911, and 1.000 for $\mathrm{CNR} = 0.001$, 0.03, and 1, against measured 0.588, 0.794, and 0.904. The ansatz is accurate in the noise-dominated condition and overpredicts as CNR rises. This is expected: the ansatz freezes w at its initial scale, whereas a tuned trajectory spends much of the budget near the optimum, where hw is small and the sign is correspondingly more noise-driven than the frozen-state calculation assumes. The ansatz therefore bounds the measured exponents from the linear side rather than tracking them.

C.3.3 MOMENTUM-SIGNSGD LEARNING-RATE SCALING ACROSS CNR CONDITIONS

Model and tuning. Figure 3(d) uses $L(w) = w^2/2$ with $h = 1$ and $c \in \{1000, 100/3, 1\}$, giving $\mathrm{CNR} \in \{0.001, 0.03, 1\}$. The budget is $T = 4096$ samples and batches are $B = 2^j$, $j = 0, \ldots, 8$. At update t, the optimizer forms $g_t = w_{t-1} + \sqrt{c/B}\, z_t$ and $m_t = \mu m_{t-1} + (1-\mu) g_t$, then updates $w_t = w_{t-1} - \eta \operatorname{sign}(m_t)$, with $m_0 = 0$ and no weight decay. The main comparison uses $\mu = 0.9$; the no-momentum control uses $\mu = 0$.

The initialization is $w_0 \sim \mathcal{N}(0, 1)$, shared across learning-rate candidates and noise conditions. A fixed w_0 cannot be used here: because SignSGD moves a constant distance η per update, the iterate stays on the lattice $w_0 - k\eta$, and every η that divides w_0 lands exactly on the optimum. At $\mathrm{CNR} = 100$ and $B = 8$ this makes the tuning objective multimodal, with 26 local minima over the search range and terminal loss exactly zero at $\eta = 1/2$; a random w_0 leaves a single local minimum. This is a structural feature of constant-step methods rather than a search failure, and it does not arise for the normalized updates of Section C.3.5.

At each batch and noise condition, a coarse grid of 121 learning rates from 10^{-6} to 2 is evaluated on 512 trajectories. We refine three separated candidate neighborhoods and select a learning rate with 2048 fresh trajectories. All refined candidates are then evaluated on eight independent groups of 1024 held-out trajectories. Shared Gaussian draws reduce comparison noise across candidates and conditions. The largest ratio of a selected candidate's held-out loss to the minimum on its validation grid is 1.0516.

Selected learning rates. For the three CNR conditions in increasing order, the selected learning rate increases by factors of 27.2, 67.7, and 135.8 from $B = 1$ to $B = 256$ at $\mu = 0.9$. Fitting $\log \eta$ against $\log B$ over the nine batch sizes gives exponents 0.588, 0.794, and 0.904. The no-momentum control gives 0.505, 0.591, and 0.853 in the same conditions. Because the noise levels enter only through $\sqrt{c/B}$ applied to a shared standard normal draw, adding a noise condition leaves the others bitwise unchanged, which we verified across all cached cells.

Frozen-state check. The response $A(B)$ of Section C.3.2 is available in closed form because the moving average of Gaussian gradients is Gaussian. We verified it against a frozen-state simulation at $w = 1$ that discards 3000 updates and averages 3000 further updates over 2×10^5 trajectories; the largest absolute discrepancy is 2.0×10^{-4} at $\mu = 0.9$.

C.3.4 RMSPROP RESPONSE

Write β_2 for the squared-gradient accumulator's decay coefficient, v_t for that accumulator, and $\epsilon > 0$ for the numerical stabilizer. The frozen-state experiment evolves $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ while holding $w = 1$. Here $q_t = g_t/(\sqrt{v_t} + \epsilon)$, so the response is the stationary mean normalized gradient $A(B) = \mathbb{E}[g_t/(\sqrt{v_t} + \epsilon)]$.

The stationary accumulator has mean $h^2 w^2 + c/B$. Replacing the random accumulator by this mean and neglecting ϵ gives a mean-denominator approximation. At $w = 1$, with $\mathrm{CNR} = h^2/c$, it predicts

$$\frac{\eta(B)}{\eta(1)} \approx B \sqrt{\frac{\mathrm{CNR} + 1/B}{\mathrm{CNR} + 1}}. \tag{44}$$

Its local logarithmic slope is $1 - [2(1 + B\,\mathrm{CNR})]^{-1}$, approaching $1/2$ when noise dominates and 1 when signal dominates. In general, $\mathbb{E}[g/\sqrt{v}] \neq \mathbb{E}[g]/\sqrt{\mathbb{E}[v]}$. The dashed curves in Figure 8 therefore use measured stationary responses in (41), rather than assuming the mean-denominator approximation is exact. The conditional gradient signal-to-noise ratio along training is $Bw^2\,\mathrm{CNR}$, which changes even though CNR remains fixed.

C.3.5 RMSPROP LEARNING-RATE SCALING ACROSS CNR CONDITIONS

The same CNR dependence appears for adaptive optimizers. We repeat the experiment of Section 5.1 with RMSProp, which divides each gradient by the square root of a moving average of squared gradients. Its one-dimensional conditional second moment is $\mathbb{E}[\widehat{g}_B(w)^2 \mid w] = h^2 w^2 + c/B$: the signal term stays fixed at a given parameter value while the noise term falls with batch size. When noise dominates, a larger batch shrinks the normalization denominator through c/B, motivating

square-root scaling; when signal dominates, the denominator barely changes and linear scaling compensates for fewer updates per sample.

In Figure 8 the exponent rises from 0.48 at CNR $= 0.01$ through 0.75 at CNR $= 1$ to 1.02 at CNR $= 100$. Adam shows the same trend once its first moment is enabled, with the exponent rising from 0.47 to 0.94 at $\beta_2 = 0.9$ and from 0.48 to 0.95 at $\beta_2 = 0.99$ across the same three conditions (Figure 10). Tuned SGD stays between 0.96 and 0.99 in all three, showing that the dependence comes from the adaptiveness introduced by the second-moment normalization.

Training and independent tuning. Figure 8 uses $L(w) = w^2/2$, $w_0 = 1$, and $c \in \{100, 1, 0.01\}$, giving CNR $\in \{0.01, 1, 100\}$. The budget is $T = 4096$ samples and batches are $B = 2^j$, $j = 0, \ldots, 8$. At update t, RMSProp forms $g_t = w_{t-1} + \sqrt{c/B}\, z_t$ and $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$, then updates $w_t = w_{t-1} - \eta g_t/(\sqrt{v_t} + \epsilon)$. Here $v_0 = 0$, $\epsilon = 10^{-8}$, and the main comparison uses $\beta_2 = 0.9$ without bias correction, first-moment momentum, or weight decay. The controls divide v_t by $1 - \beta_2^t$ before normalization, with $\beta_2 = 0.9$ or 0.99.

At each batch and noise condition, a coarse grid of 101 learning rates from 10^{-5} to 1 is evaluated on 512 trajectories. We refine three separated candidate neighborhoods and select a learning rate with 2048 fresh trajectories. All refined candidates are then evaluated on eight independent groups of 1024 held-out trajectories. Shared Gaussian draws reduce comparison noise across candidates and conditions; trajectory indices remain independent within each candidate. The largest ratio of a selected candidate's held-out loss to the minimum on its validation grid is 1.00838.

For the three CNR conditions in increasing order, the selected learning rate increases by factors of 14.09, 63.26, and 281.84 from $B = 1$ to $B = 256$. The corresponding endpoint slopes $\log[\eta(256)/\eta(1)]/\log 256$ are 0.477, 0.748, and 1.017. Candidates within 5% of the minimum held-out loss at each endpoint give slope ranges $[0.422, 0.518]$, $[0.720, 0.768]$, and $[1.004, 1.031]$. These are descriptive loss-tolerance ranges conditional on the searched grid, rather than confidence intervals. With bias correction, the slopes are $(0.533, 0.831, 1.100)$ at $\beta_2 = 0.9$ and $(0.554, 0.865, 1.128)$ at $\beta_2 = 0.99$. The SGD control selects the minimum exact expected terminal loss on a dense learning-rate grid and gives slopes $(0.986, 0.974, 0.962)$.

Frozen-state response measurement. The RMSProp response in Section C.3.4 is measured at $w = 1$ for both decay coefficients. We initialize the accumulator at its fixed-state mean, discard 2048 updates, and average 512 further updates in each of eight independent groups of 512 antithetic pairs. The state remains fixed while the squared-gradient accumulator evolves. Uncertainty is assessed across the eight groups, accounting for dependence between successive updates within a trajectory. The measured learning-rate ratios differ from the mean-denominator approximation by at most 4.31% across tested settings; their largest relative group-based standard error is 0.023%. An independent scalar implementation matches the vectorized training evaluator to absolute error below 2×10^{-16}. In a fresh 1024-trajectory check of the main selected configurations, the stabilizer is at most 0.017% of the normalization denominator.

Adam. Figure 10 repeats the sweep with Adam, adding a first moment $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ at $\beta_1 = 0.9$ and dividing both moments by their bias-correction factors before the update. The model, budget, batch grid, search grid, trajectory counts, and selection protocol are unchanged,

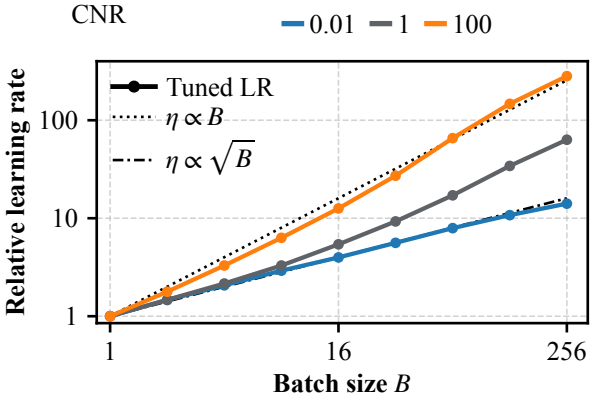

Figure 8: **CNR changes RMSProp learning-rate scaling.** Colors indicate CNR. Curves show learning rates tuned for terminal loss, each relative to its own value at $B = 1$. Dotted and dash-dotted lines mark linear and square-root scaling.

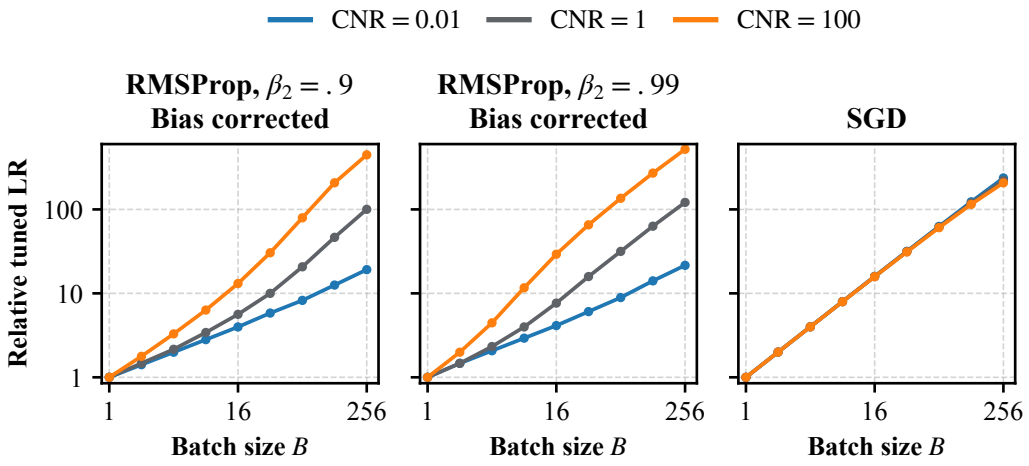

Figure 9: **The dependence on the curvature-to-noise ratio (CNR) survives accumulator bias correction.** The two RMSProp panels divide the squared-gradient exponential moving average by $1 - \beta_2^t$ at update t, with second-moment decay 0.9 and 0.99, respectively. The SGD panel selects learning rates using the exact expected terminal loss. All panels use the same one-dimensional NQM, initial point, sample budget, noise variances, and batch sizes as Figure 8. Learning rates are normalized separately at $B = 1$. SGD's relative learning-rate curves remain close to linear in batch size in these conditions, while RMSProp's curves separate with CNR.

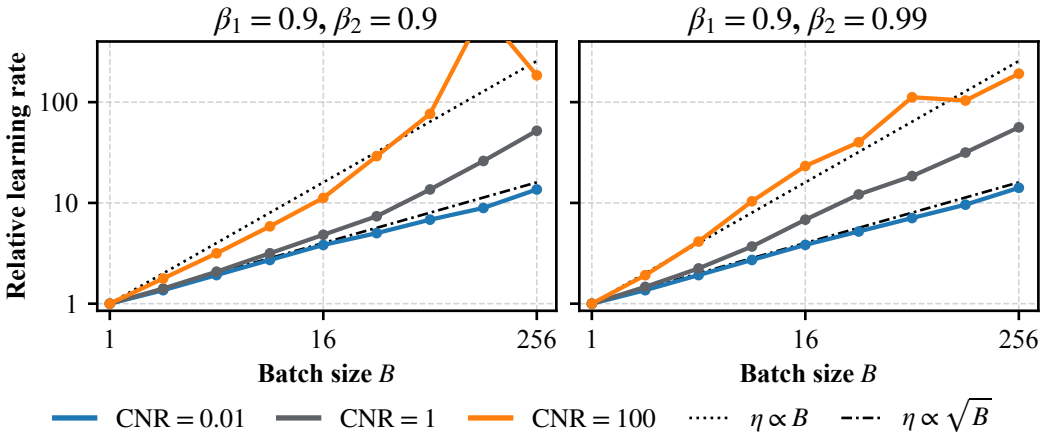

Figure 10: **Adam shows the same CNR dependence as RMSProp.** The tuning protocol, model, budget, and batch grid match Figure 8, with Adam's first moment enabled and bias correction applied to both moments. Colors indicate CNR; curves show learning rates tuned for terminal loss, each relative to its own value at $B = 1$. Dotted and dash-dotted lines mark linear and square-root scaling.

and search seeds are drawn separately from the RMSProp sweep. For the three CNR conditions in increasing order, the endpoint slopes are $(0.471, 0.713, 0.941)$ at $\beta_2 = 0.9$ and $(0.478, 0.727, 0.948)$ at $\beta_2 = 0.99$.

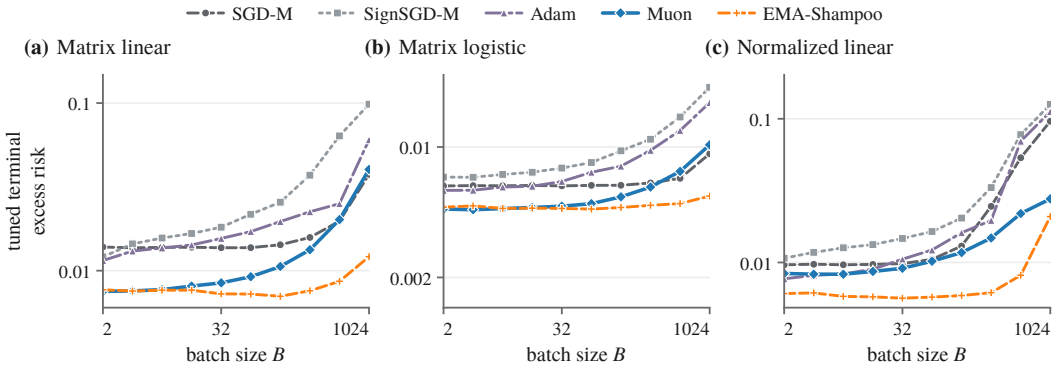

Figure 11: Tuned finite-budget terminal excess risk in the 4×4 matrix prediction models. Every point uses $T = 4096$ examples and T/B updates; each optimizer is retuned separately at every batch size. Minibatch sufficient statistics are sampled exactly from the finite-support design, while the expectation over training trajectories is estimated by independent float64 validation. Shaded bands show 95% Monte Carlo intervals for the exact population excess risk.

D Additional Matrix-Model Results

The main text shows optimizer crossover on a noisy quadratic (Figure 4). Crossovers also occur on three matrix regression tasks: at the higher noise level, the best optimizer changes from Muon at small batches to EMA-Shampoo at large batches in matrix linear and logistic regression (Figure 12). The following figures report all tuned optimizer families on all three matrix tasks at the baseline and higher noise levels.

D.1 Matrix-model definitions

Matrix linear regression. Let $X = ab^\top$ be a rank-one matrix covariate, where a and b are independent, centered random vectors satisfying $\mathbb{E}[aa^\top] = \mathsf{A}$ and $\mathbb{E}[bb^\top] = \mathsf{C}$. Under the linear teacher $y = \langle \Theta_\star, X \rangle + \epsilon$ and squared loss,

$$\ell_{\mathrm{lin}}(\Theta; X, y) = \frac{1}{2}\big(\langle \Theta, X \rangle - y\big)^2 \tag{45}$$

with $\mathbb{E}[\epsilon \mid X] = 0$. This model preserves a quadratic population objective but exposes matrix-aware optimizers to left–right feature geometry and rank-deficient per-example gradients to better simulate the neural network landscape.

Matrix logistic regression. Using the same rank-one covariates, define $\varsigma(u) = (1 + e^{-u})^{-1}$, a temperature-controlled teacher $p_\tau(X) = \varsigma(\langle \Theta_\star, X \rangle / \tau)$, and a student $p_\Theta(X) = \varsigma(\langle \Theta, X \rangle)$. The labels and population excess risk are

$$y \mid X \sim \mathrm{Bernoulli}(p_\tau(X)), \qquad L_{\log}(\Theta) - L^\star_{\log} = \mathbb{E}_X[D_{\mathrm{KL}}(\mathrm{Bern}(p_\tau(X)) \,\|\, \mathrm{Bern}(p_\Theta(X)))]. \tag{46}$$

Unlike the preceding objectives, its curvature depends on the current predictive probabilities and is therefore state dependent.

Normalized linear regression. Finally, let the predictor depend only on the direction of the matrix parameter through $Q(\Theta) = \Theta / \|\Theta\|_{\mathrm{F}}$ and $Q_\star = \Theta_\star / \|\Theta_\star\|_{\mathrm{F}}$. With the same rank-one design and a linear teacher,

$$y = \langle Q_\star, X \rangle + \epsilon, \qquad L_{\mathrm{nlin}}(\Theta) - L^\star_{\mathrm{nlin}} = \frac{1}{2}\left\| \mathsf{A}^{1/2}\big(Q(\Theta) - Q_\star\big)\mathsf{C}^{1/2} \right\|_{\mathrm{F}}^2. \tag{47}$$

The resulting objective is invariant to positive rescaling of Θ, introducing a radial null direction and optimizer-dependent angular dynamics.

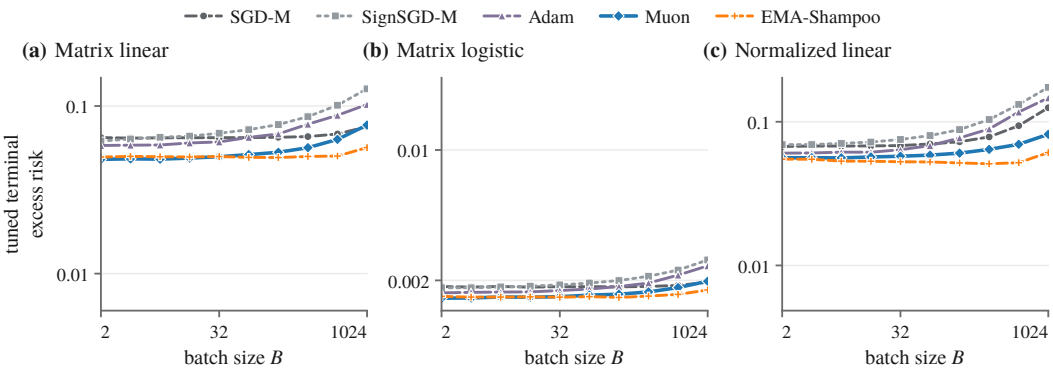

Figure 12: Higher-noise counterpart of Figure 11. The squared-loss models use $\sigma_\epsilon = 4$, and the logistic teacher uses $\tau_{\log} = 4$. The remaining settings and protocols, including the 95% Monte Carlo intervals, are unchanged.

E SCALING-RULE SWEEPS ON CIFAR-5M AND LANGUAGE MODELING

We test how well a scaling rule transfers a tuned hyperparameter configuration across batch sizes at a fixed data budget. The CIFAR-5M and language-model sweeps favor different rules, and the best rule also changes across target batches within each setting. We first define the common comparison, then report each sweep, the additional language-model weight-decay experiment, and the comparison across settings. The complete loss tables follow the results.

E.1 SHARED DESIGN AND EVALUATION

Scaling coordinates. Both experiments apply MuonW to hidden weight matrices and AdamW to the remaining parameters, which we call auxiliary parameters. We write their learning rates as η_{M} and η_{A}, their weight decays as λ_{M} and λ_{A}, Muon momentum as μ, and Adam's first- and second-moment retention coefficients as β_1 and β_2. The MuonW update is that of Definition 1, with $\omega = \mu$. In language modeling, embeddings, the output projection, and scalar parameters have distinct auxiliary learning rates; one scaling choice multiplies all three by the same factor. Let B_0 be the reference batch, B a target batch, and $\kappa = B/B_0$ their ratio. For each learning-rate or weight-decay coordinate h, with reference value h_0, the choices are

$$h(B) = h_0\kappa^p, \qquad p \in \{0, \tfrac{1}{2}, 1\},$$

corresponding to fixed, square-root, and linear scaling. For a retention coefficient $q \in \{\mu, \beta_1, \beta_2\}$, with reference value q_0, we either keep $q(B) = q_0$ or preserve its decay in data units. On CIFAR-5M, the latter choice is $q(B) = q_0^{\kappa}$, preserving the half-life in processed examples. For language modeling, let τ be the token budget and $S(B) = \lceil \tau/B \rceil$ the number of updates. We use $q(B) = q_0^{S(B_0)/S(B)}$, so $q(B)^{S(B)} = q_0^{S(B_0)}$ preserves retention over the complete run. The two exponents agree when the step-count ratio equals the batch ratio; Section E.3 specifies the rounding exception.

Complete grids. CIFAR-5M varies both learning rates and both weight decays independently, giving $3^4 2^3 = 648$ rules. The language-model grid holds auxiliary weight decay at 0.001 and varies the other six coordinates, giving $3^3 2^3 = 216$ rules. Rule selection and the comparisons below use the complete grid in each setting, with auxiliary weight decay varied independently on CIFAR-5M. In tables, LR and WD denote learning rate and weight decay; F, S, and L denote fixed, square-root, and linear scaling. E, also labeled EMA in the figures and tables, denotes the retention-preserving choice defined above for the relevant experiment. Subscripts mat and aux in the tables correspond to M and A.

Evaluation and rule selection. Let $\mathcal{B}$ be the set of target batches and $L_r(B)$ the validation loss of rule r at batch B, using the metric specified for each experiment below. Write $L_{\min}(B) = \min_{r'} L_{r'}(B)$ for the smallest loss in the tested grid. For these scaling-rule sweeps, the batch-size (BSZ) regret is the validation-loss gap to the best tested rule at the same batch size. We reserve excess loss for the loss gap to the retuned $B = 1$ reference in Figure 3(a)–(c). For each rule, we report its mean and maximum BSZ regret over the target batch-size choices,

$$\Delta_r(B) = L_r(B) - L_{\min}(B), \qquad \overline{\Delta}(r) = \frac{1}{|\mathcal{B}|}\sum_{B\in\mathcal{B}} \Delta_r(B), \qquad \Delta_{\max}(r) = \max_{B\in\mathcal{B}} \Delta_r(B).$$

We select a common rule by the smallest mean BSZ regret, weighting target batch-size choices equally. Ranking by mean loss gives the same order because all rules share the same target batches. The coordinate panels fix one coordinate choice and average loss over every combination of the remaining choices.

E.2 CIFAR-5M SWEEP

Protocol and reference configuration. We train a ResNet with GroupNorm in one pass over CIFAR-5M (Nakkiran et al., 2021). GroupNorm prevents changing batch size from changing the normalization statistics compared to BatchNorm. Every run processes the same number of examples with constant learning rates; schedule quantities are expressed as fractions of processed examples. The reference batch is $B_0 = 256$, and the targets are $B \in \{64, 128, 512, 1024, 2048, 4096\}$. Validation loss is per-example cross-entropy averaged over the final ten evaluations, equally spaced in processed

examples. We tune the hyperparameters at the reference batch size by coordinate descent with $\lambda_M = \lambda_A$, obtaining the reference configuration

$$(\eta_{\mathrm{M}}, \eta_{\mathrm{A}}, \mu, \beta_1, \beta_2, \lambda_{\mathrm{M}}, \lambda_{\mathrm{A}})_0 = (0.00681567,\ 0.000980392,\ 0.93002885,\ 0.6561,\ 0.999911571,\ 0.00125,\ 0.00125),$$

and its validation loss is 0.2314310. We evaluate all 648 rules at all six targets using the same code, data stream, and schedule, giving $3{,}888$ runs with seed 42.

Square-root learning rates with the other coordinates fixed give the best common rule. This rule minimizes mean BSZ regret as defined in Section E.1. Its mean and maximum BSZ regrets are 0.00492 and 0.00913. At $B = (64, 128, 512, 1024, 2048, 4096)$, its losses are

$$(0.23370,\ 0.23136,\ 0.23373,\ 0.22778,\ 0.23290,\ 0.23683).$$

Figure 13 shows the mean loss across rules with each coordinate choice. The individual winners still change with batch size. At $B = 64$, the best rule scales both learning rates and weight decays linearly, transports Muon momentum and Adam β_2, and keeps β_1 fixed. At $B = 4096$, it scales the Muon learning rate and auxiliary weight decay by the square root, transports β_2, and keeps the remaining coordinates fixed. Table 8 gives the complete grid.

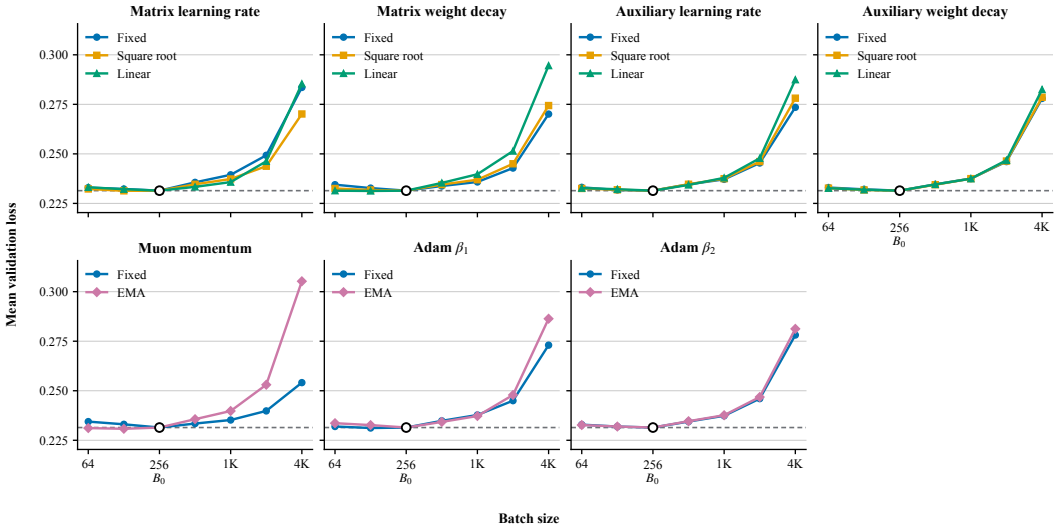

Figure 13: **All** 648 **CIFAR-5M scaling rules, summarized one coordinate at a time.** In each panel, a point at a target batch is the arithmetic mean of the final-window validation losses over all rules that make the indicated choice for that coordinate, weighting all combinations of the other six equally. The outlined marker and dashed line mark the shared reference at $B_0 = 256$. Fixed, square root, and linear denote exponents 0, $\frac{1}{2}$, and 1; EMA preserves the half-life in examples. Lower is better; all panels share the same vertical range.

Independent retuning provides a comparison beyond the scaling grid. For the full-retuning curve in Figure 2, we tune all seven hyperparameters independently at each target from 512 to 4096. The coordinate-descent procedure, detailed in Section F.3, starts from the best grid configuration at that batch. Learning rates and weight decays move by factors $\{\frac{1}{2}, \frac{1}{\sqrt{2}}, \sqrt{2}, 2\}$; retention coefficients use the same factors on $1 - q$. A candidate improving loss by more than 0.002 on seed 42 is repeated on seed 43, and also on seed 44 when the gain is below 0.006. We accept it only if its median improvement exceeds 0.002, and stop after all seven coordinates pass without an accepted move. Table 3 reports the selected configurations. The comparison curves add seeds 43 and 44 to the grid's seed 42 and report three-seed medians. At retuned configurations, the three losses span up to 0.0064, so a single-seed grid minimum can lie below the repeated-run result for the same configuration. Standard deviations over the final evaluation window describe variation during one run; they are not confidence intervals across seeds.

B	η_{M}	η_{A}	μ	β_1	β_2	λ_{M}	λ_{A}	median loss
256	0.006816	0.0009804	0.93	0.6561	0.999912	0.00125	0.00125	0.2331
512	0.01363	0.002773	0.93	0.4305	0.999823	0.000625	0.001768	0.2275
1024	0.02726	0.001961	0.93	0.6561	0.999912	0.00125	0.00125	0.2299
2048	0.03856	0.007843	0.93	0.6561	0.999912	0.00125	0.003536	0.2318
4096	0.02726	0.0009804	0.9505	0.6561	0.998586	0.00125	0.005	0.2337

Table 3: **CIFAR-5M configurations selected by coordinate descent.** Losses are medians over seeds 42–44. At the reference batch $B_0 = 256$, the hyperparameter search ties the two weight decays, $\lambda_{\mathrm{M}} = \lambda_{\mathrm{A}}$. Displayed hyperparameters are rounded.

E.3 Language-model sweep

Protocol and reference configuration. We train a decoder-only Transformer on the FineWeb10B token stream. The model has 12 layers, width 768, 6 attention heads, context length 1024, squared-ReLU feed-forward layers of width 3072, and a vocabulary of 50,304 tokens. MuonW updates the two-dimensional matrices within each Transformer block using Nesterov momentum and twelve Newton–Schulz iterations. Embeddings, the output projection, and one-dimensional parameters use AdamW. The grid uses bfloat16 on eight GPUs and seed 1. The metric is terminal token-averaged validation cross-entropy on a fixed set of 10,485,760 tokens. Table 4 gives the reference configuration, whose reference audit run has loss 3.26547. Batch labels use binary multiples, with $128\mathrm{K} = 131{,}072$ reference tokens and targets 256K, 512K, 1M, 2M, or 262,144, 524,288, 1,048,576, and 2,097,152 tokens. The budget $\tau = 1{,}703{,}936{,}000$ tokens gives 13,000 reference steps and 6,500, 3,250, 1,625, and 813 target steps. Learning rates remain at their peak for the first 20% of training and decrease linearly to zero over the remaining 80%. Both weight decays start at full strength without warmup; auxiliary decay remains 0.001 in the main grid. All 216 rules use this schedule shape at each target, giving 864 grid runs. An additional sweep at 1M and 2M tokens shows that scaling auxiliary weight decay offers little benefit.

B	η_{M}	$\eta_{\mathrm{Emb.}}$	$\eta_{\mathrm{Out.}}$	η_{Scalar}	μ	β_1	β_2	λ_{M}	λ_{A}
128K	0.0176777	0.494974	0.00282842	0.0106066	0.95	0.8	0.993629	0.025	0.001

Table 4: **Language-model reference configuration.** Batch size B counts tokens. The auxiliary learning rates $\eta_{\mathrm{Emb.}}$, $\eta_{\mathrm{Out.}}$, and η_{Scalar} apply to embeddings, the output projection, and scalar parameters, respectively. Learning rates are peak values, and the cooldown fraction is 0.8. Displayed values are rounded.

The best learning-rate scaling changes with the target batch. The best rules scale both learning-rate groups by the square root at 256K and 512K, but keep them fixed and scale matrix weight decay linearly at 1M and 2M. All four winners keep Muon momentum and Adam β_1 fixed and transport Adam β_2. Table 5 reports their losses and choices, and Figure 14 shows the mean loss across rules with each coordinate choice.

		Parameter scaling			EMA scaling		
Batch	Loss	Matrix LR	Matrix WD	Auxiliary LR	Muon μ	Adam β_1	Adam β_2
256K	3.26553	Square root	Fixed	Square root	Fixed	Fixed	EMA
512K	3.28004	Square root	Square root	Square root	Fixed	Fixed	EMA
1M	3.31183	Fixed	Linear	Fixed	Fixed	Fixed	EMA
2M	3.37977	Fixed	Linear	Fixed	Fixed	Fixed	EMA

Table 5: Lowest validation loss observed independently at each target batch size. The choice at one batch size is not required to transfer to another batch size.

Fixed learning rates and linear matrix weight decay give the smallest mean BSZ regret. The selected common rule keeps Muon momentum and Adam β_1 fixed and transports only Adam β_2. Its mean BSZ regret is 0.00151 and its maximum BSZ regret is 0.00416. It attains the grid minimum

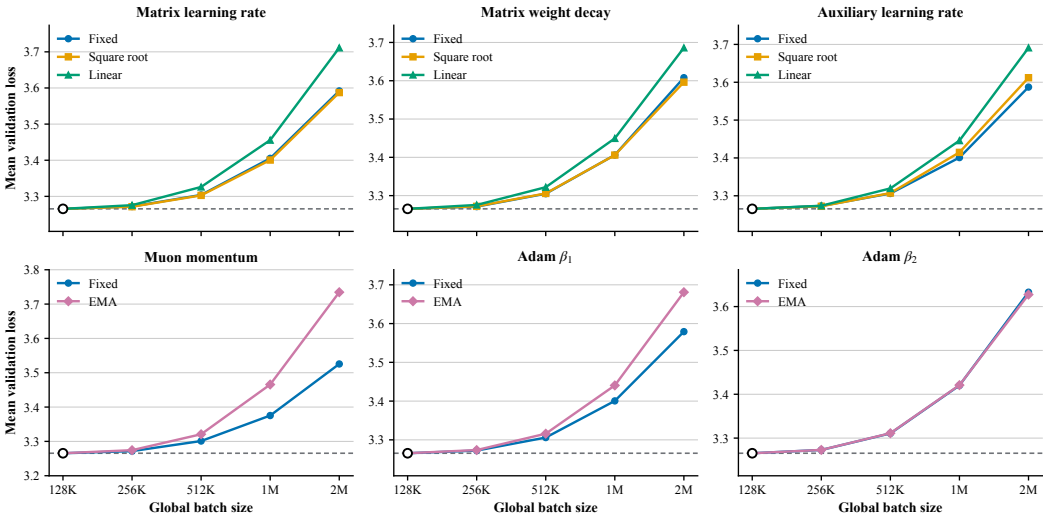

Figure 14: **All** 216 **language-model scaling rules, summarized one coordinate at a time.** Each point is the arithmetic mean of the target-batch validation losses over all rules using the indicated choice on that coordinate, weighting all combinations of the other five choices equally. Fixed, square root, and linear denote exponents $0, \frac{1}{2}, 1$; E/EMA preserves decay over the complete token budget. Lower is better; each panel uses its own focused vertical range. Every rule shares the reference run at $B_0 = 128\text{K}$, drawn as the outlined marker and the dashed horizontal line. The full grid is reported in Section E.3.

at 1M and 2M; its gaps at 256K and 512K are 0.00416 and 0.00188. Table 6 gives the resulting hyperparameters; Emb. and Out. denote the embedding and output-projection learning rates. Table 7 changes one choice at a time around this rule. Transporting Muon momentum or Adam β_1 increases loss substantially. The rule with β_2 also fixed ranks second by mean BSZ regret, 0.00240, and first by maximum BSZ regret, 0.00399. Transporting β_2 lowers mean loss by 0.000895, but raises loss by 0.00017 at 256K. It lowers loss by 0.00022, 0.00089, and 0.00264 at the other three targets. Figure 15 displays both criteria for all rules.

		Matrix hyperparameter configuration			Auxiliary hyperparameter configuration				
Batch	Steps	LR	WD	μ	Emb. LR	Out. LR	Scalar LR	β_1	β_2
128K	13000	0.0176777	0.025	0.95	0.494974	0.00282842	0.0106066	0.8	0.993629
256K	6500	0.0176777	0.05	0.95	0.494974	0.00282842	0.0106066	0.8	0.987299
512K	3250	0.0176777	0.1	0.95	0.494974	0.00282842	0.0106066	0.8	0.97476
1M	1625	0.0176777	0.2	0.95	0.494974	0.00282842	0.0106066	0.8	0.950156
2M	813	0.0176777	0.4	0.95	0.494974	0.00282842	0.0106066	0.8	0.902854

Table 6: Resolved numerical hyperparameters for the selected common rule. Values are rounded for readability. Auxiliary weight decay is 0.001 at every batch size, and the learning rate cooldown occupies the final 80 percent of training.

Only change from selected rule	256K	512K	1M	2M	Mean BSZ regret
Selected common rule	**3.26969**	**3.28192**	**3.31183**	**3.37977**	**0.00151**
Matrix LR to square root	3.27140	3.29573	3.35026	3.46358	0.03595
Matrix LR to linear	3.28325	3.33974	3.44900	3.66492	0.12493
Matrix WD to fixed	3.27030	3.28906	3.32787	3.40266	0.01318
Matrix WD to square root	3.26862	3.28334	3.31700	3.38662	0.00460
Auxiliary LR to square root	3.26763	3.28119	3.31645	3.41001	0.00953
Auxiliary LR to linear	3.26985	3.28710	3.35634	3.45897	0.03377
Muon momentum to EMA	3.27375	3.30073	3.39503	3.58196	0.07858
Adam β_1 to EMA	3.27078	3.28823	3.33705	3.45730	0.02905
Adam β_2 to fixed	3.26952	3.28214	3.31272	3.38241	0.00240

Table 7: One coordinate changes around the selected common rule. Every row changes exactly one entry and keeps the other five entries fixed. Mean BSZ regret is the equally weighted mean over the four target batch-size choices $256\text{K}, 512\text{K}, 1\text{M}, 2\text{M}$ of each row's loss gap to the best tested rule at the same batch size.

The selected matrix scaling agrees with a Power Lines prescription. It matches the fixed-learning-rate member of the Power Lines family in (24), preserving its nominal parameter-averaging timescale at fixed token budget (Bergsma et al., 2025). The choice to transport Adam β_2 does not follow from that timescale.

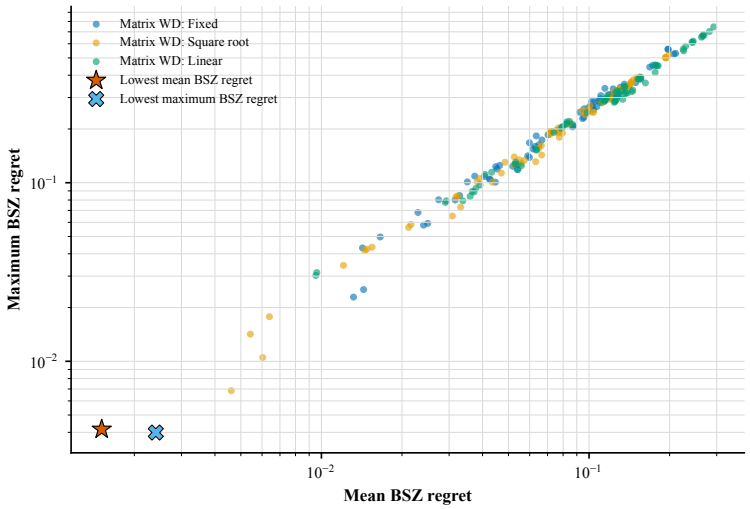

Figure 15: Mean and maximum BSZ regret for all 216 transfer rules. At each batch size, BSZ regret is the rule's validation loss minus the lowest validation loss among the tested rules at that same batch size. Each point represents one rule; the mean (horizontal axis, equal weights) and maximum (vertical axis) are taken over the four target batch-size choices $256\text{K}, 512\text{K}, 1\text{M}, 2\text{M}$ tokens. The star has the lowest mean BSZ regret, and the cross has the lowest maximum BSZ regret. Lower is better; both axes use logarithmic scales.

The retuning comparison combines the grid with additional searches and repeated runs. The language-model full-retuning curve in Figure 2 reports medians over 2–4 seeds, except at 256K, where it uses the best grid loss as described below. Other points are single runs.

At 256K, we used the coordinate-descent procedure detailed in Section F.3, with an acceptance threshold of 0.001. The initial matrix learning rate was 0.0210224, with embedding, output-projection, and scalar learning rates $(0.588627, 0.00336358, 0.0126134)$. The remaining initial values were $\mu = 0.95$, $\beta_1 = 0.8$, $\beta_2 = 0.984992$, matrix weight decay 0.05, and auxiliary cooldown fraction 0.8. No move passed the threshold. Using the same seed, the search configuration's loss 3.2678 matched the 3.2677 of square-root learning-rate scaling with all other coordinates fixed. In the original grid,

that scaling rule gave 3.2660, within 0.0005 of the best grid loss 3.2655, which additionally transports β_2. The search found no improvement exceeding its threshold, and the plotted full-retuning point uses the best grid loss 3.2655.

Scaling auxiliary weight decay offers little benefit. At 1M and 2M tokens, we extend each of the original 216 rules with square-root and linear scaling of auxiliary weight decay from its reference value 0.001, adding 432 single-seed runs per target. For the original common rule, square-root scaling reduces validation loss by only 0.0037 and 0.0027 nats at these two targets, respectively. Across all 216 rules, it instead raises loss by medians of 0.001 and 0.005 nats relative to fixed auxiliary decay. Linear scaling raises loss by medians of 0.012 and 0.033 nats, respectively. Table 9 reports the full measurements alongside the original rules.

E.4 Comparison across settings

The complete grids favor different common rules. We select a common rule from all 216 language-model rules and all 648 CIFAR-5M rules, using each experiment's target batches. The language-model sweep selects fixed learning rates and linear matrix weight decay, whereas CIFAR-5M selects square-root learning rates and fixed weight decays. Both selected rules keep Muon momentum and Adam β_1 fixed; the language-model rule additionally transports β_2.

The tested bound-minimizing Muon prescription ranks better in both settings. Both prescriptions fix matrix weight decay. The bound-minimizing prescription uses square-root learning-rate scaling with fixed momentum; trajectory matching uses linear scaling with retention-preserving momentum. Varying the unspecified AdamW choices gives 12 rules per family for language modeling and 36 for CIFAR-5M. We rank each rule by its mean loss over $\kappa \in \{2, 4, 8, 16\}$ within the complete grid, averaging tied ranks. We then average ranks within each family; lower is better. The bound-minimizing and trajectory-matching families rank 52.1 versus 141.9 among 216 language-model rules, and 59.9 versus 347.7 among 648 CIFAR-5M rules.

The rankings depend on the setting and batch range. The main grids use one seed and one reference configuration per setting. The repeated runs described above provide checks for selected configurations. A family rank averages over its AdamW completions, whereas a coordinate panel averages losses over all compatible rules. The dependence on setting and batch range motivates independent retuning of every optimizer at every batch size in the optimizer comparisons of Section 4.

E.5 Complete measurements

Table 8 reports all 3,888 CIFAR-5M measurements, with one row per rule and six target-batch losses. Its entries are cross-entropies averaged over the final ten evaluations. Table 9 combines the 864 original language-model terminal losses with the 864 additional auxiliary weight-decay measurements. Each original rule is followed by its square-root and linear auxiliary-decay variants, which were evaluated only at 1M and 2M. All 864 original language-model runs reached their terminal step with finite loss; independently collected training logs and the stored results agree on their statuses and losses. The CIFAR-5M rows and language-model groups retain their original order by mean BSZ regret and highlight the selected common rule in each setting. Language-model ranks and mean BSZ regrets use the four-batch original grid; the partial auxiliary-decay variants have no four-batch rank or BSZ regret.

Table 8: Complete CIFAR-5M validation losses for all 648 scaling rules at six target batches, ordered by mean BSZ regret $\overline{\Delta}$. This is the equally weighted mean over the six target batch-size choices of the loss gap to the best tested rule at the same batch size. Each loss is cross-entropy in nats averaged over the final ten evaluations of one run with seed 42; lower is better. Batch sizes count examples. Rule codes are F (fixed), S (square-root), L (linear), and E (retention preservation in processed examples), with $q(B) = q(256)^{B/256}$. Bold marks the selected common rule. Ordering and BSZ regret use unrounded losses; displayed values have five decimal places.

Rank	Parameter scaling				EMA scaling			Validation loss						Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	64	128	512	1024	2048	4096	$\overline{\Delta}$
1	**S**	**F**	**S**	**F**	**F**	**F**	**F**	**0.23370**	**0.23136**	**0.23373**	**0.22778**	**0.23290**	**0.23683**	**0.00492**
2	S	F	F	S	F	F	E	0.22970	0.23049	0.23440	0.23301	0.23772	0.23346	0.00533
3	L	F	S	S	F	E	E	0.23435	0.23346	0.22739	0.22892	0.23043	0.24545	0.00553
4	S	F	S	L	F	F	F	0.23374	0.23289	0.23135	0.23350	0.23206	0.23692	0.00561
5	S	F	S	F	F	F	E	0.23201	0.23276	0.23128	0.23387	0.23190	0.23986	0.00581
6	L	F	F	L	F	E	E	0.23920	0.23106	0.23005	0.23157	0.22957	0.24050	0.00586
7	S	S	S	F	F	F	F	0.23225	0.23205	0.22913	0.23220	0.23285	0.24356	0.00587
8	L	F	S	L	F	E	F	0.23617	0.23393	0.22979	0.22867	0.23112	0.24471	0.00626
9	L	F	F	S	F	E	F	0.23645	0.23487	0.23274	0.23008	0.23139	0.23926	0.00633
10	S	S	S	S	F	F	E	0.23446	0.22672	0.23286	0.23528	0.23395	0.24229	0.00646
11	S	S	F	F	F	F	F	0.23541	0.23033	0.23023	0.23335	0.23795	0.23850	0.00649
12	S	F	F	S	F	E	E	0.23222	0.23431	0.23264	0.23285	0.23470	0.23933	0.00654
13	S	F	S	S	F	F	F	0.23573	0.23188	0.23197	0.23344	0.23361	0.23946	0.00654
14	L	F	F	F	F	E	F	0.23485	0.23564	0.23082	0.22980	0.23418	0.24082	0.00655
15	S	F	S	L	F	F	E	0.23271	0.23274	0.23709	0.23294	0.23137	0.23929	0.00655
16	L	F	F	F	F	F	F	0.23466	0.23235	0.23371	0.23178	0.23467	0.23896	0.00655
17	S	S	F	L	F	F	F	0.23326	0.23247	0.23311	0.23302	0.23524	0.23908	0.00656
18	S	F	F	S	F	E	F	0.23771	0.23346	0.23212	0.23123	0.23277	0.23890	0.00656
19	S	F	L	L	F	F	F	0.23274	0.22955	0.23200	0.23322	0.23499	0.24387	0.00660
20	S	S	F	S	F	E	F	0.23575	0.22800	0.23104	0.23230	0.23775	0.24156	0.00660
21	S	F	L	S	F	F	F	0.23152	0.22898	0.23141	0.23434	0.23801	0.24223	0.00661
22	S	F	F	F	F	E	F	0.23965	0.23636	0.23282	0.22879	0.23039	0.23875	0.00666
23	L	F	S	L	F	F	F	0.23601	0.23336	0.23162	0.23086	0.23319	0.24195	0.00670
24	L	F	S	F	F	E	E	0.23797	0.23275	0.23270	0.23052	0.23373	0.23939	0.00671
25	S	F	S	L	F	E	F	0.23242	0.23465	0.23191	0.23160	0.23448	0.24208	0.00672
26	S	S	F	L	F	E	F	0.23437	0.23431	0.23329	0.22919	0.23629	0.24025	0.00681
27	L	F	L	F	F	F	F	0.23606	0.23260	0.22744	0.23053	0.23563	0.24546	0.00682
28	L	F	F	S	F	F	F	0.23506	0.23066	0.23496	0.23200	0.23529	0.23984	0.00683
29	L	F	F	S	F	F	E	0.23365	0.23415	0.23104	0.23261	0.23606	0.24062	0.00689
30	S	F	F	L	F	F	F	0.23562	0.23213	0.23162	0.23386	0.23924	0.23587	0.00692
31	L	F	F	L	F	F	F	0.23737	0.23089	0.23401	0.23361	0.23587	0.23669	0.00694
32	L	F	F	L	F	F	E	0.23440	0.23318	0.23027	0.23111	0.23851	0.24147	0.00702
33	S	F	F	L	F	F	E	0.23401	0.23265	0.23213	0.23547	0.23844	0.23642	0.00705
34	S	F	F	F	F	F	E	0.23405	0.23061	0.23345	0.23715	0.23542	0.23848	0.00706
35	L	F	F	L	F	E	F	0.23930	0.23384	0.23235	0.23353	0.23082	0.23951	0.00709
36	S	S	F	F	F	E	E	0.23446	0.23168	0.23396	0.23220	0.23711	0.24002	0.00710
37	L	F	F	S	F	E	E	0.23785	0.23729	0.23425	0.23093	0.23132	0.23796	0.00713
38	S	S	L	F	F	F	F	0.23014	0.23376	0.23333	0.23282	0.23597	0.24400	0.00720
39	S	S	S	F	F	E	F	0.23530	0.23517	0.23086	0.23279	0.23315	0.24298	0.00724
40	S	S	S	S	F	F	F	0.23548	0.22970	0.23502	0.23715	0.23296	0.24001	0.00725
41	S	F	F	L	F	E	F	0.23877	0.23369	0.22969	0.23564	0.23206	0.24056	0.00727
42	S	S	S	L	F	F	E	0.23488	0.23050	0.23564	0.23457	0.23353	0.24154	0.00731
43	S	S	S	L	F	F	F	0.23385	0.23093	0.23139	0.23673	0.23544	0.24246	0.00733
44	S	S	F	F	F	E	F	0.23458	0.23608	0.23285	0.23403	0.23239	0.24105	0.00736
45	S	F	F	F	F	E	E	0.23713	0.23283	0.23481	0.22828	0.23453	0.24361	0.00740
46	L	F	L	F	F	E	E	0.23609	0.23318	0.22875	0.22865	0.23400	0.25056	0.00740
47	S	S	F	S	F	E	E	0.23288	0.23707	0.23334	0.23267	0.23676	0.23853	0.00741
48	S	F	S	F	F	E	F	0.23708	0.23432	0.23218	0.23113	0.23380	0.24277	0.00741
49	S	F	L	S	F	E	E	0.23235	0.23258	0.22931	0.23226	0.23936	0.24574	0.00746
50	S	F	L	F	F	F	F	0.23286	0.23046	0.23393	0.23404	0.23501	0.24562	0.00752
51	S	S	L	S	F	F	F	0.23339	0.23378	0.22918	0.23350	0.23708	0.24509	0.00753
52	L	F	F	F	F	F	E	0.23776	0.23367	0.23210	0.23335	0.23781	0.23760	0.00758
53	S	S	F	L	F	E	E	0.23435	0.23329	0.23205	0.23417	0.23653	0.24195	0.00759
54	L	F	L	S	F	F	F	0.23425	0.23620	0.23274	0.23349	0.22947	0.24622	0.00759
55	S	S	L	S	F	F	E	0.23165	0.22933	0.23646	0.23449	0.23808	0.24258	0.00763
56	L	F	S	F	F	F	E	0.23661	0.23328	0.23230	0.23296	0.23495	0.24260	0.00765
57	S	F	S	L	F	E	E	0.23265	0.23523	0.23304	0.23411	0.23507	0.24305	0.00772
58	L	F	S	S	F	E	F	0.23773	0.23751	0.23085	0.22890	0.23224	0.24604	0.00774
59	S	F	F	F	F	F	F	0.23556	0.23316	0.23480	0.23572	0.23833	0.23579	0.00776
60	L	F	S	F	F	E	F	0.23699	0.23784	0.23029	0.23178	0.23148	0.24507	0.00777
61	L	S	L	S	F	F	F	0.23237	0.22930	0.23040	0.23374	0.23585	0.25203	0.00781
62	S	S	L	F	F	F	E	0.23292	0.23146	0.23463	0.23398	0.23751	0.24335	0.00784
63	L	F	L	L	F	F	F	0.23679	0.23537	0.23030	0.23407	0.23076	0.24656	0.00784
64	S	F	S	S	F	E	F	0.23565	0.23356	0.23163	0.23161	0.23667	0.24498	0.00788
65	S	L	S	L	F	F	E	0.22906	0.22799	0.23604	0.23675	0.23878	0.24567	0.00791
66	L	F	L	S	F	F	E	0.23722	0.23444	0.23099	0.23240	0.23288	0.24655	0.00794
67	S	S	F	L	F	F	E	0.23287	0.23270	0.23561	0.23524	0.23830	0.23982	0.00796

Continued on the next page

Table 8 continued

Rank	Parameter scaling				EMA scaling			Validation loss						Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	64	128	512	1024	2048	4096	$\overline{\Delta}$
68	L	S	F	S	F	E	E	0.23740	0.23360	0.23160	0.23009	0.23551	0.24637	0.00796
69	L	F	S	F	F	F	F	0.23711	0.23376	0.23065	0.23701	0.23326	0.24304	0.00800
70	S	L	S	L	F	E	F	0.23370	0.22882	0.23229	0.23223	0.23664	0.25119	0.00801
71	S	F	L	S	F	F	E	0.23365	0.23429	0.23180	0.23685	0.23503	0.24331	0.00802
72	S	S	S	L	F	E	F	0.23430	0.22796	0.23580	0.23520	0.23442	0.24739	0.00804
73	L	F	S	S	F	F	F	0.23680	0.23227	0.23094	0.23303	0.23522	0.24705	0.00808
74	L	F	F	F	F	E	E	0.23814	0.23685	0.23183	0.23150	0.23289	0.24431	0.00812
75	L	S	F	F	F	F	F	0.23192	0.23515	0.23188	0.23441	0.23653	0.24569	0.00813
76	S	F	L	L	F	F	E	0.23253	0.23312	0.23446	0.23228	0.23642	0.24683	0.00814
77	S	S	F	S	F	F	F	0.23543	0.23008	0.23556	0.23303	0.24188	0.23970	0.00815
78	S	S	F	F	F	F	E	0.23460	0.22991	0.23474	0.23641	0.24105	0.23900	0.00815
79	L	S	S	L	F	F	E	0.23356	0.23285	0.23176	0.23218	0.23529	0.25030	0.00819
80	S	S	S	F	F	F	E	0.23511	0.23078	0.23579	0.23458	0.23498	0.24473	0.00819
81	S	F	L	F	F	F	E	0.23439	0.23368	0.23528	0.23617	0.23486	0.24167	0.00821
82	S	F	S	S	F	F	E	0.23528	0.22959	0.23638	0.23887	0.23407	0.24206	0.00824
83	L	S	F	F	F	E	E	0.23504	0.23336	0.23228	0.23130	0.23731	0.24697	0.00824
84	L	F	S	L	F	F	E	0.23579	0.23472	0.23353	0.23493	0.23752	0.24000	0.00828
85	S	L	S	F	F	F	F	0.23268	0.23223	0.23366	0.23457	0.23890	0.24454	0.00829
86	L	S	F	S	F	E	F	0.23383	0.23414	0.23470	0.23432	0.23573	0.24385	0.00829
87	L	F	S	S	F	F	E	0.23616	0.23507	0.23169	0.23222	0.23494	0.24679	0.00834
88	L	F	L	S	F	E	F	0.23839	0.23777	0.22942	0.23094	0.23435	0.24623	0.00838
89	F	L	F	S	F	E	F	0.22959	0.23241	0.23371	0.23678	0.23760	0.24741	0.00845
90	L	F	L	F	F	F	E	0.23689	0.23527	0.23362	0.23004	0.23364	0.24812	0.00846
91	L	F	L	F	F	E	F	0.23588	0.23317	0.23063	0.23035	0.23287	0.25476	0.00847
92	S	L	S	S	F	F	E	0.22644	0.23206	0.23658	0.23639	0.23883	0.24755	0.00851
93	S	F	L	F	F	E	E	0.23392	0.23050	0.23167	0.23445	0.24009	0.24730	0.00852
94	S	F	F	L	F	E	E	0.24060	0.23491	0.23441	0.23198	0.23581	0.24028	0.00853
95	L	S	L	F	F	F	F	0.23194	0.23394	0.22929	0.23506	0.23610	0.25183	0.00856
96	F	L	S	L	F	F	F	0.22972	0.23176	0.23635	0.23295	0.23803	0.24938	0.00856
97	S	F	F	S	F	F	F	0.23378	0.23316	0.23333	0.23725	0.24208	0.23863	0.00857
98	S	F	S	S	F	E	E	0.23477	0.23074	0.23265	0.23478	0.23872	0.24659	0.00857
99	F	S	L	L	F	F	F	0.22879	0.23190	0.23416	0.23704	0.23851	0.24803	0.00860
100	S	S	L	L	F	F	F	0.23505	0.23256	0.23247	0.23209	0.23982	0.24656	0.00862
101	S	S	F	S	F	F	E	0.23515	0.23263	0.23491	0.23442	0.24101	0.24079	0.00868
102	L	S	F	F	F	E	F	0.23770	0.23587	0.23229	0.23143	0.23560	0.24620	0.00872
103	S	L	F	S	F	F	F	0.22854	0.22670	0.23551	0.23523	0.24501	0.24828	0.00874
104	L	S	F	L	F	E	F	0.23831	0.23524	0.23089	0.23319	0.23606	0.24631	0.00887
105	S	L	F	S	F	E	F	0.23341	0.23290	0.23388	0.23441	0.23733	0.24821	0.00889
106	L	S	F	L	F	F	F	0.23483	0.23207	0.23187	0.23462	0.23745	0.24939	0.00890
107	S	S	S	L	F	E	E	0.23386	0.23103	0.23502	0.23254	0.23897	0.24892	0.00892
108	F	S	S	L	F	F	F	0.23026	0.22906	0.23319	0.23578	0.24261	0.24948	0.00893
109	S	S	S	S	F	E	F	0.23259	0.23529	0.23460	0.23560	0.23549	0.24688	0.00894
110	L	F	S	L	F	E	E	0.23558	0.23947	0.23505	0.22983	0.23351	0.24706	0.00895
111	L	S	F	L	F	E	E	0.23873	0.23317	0.23331	0.23187	0.23597	0.24777	0.00900
112	F	S	F	S	F	F	E	0.23298	0.22720	0.23168	0.23765	0.24318	0.24843	0.00905
113	S	F	L	L	F	E	F	0.23114	0.23536	0.23216	0.23508	0.23570	0.25187	0.00908
114	S	F	S	F	F	E	E	0.23587	0.23466	0.23539	0.23275	0.23494	0.24787	0.00911
115	L	S	F	F	F	F	E	0.23357	0.23661	0.23191	0.23452	0.23838	0.24653	0.00912
116	F	F	F	S	F	E	F	0.23789	0.23153	0.23325	0.23440	0.23613	0.24862	0.00917
117	L	S	S	S	F	E	F	0.23607	0.23142	0.23416	0.23192	0.23987	0.24840	0.00917
118	S	S	L	F	F	E	F	0.23612	0.23443	0.23062	0.23610	0.23815	0.24645	0.00918
119	F	S	F	L	F	E	F	0.23423	0.23198	0.23575	0.23560	0.23636	0.24800	0.00919
120	S	S	L	L	F	F	E	0.23108	0.23136	0.23287	0.23863	0.23791	0.25021	0.00921
121	S	S	L	F	F	E	E	0.23206	0.23435	0.23116	0.23586	0.23860	0.25029	0.00925
122	L	S	S	L	F	F	F	0.23464	0.23248	0.23412	0.23365	0.23667	0.25086	0.00927
123	F	S	F	F	F	F	E	0.22983	0.23412	0.23218	0.23673	0.24176	0.24785	0.00928
124	S	L	S	F	F	E	F	0.22954	0.23069	0.23178	0.23672	0.23759	0.25619	0.00928
125	S	L	F	L	F	E	E	0.23034	0.22938	0.23275	0.23878	0.23810	0.25317	0.00929
126	L	S	S	L	F	E	E	0.23175	0.23295	0.23278	0.23163	0.23905	0.25438	0.00929
127	S	F	L	S	F	E	F	0.23742	0.23706	0.23358	0.23155	0.23690	0.24624	0.00932
128	L	S	S	F	F	F	F	0.23431	0.23479	0.23274	0.23284	0.23643	0.25195	0.00937
129	F	S	F	F	F	E	E	0.23116	0.23254	0.23420	0.23667	0.23783	0.25070	0.00938
130	S	S	S	S	F	E	E	0.23865	0.23224	0.23407	0.23286	0.23674	0.24860	0.00939
131	L	S	S	L	F	E	F	0.23998	0.23222	0.23300	0.23198	0.23872	0.24730	0.00940
132	S	L	F	L	F	F	E	0.23009	0.23107	0.23341	0.23693	0.24503	0.24674	0.00941
133	S	S	S	F	F	E	E	0.23174	0.23292	0.23553	0.23482	0.23506	0.25321	0.00941
134	F	S	L	S	F	F	F	0.23421	0.23060	0.23431	0.23675	0.23540	0.25202	0.00941
135	S	S	L	S	F	E	F	0.23211	0.23519	0.23106	0.23268	0.24032	0.25195	0.00942
136	F	S	S	S	F	F	E	0.23034	0.23200	0.23248	0.23393	0.23894	0.25565	0.00942
137	F	L	F	L	F	E	F	0.23330	0.23434	0.23494	0.23586	0.23554	0.24938	0.00942
138	F	L	S	S	F	F	E	0.23049	0.23013	0.23388	0.23648	0.24057	0.25184	0.00943
139	F	S	F	L	F	F	E	0.23061	0.23294	0.23527	0.23404	0.24394	0.24683	0.00947
140	L	S	S	F	F	F	E	0.23575	0.23043	0.23386	0.23633	0.23424	0.25320	0.00950

Continued on the next page

Table 8 continued

Rank	Parameter scaling				EMA scaling			Validation loss						Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	64	128	512	1024	2048	4096	$\overline{\Delta}$
141	S	L	F	F	F	F	F	0.23199	0.23239	0.23643	0.23585	0.24230	0.24485	0.00950
142	F	S	S	L	F	E	F	0.23409	0.23285	0.23283	0.23437	0.23994	0.24973	0.00950
143	F	S	L	F	F	F	F	0.23380	0.23304	0.23323	0.23704	0.23822	0.24851	0.00951
144	L	F	L	L	F	F	E	0.23777	0.23582	0.23305	0.23372	0.23310	0.25066	0.00955
145	F	S	F	F	F	E	F	0.23609	0.23050	0.23362	0.23783	0.23730	0.24887	0.00957
146	F	F	F	F	F	E	E	0.23690	0.23247	0.23280	0.23584	0.23525	0.25107	0.00958
147	L	F	L	S	F	E	E	0.23590	0.23441	0.23403	0.22950	0.23613	0.25438	0.00959
148	F	S	L	S	F	F	E	0.22910	0.23380	0.23352	0.23793	0.24005	0.24997	0.00959
149	S	F	L	F	F	E	F	0.23519	0.23246	0.23451	0.23407	0.23804	0.25037	0.00964
150	S	L	S	F	F	E	E	0.23181	0.23409	0.23249	0.23709	0.23647	0.25277	0.00965
151	F	S	F	L	F	F	F	0.22954	0.23247	0.23410	0.23711	0.24304	0.24849	0.00966
152	L	S	F	S	F	F	E	0.23310	0.23359	0.23062	0.23454	0.24338	0.24963	0.00967
153	F	F	F	L	F	E	F	0.23785	0.23463	0.23108	0.23394	0.23826	0.24921	0.00969
154	F	L	L	F	F	F	E	0.23182	0.22970	0.23035	0.24070	0.24164	0.25079	0.00970
155	F	F	F	S	F	E	E	0.23540	0.23309	0.23636	0.23515	0.23723	0.24779	0.00970
156	F	F	F	F	F	F	E	0.23594	0.23391	0.23540	0.23697	0.23687	0.24611	0.00973
157	S	L	F	F	F	E	F	0.23349	0.23239	0.23396	0.23491	0.23815	0.25235	0.00974
158	S	L	L	S	F	F	F	0.23393	0.22757	0.23268	0.23721	0.24017	0.25390	0.00977
159	F	S	S	S	F	E	F	0.23245	0.23400	0.23353	0.23599	0.23831	0.25122	0.00978
160	L	S	F	L	F	F	E	0.23581	0.23397	0.23495	0.23360	0.23822	0.24906	0.00980
161	F	S	F	S	F	E	F	0.23505	0.23377	0.23411	0.23491	0.23782	0.24996	0.00980
162	F	L	L	S	F	F	F	0.22995	0.23420	0.23194	0.24004	0.23931	0.25020	0.00981
163	S	L	F	S	F	E	E	0.23536	0.23395	0.23156	0.23255	0.24099	0.25128	0.00981
164	F	S	F	S	F	E	E	0.23487	0.23462	0.23337	0.23571	0.23769	0.24950	0.00983
165	F	L	L	S	F	F	E	0.23192	0.22929	0.23622	0.23943	0.23444	0.25452	0.00984
166	L	F	L	L	F	E	E	0.23580	0.23349	0.23366	0.23178	0.23372	0.25759	0.00987
167	F	L	L	F	F	F	F	0.22787	0.23049	0.23663	0.23944	0.23835	0.25344	0.00990
168	F	S	F	L	F	E	E	0.23549	0.23136	0.23688	0.23438	0.23804	0.25022	0.00993
169	S	S	L	S	F	E	E	0.23390	0.23184	0.23408	0.23406	0.23732	0.25528	0.00995
170	S	L	F	L	F	F	F	0.23173	0.23252	0.23202	0.24164	0.24144	0.24718	0.00996
171	L	S	L	S	F	F	E	0.23305	0.23262	0.23527	0.23512	0.23956	0.25097	0.00996
172	F	F	S	F	F	F	E	0.23691	0.23118	0.23449	0.23457	0.23632	0.25314	0.00997
173	F	F	S	L	F	F	F	0.23316	0.23547	0.23216	0.23387	0.23985	0.25210	0.00997
174	F	S	F	F	F	F	F	0.23415	0.23158	0.23381	0.23699	0.24278	0.24731	0.00997
175	F	L	S	L	F	E	E	0.22910	0.23292	0.23217	0.23454	0.24480	0.25315	0.00998
176	F	L	S	F	F	F	E	0.22831	0.23221	0.23600	0.23646	0.24254	0.25125	0.00999
177	L	S	L	S	F	E	E	0.23883	0.23318	0.23058	0.23107	0.23949	0.25364	0.01000
178	L	S	S	S	F	F	F	0.23772	0.23244	0.23435	0.23613	0.23554	0.25074	0.01002
179	L	S	F	S	F	F	F	0.23466	0.23181	0.23513	0.23808	0.23948	0.24798	0.01006
180	S	L	L	F	F	F	F	0.23231	0.23082	0.23496	0.23758	0.24027	0.25128	0.01007
181	F	L	F	F	F	F	E	0.22905	0.23250	0.23344	0.24055	0.24220	0.24974	0.01011
182	L	F	L	L	F	E	F	0.24048	0.23571	0.22994	0.23004	0.23677	0.25463	0.01013
183	F	S	F	S	F	F	F	0.23462	0.23071	0.23032	0.23857	0.24661	0.24676	0.01013
184	F	L	S	S	F	F	F	0.22976	0.23251	0.23057	0.23802	0.24277	0.25402	0.01014
185	S	L	L	L	F	F	E	0.23023	0.23084	0.23155	0.23853	0.24269	0.25401	0.01017
186	L	S	L	L	F	E	E	0.23511	0.23091	0.22970	0.23013	0.24277	0.25925	0.01018
187	F	L	F	L	F	F	E	0.23412	0.23314	0.23451	0.23687	0.24265	0.24670	0.01020
188	L	S	S	F	F	E	E	0.23582	0.23584	0.23007	0.23082	0.23855	0.25696	0.01021
189	F	F	L	S	F	F	F	0.23512	0.23586	0.23313	0.23615	0.23995	0.24808	0.01025
190	S	L	S	S	F	F	F	0.23415	0.23270	0.23465	0.23924	0.23921	0.24840	0.01026
191	F	F	S	F	F	E	E	0.23273	0.23702	0.23082	0.23495	0.23874	0.25412	0.01026
192	S	L	S	L	F	F	F	0.23064	0.23022	0.23808	0.23525	0.24343	0.25074	0.01026
193	F	F	S	F	F	F	F	0.23506	0.23448	0.23098	0.23534	0.24184	0.25079	0.01028
194	S	L	L	F	F	F	E	0.23341	0.23308	0.23281	0.23657	0.24153	0.25112	0.01028
195	S	L	S	S	F	E	E	0.23232	0.23341	0.23234	0.23504	0.23707	0.25847	0.01031
196	F	S	S	F	F	F	F	0.23164	0.23336	0.23258	0.23873	0.23991	0.25253	0.01032
197	F	F	F	L	F	F	F	0.23727	0.23470	0.23516	0.23449	0.24038	0.24687	0.01034
198	F	S	L	F	F	F	E	0.23270	0.23065	0.23251	0.23902	0.24119	0.25283	0.01035
199	F	L	F	F	F	E	E	0.23221	0.23550	0.23366	0.23705	0.23888	0.25161	0.01035
200	F	F	F	L	F	F	E	0.23312	0.23383	0.23287	0.23938	0.24045	0.24937	0.01037
201	F	F	F	S	F	F	F	0.23469	0.23252	0.23491	0.23665	0.24177	0.24855	0.01038
202	F	F	L	L	F	F	F	0.23854	0.23054	0.23289	0.24134	0.23598	0.24981	0.01038
203	F	S	S	F	F	E	F	0.23431	0.23478	0.23251	0.23677	0.23964	0.25118	0.01040
204	S	S	L	L	F	E	F	0.23276	0.23381	0.23429	0.23340	0.24250	0.25248	0.01040
205	F	F	F	L	F	E	E	0.23925	0.23353	0.23415	0.23709	0.23737	0.24792	0.01042
206	F	L	S	S	F	E	F	0.22960	0.23012	0.23636	0.23878	0.24063	0.25387	0.01042
207	S	L	L	S	F	F	E	0.23153	0.23147	0.23301	0.23973	0.23881	0.25484	0.01043
208	F	L	S	S	F	E	E	0.23028	0.23332	0.23256	0.23544	0.24205	0.25590	0.01046
209	F	L	F	S	F	F	F	0.23475	0.23015	0.23667	0.24042	0.24067	0.24690	0.01046
210	S	L	S	S	F	E	F	0.23122	0.23577	0.23536	0.23650	0.24026	0.25049	0.01047
211	F	F	F	S	F	F	E	0.23583	0.23280	0.23374	0.23552	0.24237	0.24940	0.01048
212	F	S	S	L	F	F	E	0.23068	0.23357	0.23688	0.23830	0.23711	0.25323	0.01049
213	F	F	S	F	F	E	F	0.23711	0.23678	0.23027	0.23693	0.23980	0.24891	0.01050

Continued on the next page

Table 8 continued

Rank	Parameter scaling				EMA scaling			Validation loss						Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	64	128	512	1024	2048	4096	$\overline{\Delta}$
214	S	L	S	F	F	F	E	0.23169	0.22845	0.23675	0.24247	0.23944	0.25101	0.01050
215	F	L	F	F	F	E	F	0.23603	0.23049	0.23909	0.23759	0.23744	0.24923	0.01051
216	L	S	S	S	F	F	E	0.23750	0.23487	0.22872	0.23599	0.24057	0.25250	0.01055
217	S	L	F	F	F	E	E	0.23560	0.23249	0.23386	0.23854	0.24028	0.24946	0.01057
218	F	L	L	L	F	F	F	0.22937	0.22998	0.23559	0.23964	0.24061	0.25517	0.01059
219	S	L	L	F	F	E	F	0.23163	0.23197	0.23484	0.23849	0.24170	0.25190	0.01062
220	L	S	S	F	F	E	F	0.23845	0.23262	0.23519	0.23506	0.23609	0.25320	0.01063
221	L	S	L	F	F	F	E	0.23571	0.23472	0.23224	0.23545	0.24038	0.25215	0.01064
222	S	L	F	F	F	F	E	0.23430	0.23350	0.23625	0.23430	0.24703	0.24538	0.01066
223	L	S	L	L	F	F	F	0.23874	0.23223	0.23147	0.23332	0.23834	0.25676	0.01067
224	L	F	F	L	E	F	E	0.22805	0.23199	0.23391	0.23223	0.23789	0.26686	0.01069
225	F	S	L	L	F	E	E	0.23649	0.22878	0.23222	0.23615	0.24148	0.25585	0.01069
226	S	L	F	L	F	E	F	0.23627	0.23171	0.23596	0.23857	0.23761	0.25100	0.01072
227	F	F	F	F	F	F	F	0.23811	0.23303	0.23354	0.23711	0.24223	0.24712	0.01072
228	L	F	F	S	E	F	F	0.23101	0.22678	0.23362	0.23190	0.23618	0.27168	0.01073
229	L	S	S	S	F	E	E	0.23535	0.23322	0.23495	0.23215	0.23885	0.25670	0.01074
230	F	S	S	F	F	F	E	0.23430	0.23262	0.23431	0.24039	0.23937	0.25030	0.01075
231	L	S	L	F	F	E	E	0.23528	0.23019	0.23425	0.23298	0.24153	0.25722	0.01077
232	F	L	F	S	F	E	E	0.23452	0.23104	0.23679	0.23793	0.24117	0.25004	0.01078
233	F	F	S	S	F	E	E	0.23733	0.23015	0.23493	0.23436	0.24300	0.25186	0.01080
234	F	F	S	L	F	E	E	0.23695	0.23478	0.23261	0.23466	0.24016	0.25296	0.01088
235	F	S	L	L	F	F	E	0.23404	0.23034	0.23372	0.23923	0.24190	0.25293	0.01089
236	S	L	S	L	F	E	E	0.23576	0.23124	0.23387	0.23843	0.24044	0.25255	0.01091
237	S	L	L	L	F	F	F	0.23125	0.23276	0.23559	0.23646	0.24033	0.25615	0.01095
238	F	F	L	F	F	F	E	0.23593	0.23343	0.23283	0.24083	0.23785	0.25168	0.01096
239	F	L	F	L	F	F	F	0.23107	0.23170	0.23708	0.23719	0.24590	0.24967	0.01097
240	F	F	F	F	F	E	F	0.23967	0.23572	0.23636	0.23543	0.23501	0.25071	0.01102
241	F	L	S	L	F	F	E	0.23169	0.23246	0.23278	0.23850	0.24285	0.25464	0.01102
242	L	F	F	L	E	F	F	0.23317	0.22725	0.23450	0.23173	0.23713	0.26921	0.01103
243	L	F	F	F	E	F	F	0.23395	0.23284	0.22757	0.23475	0.23573	0.26824	0.01104
244	F	L	F	F	F	F	F	0.23496	0.23080	0.23805	0.23984	0.24335	0.24608	0.01105
245	F	S	L	F	F	E	F	0.23495	0.23658	0.23315	0.23499	0.24222	0.25122	0.01105
246	F	S	S	S	F	E	E	0.23276	0.23471	0.23356	0.23723	0.24185	0.25304	0.01106
247	S	L	F	S	F	F	E	0.23515	0.23270	0.23747	0.23772	0.24098	0.24923	0.01107
248	L	S	L	L	F	E	F	0.23700	0.23669	0.22975	0.23069	0.23657	0.26255	0.01107
249	F	S	S	S	F	F	F	0.23307	0.23088	0.23696	0.23563	0.24358	0.25357	0.01115
250	F	L	F	L	F	E	E	0.23076	0.23431	0.23685	0.24175	0.24047	0.24968	0.01117
251	F	F	L	L	F	F	E	0.23570	0.23516	0.23409	0.23774	0.24016	0.25100	0.01118
252	F	L	S	F	F	E	E	0.23165	0.23260	0.23722	0.23443	0.24239	0.25580	0.01121
253	S	L	L	F	F	E	E	0.23135	0.23433	0.23317	0.23856	0.24371	0.25299	0.01121
254	F	F	L	S	F	F	E	0.23698	0.23619	0.23527	0.23762	0.23782	0.25028	0.01122
255	S	S	L	L	F	E	E	0.23598	0.23338	0.23416	0.23143	0.24190	0.25755	0.01126
256	F	S	S	F	F	E	E	0.23577	0.23367	0.23565	0.23230	0.24278	0.25422	0.01126
257	F	L	L	L	F	F	E	0.23126	0.23143	0.23420	0.23925	0.24396	0.25442	0.01129
258	F	L	L	F	F	E	E	0.23211	0.23094	0.23760	0.23548	0.24561	0.25283	0.01129
259	F	F	L	L	F	E	F	0.23933	0.23352	0.23065	0.23631	0.24148	0.25343	0.01132
260	F	F	L	F	F	E	E	0.23789	0.23631	0.23362	0.23410	0.23986	0.25302	0.01133
261	L	S	L	L	F	F	E	0.23638	0.23346	0.23284	0.23186	0.23713	0.26338	0.01137
262	L	S	L	S	F	E	F	0.23536	0.23410	0.22993	0.23678	0.23984	0.25907	0.01138
263	S	L	L	S	F	E	E	0.23631	0.23062	0.23228	0.23471	0.24357	0.25777	0.01141
264	F	F	S	L	F	F	E	0.23584	0.23416	0.23224	0.23932	0.24089	0.25326	0.01148
265	F	L	S	F	F	F	F	0.23174	0.23193	0.23678	0.23914	0.24295	0.25330	0.01150
266	F	S	S	L	F	E	E	0.23278	0.23759	0.23543	0.23184	0.24161	0.25662	0.01151
267	F	L	F	S	F	F	E	0.23197	0.23506	0.23768	0.23373	0.24933	0.24816	0.01152
268	F	L	S	F	F	E	F	0.23028	0.23506	0.23686	0.23858	0.24119	0.25407	0.01154
269	F	F	L	S	F	E	E	0.23511	0.23173	0.23344	0.23499	0.24345	0.25742	0.01155
270	F	F	L	F	F	F	F	0.23453	0.23715	0.23778	0.23477	0.24144	0.25055	0.01157
271	L	S	L	F	F	E	F	0.23857	0.23756	0.23256	0.23361	0.23798	0.25651	0.01166
272	F	F	S	L	F	E	F	0.23707	0.23545	0.23509	0.23868	0.23969	0.25098	0.01169
273	F	F	S	S	F	F	E	0.23534	0.23614	0.23375	0.23821	0.24185	0.25224	0.01179
274	F	L	L	S	F	E	F	0.22984	0.23453	0.23368	0.23941	0.24495	0.25559	0.01187
275	F	F	S	S	F	F	F	0.23695	0.23688	0.23347	0.23716	0.24227	0.25183	0.01196
276	F	L	L	F	F	E	F	0.23253	0.23224	0.23741	0.23662	0.24094	0.25883	0.01196
277	F	L	S	L	F	E	F	0.23427	0.23266	0.23639	0.23833	0.24204	0.25526	0.01202
278	S	F	L	L	F	E	E	0.23908	0.23786	0.23426	0.23115	0.24219	0.25546	0.01220
279	S	L	L	S	F	E	F	0.23588	0.23232	0.23301	0.23679	0.24087	0.26114	0.01220
280	F	S	L	S	F	E	F	0.23794	0.23439	0.23533	0.23364	0.24706	0.25187	0.01224
281	F	F	S	S	F	E	F	0.24013	0.23561	0.23605	0.23308	0.24273	0.25290	0.01228
282	F	F	L	F	F	E	F	0.23703	0.23559	0.23458	0.23462	0.24413	0.25460	0.01229
283	F	L	L	L	F	E	F	0.23379	0.23308	0.23531	0.23691	0.24308	0.25882	0.01236
284	L	F	S	F	E	F	E	0.22864	0.23143	0.23488	0.23133	0.24002	0.27529	0.01246
285	L	F	S	S	E	F	E	0.23042	0.23014	0.23249	0.23577	0.23849	0.27430	0.01247
286	F	S	L	S	F	E	E	0.23640	0.23320	0.23655	0.23766	0.24179	0.25605	0.01247

Continued on the next page

Table 8 continued

Rank	Parameter scaling				EMA scaling			Validation loss						Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	64	128	512	1024	2048	4096	$\overline{\Delta}$
287	L	F	F	S	E	F	E	0.23349	0.23008	0.23520	0.23593	0.24030	0.26694	0.01252
288	F	S	L	F	F	E	E	0.23738	0.23326	0.23637	0.23370	0.24235	0.25900	0.01254
289	S	S	F	F	E	F	E	0.23051	0.22993	0.23125	0.23647	0.24085	0.27428	0.01275
290	L	F	L	F	E	F	F	0.22984	0.23021	0.23513	0.23548	0.23878	0.27385	0.01275
291	F	L	L	S	F	E	E	0.23085	0.23492	0.23432	0.23840	0.24895	0.25604	0.01278
292	S	F	S	L	E	F	F	0.23052	0.22983	0.23767	0.23365	0.24210	0.27028	0.01287
293	F	F	L	S	F	E	F	0.23927	0.23632	0.23513	0.23552	0.24163	0.25689	0.01299
294	L	F	S	L	E	F	F	0.22825	0.22848	0.23331	0.23452	0.24263	0.27786	0.01304
295	L	F	S	S	E	F	F	0.22878	0.23164	0.23569	0.23108	0.24152	0.27636	0.01304
296	F	F	L	L	F	E	E	0.23319	0.23403	0.23246	0.23775	0.24784	0.26066	0.01319
297	L	F	S	F	E	F	F	0.23298	0.23131	0.23362	0.23465	0.23982	0.27413	0.01328
298	L	F	L	S	E	F	F	0.23242	0.23114	0.23545	0.23290	0.24008	0.27512	0.01338
299	S	F	S	S	E	F	F	0.23396	0.22636	0.23344	0.23787	0.24557	0.27025	0.01344
300	S	F	F	F	E	F	E	0.22986	0.22922	0.23310	0.24003	0.24190	0.27340	0.01345
301	L	F	F	F	E	F	E	0.23263	0.23361	0.23447	0.23515	0.23881	0.27291	0.01346
302	S	F	F	F	E	F	F	0.23386	0.23015	0.23542	0.23629	0.24299	0.26919	0.01352
303	S	F	F	S	E	F	F	0.23108	0.23145	0.23782	0.23500	0.24165	0.27110	0.01355
304	L	F	S	L	E	F	E	0.23193	0.23218	0.23337	0.23477	0.24025	0.27620	0.01365
305	L	S	S	L	E	F	F	0.22770	0.22712	0.23269	0.23488	0.24292	0.28352	0.01367
306	S	F	S	F	E	F	F	0.23146	0.22638	0.23666	0.23676	0.24520	0.27239	0.01367
307	S	S	S	L	E	F	F	0.23056	0.23131	0.23609	0.23783	0.24394	0.26918	0.01368
308	L	S	F	L	E	F	E	0.22885	0.22667	0.23503	0.23707	0.24241	0.27924	0.01374
309	S	F	F	S	E	F	E	0.23131	0.23299	0.23532	0.23873	0.24247	0.26869	0.01378
310	S	S	S	F	E	F	F	0.22931	0.23117	0.23424	0.23924	0.24361	0.27229	0.01384
311	S	S	F	S	E	F	F	0.23046	0.23077	0.23453	0.23887	0.24588	0.26957	0.01388
312	S	F	F	L	E	F	F	0.22846	0.23072	0.23656	0.23808	0.24576	0.27054	0.01388
313	S	F	L	F	E	F	F	0.23201	0.22743	0.23439	0.23797	0.24545	0.27287	0.01389
314	S	F	L	S	E	F	F	0.22981	0.22971	0.23573	0.23950	0.24312	0.27237	0.01391
315	F	L	L	L	F	E	E	0.23618	0.23252	0.23337	0.23702	0.24761	0.26373	0.01394
316	S	S	L	F	E	F	F	0.22878	0.23088	0.23619	0.23528	0.24463	0.27511	0.01401
317	S	F	S	F	E	F	E	0.23302	0.22858	0.23518	0.23628	0.24334	0.27505	0.01411
318	S	F	S	L	E	F	E	0.23156	0.22918	0.23717	0.23843	0.24519	0.27006	0.01413
319	S	F	S	S	E	F	E	0.23185	0.23292	0.23989	0.23608	0.24152	0.26932	0.01413
320	L	S	F	S	E	F	E	0.22811	0.22625	0.23556	0.23305	0.24651	0.28211	0.01413
321	S	S	F	L	E	F	E	0.22761	0.22960	0.23517	0.24044	0.24744	0.27155	0.01417
322	S	S	S	F	E	F	E	0.22793	0.22729	0.23616	0.23792	0.24417	0.27891	0.01426
323	L	F	L	F	E	F	E	0.23010	0.22615	0.23277	0.23767	0.24384	0.28189	0.01427
324	L	F	F	L	E	E	E	0.23143	0.23244	0.23064	0.23183	0.24546	0.28068	0.01428
325	S	L	L	L	F	E	F	0.23346	0.23094	0.23553	0.23619	0.24401	0.27299	0.01439
326	S	F	F	L	E	F	E	0.23182	0.23107	0.23731	0.23880	0.24347	0.27107	0.01445
327	S	S	F	L	E	F	F	0.23093	0.22889	0.23323	0.24130	0.24467	0.27455	0.01446
328	S	F	L	F	E	F	E	0.22996	0.22766	0.23527	0.24188	0.24352	0.27545	0.01449
329	S	S	S	S	E	F	F	0.22724	0.23020	0.23549	0.23781	0.24754	0.27590	0.01456
330	L	F	L	S	E	F	E	0.22969	0.23369	0.23376	0.23393	0.24151	0.28250	0.01471
331	L	F	F	F	E	E	F	0.23321	0.23217	0.23515	0.23052	0.23875	0.28531	0.01472
332	L	S	F	F	E	F	E	0.22814	0.23308	0.23491	0.23619	0.24192	0.28092	0.01472
333	L	S	F	L	E	F	F	0.22489	0.23005	0.23593	0.23858	0.24410	0.28171	0.01474
334	S	S	F	S	E	F	E	0.22973	0.23040	0.23284	0.24001	0.24815	0.27444	0.01479
335	S	S	S	S	E	F	E	0.23049	0.22463	0.23423	0.24173	0.24491	0.27996	0.01486
336	S	F	L	S	E	F	E	0.22992	0.22950	0.23163	0.23955	0.24703	0.27837	0.01486
337	L	F	S	S	E	E	F	0.23034	0.23163	0.23190	0.22920	0.24807	0.28525	0.01493
338	S	S	F	F	E	F	F	0.22765	0.22946	0.23645	0.24013	0.24786	0.27496	0.01495
339	L	F	F	S	E	E	F	0.23274	0.22936	0.23280	0.23271	0.24532	0.28372	0.01498
340	L	S	F	F	E	F	F	0.22971	0.22940	0.23770	0.23777	0.23925	0.28362	0.01510
341	S	F	L	L	E	F	F	0.23105	0.23064	0.23509	0.23741	0.24578	0.27775	0.01515
342	L	F	F	F	E	E	E	0.23107	0.23238	0.23419	0.23333	0.24431	0.28289	0.01522
343	S	S	L	L	E	F	F	0.22943	0.22957	0.23357	0.24108	0.24843	0.27739	0.01545
344	L	L	L	L	F	F	E	0.23279	0.23132	0.22954	0.23935	0.24660	0.27993	0.01545
345	L	F	L	L	E	F	E	0.23108	0.23629	0.23432	0.23334	0.24278	0.28185	0.01547
346	S	F	F	F	E	E	F	0.23554	0.22665	0.23161	0.23884	0.24752	0.27972	0.01551
347	S	S	L	F	E	F	E	0.22755	0.22787	0.23619	0.24082	0.24855	0.27940	0.01559
348	S	S	L	S	E	F	E	0.23242	0.23116	0.23629	0.23872	0.24663	0.27574	0.01569
349	F	S	L	L	F	E	F	0.23283	0.23402	0.23475	0.23198	0.24370	0.28380	0.01571
350	L	L	F	F	F	E	F	0.23763	0.23301	0.23152	0.23466	0.25073	0.27358	0.01572
351	L	L	L	F	F	F	E	0.23367	0.23028	0.23228	0.23708	0.24964	0.27834	0.01575
352	S	S	L	S	E	F	F	0.23318	0.23137	0.23673	0.23715	0.24468	0.27861	0.01582
353	L	S	F	S	E	F	F	0.23135	0.22945	0.23610	0.23542	0.24233	0.28743	0.01588
354	L	S	S	L	E	F	E	0.22794	0.23148	0.23711	0.23807	0.24475	0.28301	0.01592
355	S	S	S	L	E	F	E	0.23179	0.23056	0.23736	0.23585	0.24501	0.28216	0.01599
356	L	S	L	S	E	F	F	0.22708	0.22846	0.23606	0.23633	0.24298	0.29204	0.01602
357	L	S	L	S	E	F	E	0.22796	0.22886	0.23664	0.23748	0.24325	0.28892	0.01605
358	L	F	S	F	E	E	F	0.23234	0.22857	0.23136	0.23475	0.24541	0.29073	0.01606
359	L	L	S	L	F	E	E	0.23398	0.23288	0.23592	0.23519	0.24961	0.27561	0.01606

Continued on the next page

Table 8 continued

Rank	Parameter scaling				EMA scaling			Validation loss						Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	64	128	512	1024	2048	4096	$\overline{\Delta}$
360	L	S	S	F	E	F	F	0.22994	0.22972	0.23497	0.23545	0.24632	0.28718	0.01613
361	L	L	L	S	F	F	F	0.23140	0.23233	0.23415	0.23814	0.24737	0.28023	0.01613
362	S	F	F	L	E	E	E	0.23272	0.23151	0.23342	0.23670	0.24897	0.28046	0.01616
363	S	F	L	L	E	F	E	0.23017	0.23428	0.23874	0.23815	0.24478	0.27782	0.01619
364	L	L	F	L	F	E	E	0.23763	0.23502	0.23238	0.23592	0.24822	0.27490	0.01621
365	L	L	L	S	F	F	E	0.23354	0.23446	0.23597	0.23428	0.24584	0.28002	0.01622
366	S	F	F	L	E	E	F	0.23738	0.22947	0.23147	0.23932	0.24693	0.27966	0.01624
367	L	L	S	S	F	E	F	0.23259	0.23309	0.23390	0.23638	0.24871	0.27967	0.01625
368	L	S	L	F	E	F	F	0.23235	0.22633	0.23341	0.23834	0.24493	0.29034	0.01648
369	L	F	L	L	E	F	F	0.23205	0.23211	0.23600	0.23352	0.24240	0.28964	0.01649
370	L	F	F	S	E	E	E	0.23066	0.23357	0.23471	0.23170	0.24743	0.28777	0.01650
371	L	L	S	S	F	F	E	0.23239	0.23183	0.23175	0.23811	0.25134	0.28058	0.01653
372	L	L	F	S	F	F	E	0.23482	0.23368	0.23312	0.24331	0.24679	0.27427	0.01653
373	L	L	F	L	F	F	F	0.23507	0.23372	0.23520	0.23511	0.25074	0.27644	0.01658
374	S	S	F	F	E	E	F	0.23148	0.23241	0.23064	0.24049	0.24864	0.28298	0.01664
375	S	S	F	S	E	E	F	0.23035	0.23006	0.23642	0.23980	0.24794	0.28215	0.01665
376	L	S	S	S	E	F	F	0.22992	0.22837	0.23863	0.23586	0.24477	0.28936	0.01668
377	L	L	S	L	F	F	F	0.23719	0.22986	0.23585	0.24006	0.25041	0.27359	0.01669
378	L	L	F	S	F	E	F	0.23898	0.23134	0.23435	0.23762	0.24902	0.27566	0.01669
379	L	F	S	S	E	E	E	0.23089	0.23088	0.22972	0.23619	0.24576	0.29356	0.01670
380	L	L	F	L	F	F	E	0.23312	0.23467	0.23451	0.24099	0.24926	0.27466	0.01673
381	S	F	S	S	E	E	F	0.23166	0.23084	0.23674	0.23702	0.24654	0.28452	0.01675
382	L	S	S	S	E	F	E	0.23003	0.22639	0.23698	0.23911	0.24699	0.28785	0.01675
383	S	L	F	S	E	F	E	0.22913	0.22614	0.23623	0.24030	0.24476	0.29081	0.01676
384	L	L	F	F	F	F	F	0.23359	0.23506	0.23230	0.23938	0.25156	0.27564	0.01679
385	L	L	L	F	F	F	F	0.23237	0.23250	0.23617	0.23792	0.25217	0.27659	0.01682
386	L	F	S	L	E	E	E	0.23209	0.23014	0.23249	0.23722	0.24642	0.28979	0.01689
387	S	L	F	F	E	F	E	0.23043	0.22414	0.23585	0.24049	0.24999	0.28728	0.01690
388	L	S	L	F	E	F	E	0.22908	0.23157	0.23504	0.23896	0.24455	0.28917	0.01693
389	S	F	F	S	E	E	F	0.23513	0.23205	0.23521	0.23750	0.24855	0.28016	0.01697
390	L	L	F	L	F	E	F	0.23549	0.23489	0.23227	0.23878	0.24726	0.28010	0.01700
391	L	L	S	L	F	F	E	0.23188	0.23256	0.23294	0.23711	0.25081	0.28381	0.01705
392	L	F	S	F	E	E	E	0.22876	0.23259	0.23115	0.23623	0.24973	0.29074	0.01706
393	S	S	L	L	E	F	E	0.23039	0.23257	0.24007	0.23900	0.24865	0.27852	0.01707
394	L	F	S	L	E	E	F	0.23587	0.23160	0.23223	0.23660	0.24420	0.28915	0.01714
395	L	S	L	L	E	F	E	0.22882	0.22990	0.23432	0.23664	0.24407	0.29595	0.01715
396	S	L	F	S	E	F	F	0.23028	0.22940	0.23688	0.23741	0.25059	0.28565	0.01723
397	L	F	F	L	E	E	F	0.23504	0.23473	0.23558	0.23676	0.24439	0.28375	0.01724
398	L	L	F	S	F	E	E	0.23564	0.23222	0.23618	0.23723	0.25000	0.27901	0.01725
399	S	F	S	L	E	E	F	0.23158	0.23308	0.23425	0.23913	0.24878	0.28416	0.01736
400	S	F	F	F	E	E	E	0.23343	0.23161	0.23872	0.23877	0.24857	0.27993	0.01737
401	L	S	L	L	E	F	F	0.22999	0.22999	0.23297	0.23273	0.24841	0.29696	0.01737
402	L	L	F	F	F	E	E	0.23632	0.23586	0.23477	0.23751	0.24723	0.27938	0.01738
403	L	L	L	L	F	F	F	0.23244	0.23385	0.23133	0.24062	0.24637	0.28663	0.01741
404	S	S	F	L	E	E	E	0.23160	0.22846	0.23633	0.23857	0.25101	0.28660	0.01763
405	S	F	F	S	E	E	E	0.23181	0.23594	0.23477	0.24097	0.24950	0.27987	0.01768
406	L	L	S	S	F	F	F	0.23831	0.23320	0.23057	0.23862	0.25244	0.28057	0.01782
407	S	L	F	L	E	F	E	0.22551	0.22969	0.23807	0.24115	0.25163	0.28790	0.01786
408	L	L	S	F	F	E	F	0.23692	0.23418	0.23386	0.23448	0.25199	0.28256	0.01786
409	L	S	F	F	E	E	F	0.23161	0.23133	0.23330	0.23360	0.24849	0.29572	0.01787
410	S	L	F	F	E	F	F	0.23093	0.22954	0.23686	0.24052	0.24902	0.28720	0.01788
411	L	S	F	L	E	E	F	0.23119	0.23314	0.23256	0.23520	0.24717	0.29493	0.01789
412	S	S	F	F	E	E	E	0.22992	0.23226	0.23563	0.24008	0.25062	0.28617	0.01798
413	L	L	S	F	F	E	E	0.23765	0.23130	0.23387	0.23849	0.25095	0.28255	0.01800
414	L	L	S	F	F	F	F	0.23358	0.23449	0.23220	0.24030	0.25140	0.28284	0.01800
415	L	L	S	F	F	F	E	0.23509	0.23167	0.23484	0.23667	0.25117	0.28545	0.01801
416	S	L	S	S	E	F	F	0.22652	0.23017	0.23722	0.24102	0.24983	0.29034	0.01805
417	S	F	S	F	E	E	F	0.23447	0.23036	0.23654	0.23799	0.24964	0.28611	0.01805
418	L	L	S	L	F	E	F	0.23538	0.23518	0.23066	0.23995	0.25128	0.28274	0.01806
419	S	L	S	F	E	F	F	0.22929	0.22981	0.23541	0.23808	0.25031	0.29250	0.01810
420	S	S	S	S	E	E	F	0.23146	0.22952	0.23633	0.24215	0.24889	0.28750	0.01817
421	S	S	S	F	E	E	F	0.23069	0.22937	0.23373	0.23684	0.25002	0.29543	0.01821
422	S	L	L	F	E	F	E	0.22764	0.22922	0.23783	0.24189	0.24982	0.28999	0.01826
423	S	L	L	S	E	F	F	0.22611	0.22858	0.23648	0.24187	0.24882	0.29462	0.01828
424	L	S	F	F	E	E	E	0.23541	0.22976	0.23256	0.23605	0.25091	0.29193	0.01830
425	L	L	L	L	F	E	F	0.23646	0.23626	0.23121	0.23229	0.25687	0.28421	0.01841
426	L	L	L	F	F	E	F	0.23603	0.23285	0.23407	0.23944	0.25213	0.28321	0.01849
427	S	F	S	L	E	E	E	0.23290	0.22984	0.23444	0.23777	0.25052	0.29271	0.01856
428	L	L	F	S	F	F	F	0.23507	0.23387	0.23707	0.23985	0.25075	0.28171	0.01858
429	S	L	S	L	E	F	E	0.22518	0.22956	0.23819	0.24227	0.24957	0.29360	0.01860
430	L	L	F	F	F	F	E	0.23725	0.23342	0.23736	0.23735	0.24975	0.28359	0.01865
431	L	S	S	F	E	F	E	0.23132	0.23379	0.23466	0.23829	0.24736	0.29357	0.01870
432	S	L	L	F	E	F	F	0.23047	0.22822	0.23650	0.24013	0.25422	0.28995	0.01878

Continued on the next page

Table 8 continued

Rank	Parameter scaling				EMA scaling			Validation loss						Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	64	128	512	1024	2048	4096	$\overline{\Delta}$
433	L	L	S	S	F	E	E	0.23811	0.23517	0.23374	0.23807	0.25006	0.28496	0.01888
434	S	L	S	L	E	F	F	0.22842	0.22958	0.23679	0.24229	0.25199	0.29112	0.01890
435	L	L	L	F	F	E	E	0.23504	0.23679	0.23428	0.23771	0.25322	0.28324	0.01891
436	S	S	F	L	E	E	F	0.23501	0.23077	0.23456	0.24414	0.25037	0.28546	0.01892
437	S	F	S	F	E	E	E	0.23263	0.23404	0.23487	0.23926	0.24825	0.29128	0.01892
438	S	S	S	L	E	E	E	0.22953	0.23204	0.23365	0.23997	0.25096	0.29475	0.01902
439	L	S	F	S	E	E	F	0.23004	0.23072	0.23487	0.23808	0.24879	0.29865	0.01905
440	L	S	S	S	E	E	F	0.22777	0.23323	0.23468	0.23730	0.25111	0.29747	0.01913
441	S	S	S	L	E	E	F	0.23196	0.23095	0.23605	0.24048	0.25058	0.29164	0.01914
442	L	L	L	S	F	E	F	0.23402	0.23548	0.23166	0.23621	0.25447	0.28984	0.01915
443	S	L	F	L	E	F	F	0.23233	0.22872	0.23866	0.24246	0.25279	0.28693	0.01918
444	S	L	S	S	E	F	E	0.22813	0.23030	0.23760	0.24268	0.25231	0.29095	0.01919
445	L	F	L	S	E	E	E	0.22934	0.23114	0.22966	0.23506	0.24395	0.31291	0.01921
446	F	S	F	L	E	F	F	0.23029	0.23039	0.23697	0.23456	0.25007	0.29980	0.01921
447	S	L	L	L	E	F	E	0.22823	0.22752	0.23566	0.24406	0.25176	0.29495	0.01922
448	L	F	L	F	E	E	E	0.23190	0.23202	0.22996	0.23484	0.24646	0.30721	0.01926
449	S	S	F	S	E	E	E	0.23246	0.23400	0.23532	0.23985	0.25389	0.28738	0.01935
450	L	S	F	S	E	E	E	0.23180	0.23220	0.23223	0.23868	0.24998	0.29824	0.01939
451	L	S	F	L	E	E	E	0.22867	0.23118	0.23703	0.23688	0.25103	0.29860	0.01943
452	S	L	L	L	E	F	F	0.22629	0.22869	0.23868	0.24204	0.25117	0.29663	0.01945
453	L	S	S	S	E	E	E	0.22875	0.22709	0.23236	0.23713	0.25191	0.30636	0.01947
454	F	S	F	L	E	F	E	0.22826	0.22920	0.23515	0.24026	0.25294	0.29954	0.01976
455	F	S	F	F	E	F	E	0.22857	0.22761	0.23572	0.23834	0.25311	0.30203	0.01976
456	L	L	L	L	F	E	E	0.23711	0.23309	0.23037	0.23769	0.25736	0.28981	0.01977
457	S	S	S	S	E	E	E	0.23090	0.23077	0.23548	0.24235	0.24878	0.29778	0.01988
458	S	L	F	S	E	E	F	0.23075	0.22666	0.23486	0.24138	0.25468	0.29789	0.01990
459	S	L	S	F	E	F	E	0.22809	0.23295	0.23680	0.23932	0.25488	0.29425	0.01992
460	S	L	L	S	E	F	E	0.23313	0.23081	0.23854	0.24321	0.24977	0.29084	0.01992
461	L	S	S	L	E	E	F	0.22819	0.22943	0.23574	0.23757	0.25002	0.30619	0.02005
462	S	F	L	F	E	E	E	0.23317	0.23092	0.23147	0.23577	0.25246	0.30358	0.02009
463	L	F	L	S	E	E	F	0.23653	0.23017	0.23084	0.23421	0.24710	0.30859	0.02011
464	L	S	S	F	E	E	F	0.23045	0.22969	0.23487	0.24000	0.25254	0.29990	0.02011
465	L	F	L	L	E	E	F	0.22616	0.23249	0.23004	0.23717	0.25119	0.31080	0.02017
466	S	S	L	F	E	E	F	0.23312	0.23232	0.23512	0.24017	0.24998	0.29760	0.02025
467	S	F	S	S	E	E	E	0.23386	0.23413	0.23797	0.23895	0.25172	0.29174	0.02026
468	S	F	L	S	E	E	F	0.23475	0.23246	0.23151	0.24219	0.24965	0.29837	0.02035
469	S	S	S	F	E	E	E	0.23166	0.23355	0.23797	0.23916	0.25178	0.29487	0.02036
470	F	F	F	S	E	F	E	0.23366	0.23089	0.23381	0.23956	0.25129	0.29989	0.02038
471	L	L	L	S	F	E	E	0.23593	0.23655	0.23227	0.23614	0.25506	0.29319	0.02039
472	F	S	F	S	E	F	F	0.22923	0.22942	0.23642	0.24127	0.25261	0.30036	0.02042
473	S	L	S	S	E	E	F	0.22936	0.22814	0.23532	0.24536	0.25544	0.29592	0.02045
474	F	S	F	F	E	F	F	0.22883	0.22843	0.23472	0.24059	0.25523	0.30227	0.02054
475	F	L	F	L	E	F	F	0.22735	0.22884	0.23676	0.23954	0.25543	0.30234	0.02057
476	S	L	F	L	E	E	F	0.22705	0.23202	0.23534	0.24266	0.25528	0.29797	0.02059
477	F	F	F	F	E	F	F	0.23574	0.22964	0.23578	0.23950	0.25239	0.29739	0.02060
478	F	L	F	F	E	F	E	0.22951	0.22811	0.23544	0.24160	0.25422	0.30186	0.02066
479	L	S	S	F	E	E	E	0.23379	0.23013	0.23370	0.23647	0.25340	0.30411	0.02080
480	S	L	S	F	E	E	F	0.23105	0.22834	0.23896	0.24504	0.25338	0.29486	0.02080
481	F	L	F	S	E	F	F	0.22984	0.23054	0.23640	0.23813	0.25532	0.30197	0.02090
482	F	L	F	F	E	F	F	0.23023	0.22956	0.23957	0.23977	0.25429	0.29880	0.02090
483	S	L	F	F	E	E	F	0.22949	0.23050	0.23730	0.24238	0.25541	0.29742	0.02095
484	F	S	S	F	E	F	E	0.22981	0.22654	0.23717	0.24089	0.25421	0.30447	0.02105
485	L	F	L	F	E	E	F	0.23026	0.23548	0.23093	0.23509	0.24957	0.31191	0.02107
486	F	L	F	S	E	F	E	0.22807	0.22826	0.23615	0.24283	0.25518	0.30304	0.02112
487	F	F	S	F	E	F	F	0.23162	0.22931	0.23684	0.24164	0.25366	0.30054	0.02113
488	F	S	F	S	E	F	E	0.23249	0.22911	0.23620	0.24075	0.25416	0.30119	0.02118
489	F	L	F	L	E	F	E	0.23016	0.22725	0.23756	0.24100	0.25328	0.30501	0.02124
490	S	S	L	S	E	E	F	0.23041	0.23516	0.23387	0.24035	0.25310	0.30141	0.02125
491	F	L	S	L	E	F	F	0.22859	0.22842	0.23698	0.24277	0.25236	0.30549	0.02130
492	F	S	S	L	E	F	F	0.22967	0.22883	0.23896	0.23887	0.25571	0.30257	0.02130
493	L	S	L	S	E	E	E	0.23068	0.23037	0.22989	0.23474	0.25238	0.31670	0.02132
494	F	F	F	F	E	F	E	0.23519	0.23135	0.23561	0.23694	0.25279	0.30309	0.02136
495	S	L	F	S	E	E	E	0.23030	0.22796	0.23490	0.24703	0.25897	0.29614	0.02141
496	L	S	L	S	E	E	F	0.23006	0.23340	0.23271	0.23548	0.25318	0.31059	0.02144
497	S	F	L	F	E	E	F	0.23218	0.23071	0.23381	0.23994	0.25096	0.30810	0.02148
498	F	F	F	L	E	F	F	0.23630	0.23052	0.23556	0.23969	0.25508	0.29882	0.02153
499	S	L	F	L	E	E	E	0.22969	0.23120	0.23330	0.24308	0.25431	0.30516	0.02166
500	F	F	S	L	E	F	F	0.23410	0.23142	0.23577	0.24068	0.25303	0.30200	0.02170
501	S	L	S	L	E	E	E	0.23017	0.23443	0.23298	0.24134	0.25437	0.30412	0.02177
502	S	S	L	F	E	E	E	0.23382	0.23018	0.23661	0.23931	0.25349	0.30417	0.02180
503	F	F	F	S	E	F	F	0.23433	0.23351	0.23608	0.24001	0.25379	0.29998	0.02181
504	F	S	L	S	E	F	F	0.22758	0.23066	0.23884	0.24107	0.25435	0.30592	0.02193
505	F	S	S	S	E	F	E	0.22996	0.22959	0.23624	0.24255	0.25311	0.30702	0.02194

Continued on the next page

Table 8 continued

Rank	Parameter scaling				EMA scaling			Validation loss						Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	64	128	512	1024	2048	4096	$\overline{\Delta}$
506	F	S	S	S	E	F	F	0.23349	0.23011	0.23365	0.24129	0.25579	0.30429	0.02197
507	S	F	L	S	E	E	E	0.23187	0.23198	0.23379	0.23692	0.25280	0.31155	0.02202
508	S	L	F	F	E	E	E	0.23340	0.23162	0.23800	0.24451	0.25473	0.29667	0.02202
509	F	L	L	S	E	F	F	0.22760	0.22755	0.23836	0.24368	0.25470	0.30716	0.02204
510	F	S	S	F	E	F	F	0.23049	0.23134	0.23860	0.24331	0.25579	0.29971	0.02207
511	F	L	S	S	E	F	F	0.22877	0.22834	0.23756	0.24276	0.25659	0.30528	0.02208
512	F	S	F	S	E	E	F	0.22951	0.22952	0.23439	0.23953	0.25526	0.31168	0.02218
513	F	F	S	S	E	F	F	0.23745	0.23309	0.23491	0.24117	0.25368	0.30061	0.02235
514	F	F	S	S	E	F	E	0.23210	0.23172	0.23563	0.24211	0.25474	0.30475	0.02237
515	F	L	S	F	E	F	F	0.22942	0.23412	0.23823	0.24150	0.25452	0.30345	0.02240
516	L	S	L	F	E	E	F	0.23145	0.23152	0.23174	0.23591	0.25143	0.31981	0.02251
517	S	L	L	S	E	E	F	0.23048	0.23212	0.23534	0.23982	0.25617	0.30883	0.02266
518	F	F	F	L	E	F	E	0.23681	0.23196	0.23788	0.23996	0.25533	0.30084	0.02266
519	F	F	L	L	E	F	E	0.23133	0.22705	0.23305	0.24424	0.25579	0.31156	0.02270
520	F	L	S	S	E	F	E	0.22930	0.23061	0.23736	0.24504	0.25372	0.30706	0.02271
521	F	L	L	F	E	F	F	0.22814	0.23112	0.23919	0.24343	0.25757	0.30387	0.02275
522	F	F	S	F	E	F	E	0.23765	0.23201	0.23728	0.23695	0.25613	0.30354	0.02279
523	S	S	L	S	E	E	E	0.23020	0.23273	0.23693	0.24226	0.25459	0.30703	0.02282
524	S	L	S	S	E	E	E	0.23089	0.22904	0.23535	0.24288	0.25726	0.30911	0.02295
525	F	L	S	L	E	F	E	0.22707	0.23447	0.23834	0.24402	0.25705	0.30361	0.02296
526	S	L	S	F	E	E	E	0.23269	0.23074	0.23719	0.24012	0.25691	0.30726	0.02302
527	F	L	S	F	E	F	E	0.23261	0.22798	0.23924	0.24300	0.25606	0.30606	0.02302
528	F	S	L	F	E	F	F	0.23003	0.23110	0.23920	0.24530	0.25530	0.30420	0.02305
529	F	F	L	S	E	F	F	0.23503	0.22841	0.23929	0.23865	0.25605	0.30777	0.02306
530	L	S	L	F	E	E	E	0.23173	0.23123	0.23393	0.23614	0.25765	0.31493	0.02313
531	F	S	S	L	E	F	E	0.22969	0.23337	0.23893	0.24104	0.25777	0.30503	0.02317
532	F	L	L	F	E	F	E	0.22908	0.23218	0.23748	0.24302	0.25733	0.30692	0.02320
533	S	F	L	L	E	E	F	0.23290	0.23211	0.23898	0.23615	0.25220	0.31377	0.02322
534	F	L	F	L	E	E	F	0.23167	0.22851	0.23532	0.24054	0.25580	0.31470	0.02329
535	S	L	S	L	E	E	F	0.22986	0.23057	0.23835	0.24194	0.25445	0.31140	0.02330
536	F	S	L	F	E	F	E	0.23189	0.22805	0.23876	0.24319	0.25624	0.30862	0.02332
537	S	F	L	L	E	E	E	0.23568	0.23359	0.23785	0.23986	0.25268	0.30737	0.02337
538	F	L	L	L	E	F	E	0.22787	0.22911	0.23647	0.24322	0.25779	0.31339	0.02350
539	F	F	F	S	E	E	E	0.23366	0.23133	0.23474	0.23880	0.25540	0.31444	0.02359
540	L	S	S	L	E	E	E	0.23480	0.23664	0.23344	0.23713	0.25523	0.31124	0.02361
541	L	F	L	L	E	E	E	0.23330	0.22921	0.23492	0.23426	0.25209	0.32484	0.02363
542	F	F	F	S	E	E	F	0.23657	0.23504	0.23516	0.23768	0.25356	0.31127	0.02375
543	F	L	F	F	E	E	F	0.22939	0.23309	0.23775	0.24112	0.25557	0.31305	0.02386
544	F	S	L	S	E	F	E	0.23247	0.22927	0.23678	0.24421	0.25724	0.31007	0.02387
545	F	S	F	L	E	E	F	0.23588	0.23292	0.23358	0.24315	0.25449	0.31053	0.02396
546	F	S	F	F	E	E	F	0.23154	0.23338	0.24039	0.24196	0.25526	0.30813	0.02397
547	F	F	S	L	E	F	E	0.23365	0.23770	0.23698	0.24083	0.25522	0.30663	0.02403
548	F	F	F	F	E	E	E	0.23326	0.23394	0.23236	0.24322	0.25598	0.31311	0.02418
549	F	L	L	S	E	F	E	0.23021	0.22874	0.23953	0.24759	0.26048	0.30584	0.02426
550	S	S	L	L	E	E	E	0.23120	0.22902	0.23319	0.24134	0.25526	0.32249	0.02428
551	F	F	F	L	E	E	F	0.23325	0.23252	0.23808	0.24032	0.25800	0.31078	0.02436
552	F	F	L	S	E	F	E	0.23402	0.23337	0.23487	0.24449	0.25768	0.30858	0.02436
553	F	F	S	S	E	E	F	0.23448	0.23028	0.23470	0.24005	0.25633	0.31774	0.02446
554	F	L	F	S	E	E	F	0.22863	0.22951	0.23681	0.24418	0.25721	0.31734	0.02448
555	F	L	F	S	E	E	E	0.23337	0.23168	0.23711	0.24026	0.25813	0.31348	0.02454
556	F	S	F	S	E	E	E	0.23452	0.23174	0.23657	0.24159	0.25538	0.31432	0.02455
557	F	L	L	L	E	F	F	0.22864	0.23285	0.23749	0.24323	0.25947	0.31248	0.02456
558	F	L	F	L	E	E	E	0.23118	0.22830	0.23788	0.24078	0.25873	0.31734	0.02457
559	F	S	F	F	E	E	E	0.23312	0.23100	0.23612	0.24280	0.25656	0.31477	0.02459
560	F	F	L	F	E	F	F	0.23669	0.23160	0.23892	0.24222	0.25806	0.30701	0.02462
561	F	S	F	L	E	E	E	0.23277	0.23279	0.23535	0.24052	0.25831	0.31500	0.02466
562	S	S	L	L	E	E	F	0.22935	0.23287	0.23854	0.23873	0.25653	0.31967	0.02481
563	F	L	F	F	E	E	E	0.22968	0.23159	0.23683	0.24538	0.25601	0.31644	0.02485
564	L	S	L	L	E	E	F	0.23237	0.23307	0.23486	0.23807	0.25260	0.32496	0.02485
565	F	F	F	L	E	E	E	0.23435	0.23160	0.23671	0.23947	0.25951	0.31505	0.02498
566	L	L	F	L	E	F	F	0.23004	0.22991	0.23500	0.23634	0.25652	0.32905	0.02501
567	F	F	L	L	E	F	F	0.23161	0.23201	0.23677	0.24764	0.25640	0.31258	0.02503
568	F	F	L	F	E	F	E	0.23580	0.23384	0.23393	0.24333	0.25873	0.31152	0.02506
569	S	L	L	F	E	E	F	0.23163	0.23317	0.23572	0.24108	0.25501	0.32204	0.02531
570	L	S	L	L	E	E	E	0.23072	0.22919	0.23522	0.23791	0.25849	0.32877	0.02558
571	F	F	F	F	E	E	F	0.23926	0.23420	0.23604	0.24294	0.25579	0.31243	0.02564
572	S	L	L	L	E	E	F	0.23035	0.22812	0.23317	0.24175	0.25613	0.33137	0.02568
573	F	S	L	L	E	F	F	0.23117	0.23119	0.24104	0.24582	0.25797	0.31393	0.02572
574	F	L	S	L	E	E	F	0.22955	0.23041	0.23881	0.24600	0.25933	0.31760	0.02581
575	F	S	L	L	E	F	E	0.23134	0.23225	0.23868	0.24545	0.25916	0.31495	0.02584
576	S	L	L	S	E	E	E	0.23146	0.22858	0.23575	0.24189	0.26021	0.32476	0.02597
577	F	S	S	L	E	E	F	0.23189	0.22932	0.23854	0.24122	0.26034	0.32170	0.02603
578	S	L	L	L	E	E	E	0.23267	0.23078	0.23634	0.24709	0.26036	0.31661	0.02618

Continued on the next page

Table 8 continued

Rank	Parameter scaling				EMA scaling			Validation loss						Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	64	128	512	1024	2048	4096	$\overline{\Delta}$
579	F	L	S	F	E	E	F	0.22866	0.23030	0.23922	0.24305	0.26105	0.32347	0.02649
580	L	L	L	S	E	F	E	0.22598	0.23101	0.23564	0.23950	0.25450	0.33928	0.02651
581	S	L	L	F	E	E	E	0.23560	0.23126	0.23938	0.24329	0.25839	0.31850	0.02660
582	F	S	S	F	E	E	E	0.22956	0.22951	0.23723	0.24179	0.26136	0.32713	0.02663
583	F	F	S	S	E	E	E	0.23204	0.23307	0.23623	0.24224	0.25905	0.32404	0.02664
584	F	L	S	L	E	E	E	0.22925	0.23188	0.23916	0.24469	0.26050	0.32189	0.02676
585	F	S	S	S	E	E	F	0.23413	0.23291	0.23543	0.24002	0.25912	0.32601	0.02680
586	F	L	S	S	E	E	F	0.22716	0.23292	0.23908	0.24479	0.25995	0.32473	0.02697
587	F	S	S	L	E	E	E	0.23175	0.23265	0.24040	0.24247	0.26081	0.32122	0.02708
588	F	L	S	S	E	E	E	0.22756	0.23067	0.23690	0.24742	0.25958	0.32737	0.02711
589	F	L	L	F	E	E	F	0.23063	0.23141	0.23517	0.24359	0.26163	0.32730	0.02715
590	F	F	S	F	E	E	F	0.23562	0.23461	0.23479	0.24103	0.25984	0.32389	0.02716
591	L	L	L	S	E	F	F	0.22708	0.22982	0.23638	0.24430	0.25878	0.33395	0.02725
592	F	L	S	F	E	E	E	0.22961	0.23335	0.23765	0.24387	0.25938	0.32699	0.02734
593	L	L	S	S	E	F	F	0.23161	0.22809	0.23769	0.24169	0.26472	0.32838	0.02756
594	F	F	S	L	E	E	F	0.23414	0.23223	0.23739	0.24634	0.25874	0.32420	0.02771
595	L	L	L	L	E	F	F	0.22740	0.23046	0.23480	0.24163	0.25963	0.33925	0.02773
596	L	L	F	S	E	F	E	0.22940	0.22896	0.23349	0.24200	0.25609	0.34355	0.02778
597	L	L	L	L	E	F	E	0.22457	0.22623	0.23259	0.23957	0.26393	0.34691	0.02783
598	L	L	L	F	E	F	F	0.22983	0.23200	0.23494	0.24391	0.25436	0.33890	0.02786
599	F	S	L	F	E	E	F	0.23088	0.23469	0.23912	0.24386	0.26020	0.32545	0.02790
600	F	F	L	F	E	E	E	0.23714	0.23240	0.23431	0.24091	0.26716	0.32267	0.02796
601	L	L	F	F	E	F	E	0.22889	0.22672	0.23800	0.24073	0.26039	0.34045	0.02806
602	L	L	F	L	E	F	E	0.22775	0.22715	0.23783	0.23994	0.26330	0.33988	0.02817
603	F	S	S	F	E	E	F	0.23483	0.23633	0.23722	0.24422	0.25912	0.32438	0.02822
604	F	S	S	S	E	E	E	0.23452	0.23489	0.23935	0.24348	0.26021	0.32444	0.02834
605	L	L	F	F	E	F	F	0.23035	0.23374	0.23703	0.24167	0.25661	0.33767	0.02838
606	F	F	S	L	E	E	E	0.23566	0.23604	0.23620	0.24044	0.26074	0.32809	0.02839
607	F	L	L	S	E	E	F	0.23163	0.23196	0.23398	0.24442	0.26184	0.33405	0.02851
608	F	S	L	S	E	E	F	0.22997	0.23552	0.23541	0.24220	0.26117	0.33394	0.02857
609	L	L	S	S	E	F	E	0.22941	0.22881	0.23494	0.23910	0.26484	0.34160	0.02865
610	F	F	S	F	E	E	E	0.23654	0.23620	0.23940	0.24285	0.25874	0.32518	0.02868
611	L	L	S	L	E	F	F	0.22848	0.22771	0.23370	0.24216	0.26395	0.34520	0.02906
612	L	L	L	F	E	F	E	0.22816	0.23178	0.23834	0.24103	0.25870	0.34346	0.02911
613	F	L	L	S	E	E	E	0.23227	0.22972	0.23762	0.24479	0.26351	0.33430	0.02923
614	L	L	S	F	E	F	E	0.22873	0.22919	0.23421	0.24286	0.26593	0.34265	0.02946
615	L	L	F	S	E	F	F	0.22990	0.23057	0.23752	0.23909	0.26090	0.34705	0.02970
616	F	F	L	F	E	E	F	0.23805	0.23345	0.23664	0.24594	0.26034	0.33072	0.02972
617	L	L	S	L	E	F	E	0.23255	0.22913	0.23441	0.24400	0.26012	0.34685	0.03004
618	L	L	S	F	E	F	F	0.23075	0.22856	0.23547	0.24384	0.26332	0.34575	0.03015
619	L	L	S	F	E	E	F	0.23104	0.23111	0.23489	0.24181	0.26660	0.34263	0.03021
620	F	L	L	L	E	E	F	0.23356	0.22830	0.23745	0.24076	0.26499	0.34588	0.03069
621	F	S	L	F	E	E	E	0.23435	0.23580	0.24065	0.24306	0.26510	0.33381	0.03099
622	L	L	S	L	E	E	F	0.22808	0.22935	0.23278	0.23977	0.26941	0.35424	0.03114
623	F	F	L	S	E	E	F	0.24099	0.23563	0.23542	0.24105	0.26497	0.33560	0.03114
624	F	S	L	S	E	E	E	0.23291	0.23275	0.23530	0.24529	0.26357	0.34472	0.03129
625	F	L	L	F	E	E	E	0.22901	0.23101	0.23939	0.24600	0.26441	0.34494	0.03133
626	F	F	L	L	E	E	F	0.23175	0.23327	0.23512	0.24206	0.26468	0.34845	0.03142
627	F	S	L	L	E	E	F	0.23178	0.23388	0.23869	0.23934	0.26560	0.34736	0.03164
628	L	L	F	F	E	E	E	0.23003	0.23387	0.23617	0.23894	0.26731	0.35253	0.03200
629	L	L	F	L	E	E	E	0.23253	0.22763	0.23697	0.23992	0.26852	0.35341	0.03203
630	F	F	L	S	E	E	E	0.23699	0.23474	0.23606	0.24230	0.26409	0.34508	0.03207
631	L	L	S	S	E	E	E	0.22947	0.22845	0.23135	0.24045	0.26408	0.36749	0.03241
632	F	L	L	L	E	E	E	0.23360	0.23405	0.23653	0.24415	0.26663	0.34662	0.03246
633	L	L	S	L	E	E	E	0.23133	0.22730	0.23553	0.24126	0.26637	0.36042	0.03257
634	L	L	F	S	E	E	F	0.22934	0.22856	0.23790	0.24161	0.26241	0.36268	0.03262
635	L	L	S	F	E	E	E	0.23061	0.23381	0.23510	0.23832	0.26763	0.35777	0.03274
636	L	L	F	L	E	E	F	0.23039	0.22996	0.23335	0.24044	0.26601	0.36354	0.03281
637	F	S	L	L	E	E	E	0.23515	0.23438	0.23766	0.24412	0.26546	0.34730	0.03288
638	L	L	S	S	E	E	F	0.23259	0.23239	0.23449	0.24162	0.26818	0.35728	0.03329
639	F	F	L	L	E	E	E	0.23864	0.23056	0.23742	0.24281	0.26696	0.35118	0.03346
640	L	L	F	F	E	E	F	0.23465	0.23224	0.23612	0.23948	0.26508	0.36117	0.03366
641	L	L	F	S	E	E	E	0.23158	0.22954	0.23805	0.23938	0.26503	0.36720	0.03399
642	L	L	L	F	E	E	F	0.22992	0.22951	0.23297	0.23897	0.26847	0.37493	0.03466
643	L	L	L	F	E	E	E	0.23170	0.23137	0.23471	0.24273	0.26307	0.37171	0.03475
644	L	L	L	S	E	E	F	0.22655	0.23226	0.23194	0.24024	0.27391	0.37559	0.03561
645	L	L	L	S	E	E	E	0.22850	0.23071	0.23450	0.23897	0.27188	0.37598	0.03562
646	L	L	L	L	E	E	E	0.23353	0.23158	0.23135	0.24335	0.27399	0.37266	0.03661
647	L	L	L	L	E	E	F	0.23318	0.23046	0.23282	0.24366	0.26786	0.38987	0.03851
648	S	L	L	L	F	E	E	0.22924	0.23643	0.23497	0.23138	0.24860	0.78509	0.09982

Table 9: Terminal validation losses for the 216 original language-model rules and their auxiliary weight-decay extensions. Each group contains fixed (F), square-root (S), and linear (L) auxiliary decay; E denotes retention preservation over the complete run. Groups retain the original ranking by mean BSZ regret, the equally weighted mean over the four target batch-size choices of the loss gap to the original grid minimum at the same batch size. Dashes mark unmeasured losses and unavailable four-batch ranks and BSZ regrets. Bold marks the common rule selected from the original grid.

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
1	**F**	**L**	**F**	**F**	**F**	**F**	**E**	**3.26969**	**3.28192**	**3.31183**	**3.37977**	**0.00151**
–	F	L	F	S	F	F	E	–	–	3.30815	3.37707	–
–	F	L	F	L	F	F	E	–	–	3.30896	3.37751	–
2	F	L	F	F	F	F	F	3.26952	3.28214	3.31272	3.38241	0.00240
–	F	L	F	S	F	F	F	–	–	3.30956	3.38014	–
–	F	L	F	L	F	F	F	–	–	3.30969	3.38111	–
3	F	S	F	F	F	F	E	3.26862	3.28334	3.31700	3.38662	0.00460
–	F	S	F	S	F	F	E	–	–	3.31425	3.38392	–
–	F	S	F	L	F	F	E	–	–	3.31591	3.38604	–
4	S	S	F	F	F	F	E	3.26769	3.28087	3.31636	3.39396	0.00543
–	S	S	F	S	F	F	E	–	–	3.31179	3.38877	–
–	S	S	F	L	F	F	E	–	–	3.31247	3.38825	–
5	F	S	F	F	F	F	F	3.26869	3.28377	3.31857	3.39027	0.00603
–	F	S	F	S	F	F	F	–	–	3.31557	3.38677	–
–	F	S	F	L	F	F	F	–	–	3.31686	3.39062	–
6	S	S	F	F	F	F	F	3.26776	3.28045	3.31698	3.39755	0.00639
–	S	S	F	S	F	F	F	–	–	3.31232	3.39053	–
–	S	S	F	L	F	F	F	–	–	3.31342	3.39234	–
7	F	L	S	F	F	F	E	3.26763	3.28119	3.31645	3.41001	0.00953
–	F	L	S	S	F	F	E	–	–	3.31747	3.40957	–
–	F	L	S	L	F	F	E	–	–	3.32578	3.42959	–
8	F	L	S	F	F	F	F	3.26709	3.28046	3.31689	3.41118	0.00961
–	F	L	S	S	F	F	F	–	–	3.31914	3.41088	–
–	F	L	S	L	F	F	F	–	–	3.32777	3.43302	–
9	S	S	S	F	F	F	E	3.26603	3.28004	3.32522	3.41423	0.01209
–	S	S	S	S	F	F	E	–	–	3.32329	3.41132	–
–	S	S	S	L	F	F	E	–	–	3.33074	3.42718	–
10	F	F	F	F	F	F	E	3.27030	3.28906	3.32787	3.40266	0.01318
–	F	F	F	S	F	F	E	–	–	3.32483	3.39844	–
–	F	F	F	L	F	F	E	–	–	3.32748	3.40226	–
11	S	F	F	F	F	F	E	3.26707	3.28109	3.32307	3.42290	0.01424
–	S	F	F	S	F	F	E	–	–	3.31890	3.41903	–
–	S	F	F	L	F	F	E	–	–	3.32048	3.41402	–
12	F	F	F	F	F	F	F	3.27018	3.28989	3.32961	3.40498	0.01437
–	F	F	F	S	F	F	F	–	–	3.32646	3.40165	–
–	F	F	F	L	F	F	F	–	–	3.32906	3.40586	–
13	S	S	S	F	F	F	F	3.26684	3.28116	3.32571	3.42156	0.01452
–	S	S	S	S	F	F	F	–	–	3.32397	3.41626	–
–	S	S	S	L	F	F	F	–	–	3.33164	3.43217	–
14	F	S	S	F	F	F	E	3.26685	3.28379	3.32347	3.42222	0.01479
–	F	S	S	S	F	F	E	–	–	3.32510	3.42303	–
–	F	S	S	L	F	F	E	–	–	3.33479	3.44695	–
15	F	S	S	F	F	F	F	3.26674	3.28388	3.32509	3.42330	0.01546
–	F	S	S	S	F	F	F	–	–	3.32762	3.42552	–
–	F	S	S	L	F	F	F	–	–	3.33634	3.44857	–
16	S	F	F	F	F	F	F	3.26698	3.28239	3.32469	3.42945	0.01658
–	S	F	F	S	F	F	F	–	–	3.32055	3.42077	–
–	S	F	F	L	F	F	F	–	–	3.32176	3.42727	–
17	S	S	F	F	F	E	E	3.27009	3.28723	3.32853	3.43599	0.02117
–	S	S	F	S	F	E	E	–	–	3.32988	3.43297	–
–	S	S	F	L	F	E	E	–	–	3.33161	3.44361	–
18	S	S	F	F	F	E	F	3.26947	3.28703	3.32895	3.43819	0.02162
–	S	S	F	S	F	E	F	–	–	3.32859	3.43691	–
–	S	S	F	L	F	E	F	–	–	3.33230	3.44644	–
19	S	F	S	F	F	F	E	3.26553	3.28225	3.33330	3.44790	0.02295
–	S	F	S	S	F	F	E	–	–	3.33469	3.44498	–
–	S	F	S	L	F	F	E	–	–	3.34402	3.46281	–
20	F	F	S	F	F	F	E	3.26878	3.29065	3.33646	3.43761	0.02408
–	F	F	S	S	F	F	E	–	–	3.33811	3.44011	–
–	F	F	S	L	F	F	E	–	–	3.34862	3.46473	–
21	F	F	S	F	F	F	F	3.26933	3.29168	3.33705	3.43897	0.02496
–	F	F	S	S	F	F	F	–	–	3.34039	3.44386	–
–	F	F	S	L	F	F	F	–	–	3.35028	3.46872	–

Continued on the next page

Table 9 continued

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
22	S	F	S	F	F	F	F	3.26602	3.28256	3.33821	3.46006	0.02742
–	S	F	S	S	F	F	F	–	–	3.33707	3.45351	–
–	S	F	S	L	F	F	F	–	–	3.34595	3.47147	–
23	F	L	F	F	F	E	E	3.27078	3.28823	3.33705	3.45730	0.02905
–	F	L	F	S	F	E	E	–	–	3.33696	3.45543	–
–	F	L	F	L	F	E	E	–	–	3.34022	3.46689	–
24	F	L	F	F	F	E	F	3.27075	3.28799	3.33649	3.45899	0.02926
–	F	L	F	S	F	E	F	–	–	3.33692	3.45878	–
–	F	L	F	L	F	E	F	–	–	3.33938	3.47235	–
25	S	S	L	F	F	F	E	3.27096	3.29355	3.35125	3.44485	0.03086
–	S	S	L	S	F	F	E	–	–	3.35605	3.46068	–
–	S	S	L	L	F	F	E	–	–	3.37536	3.50578	–
26	L	F	F	F	F	F	E	3.27161	3.29175	3.34037	3.45983	0.03160
–	L	F	F	S	F	F	E	–	–	3.34124	3.45717	–
–	L	F	F	L	F	F	E	–	–	3.34080	3.45409	–
27	F	S	F	F	F	E	F	3.27080	3.28948	3.34115	3.46299	0.03181
–	F	S	F	S	F	E	F	–	–	3.34167	3.46444	–
–	F	S	F	L	F	E	F	–	–	3.34595	3.47907	–
28	F	S	F	F	F	E	E	3.27052	3.28904	3.34257	3.46411	0.03227
–	F	S	F	S	F	E	E	–	–	3.34292	3.46128	–
–	F	S	F	L	F	E	E	–	–	3.34624	3.47652	–
29	L	F	F	F	F	F	F	3.26996	3.29120	3.34286	3.46456	0.03285
–	L	F	F	S	F	F	F	–	–	3.34319	3.46031	–
–	L	F	F	L	F	F	F	–	–	3.34259	3.46376	–
30	S	S	L	F	F	F	F	3.27052	3.29347	3.35301	3.45268	0.03313
–	S	S	L	S	F	F	F	–	–	3.35815	3.46853	–
–	S	S	L	L	F	F	F	–	–	3.37625	3.51069	–
31	F	L	L	F	F	F	E	3.26985	3.28710	3.35634	3.45897	0.03377
–	F	L	L	S	F	F	E	–	–	3.35913	3.47296	–
–	F	L	L	L	F	F	E	–	–	3.37752	3.51928	–
32	S	F	F	F	F	E	E	3.26852	3.28800	3.34025	3.48079	0.03510
–	S	F	F	S	F	E	E	–	–	3.33740	3.47934	–
–	S	F	F	L	F	E	E	–	–	3.34190	3.49182	–
33	S	L	F	F	F	F	E	3.27140	3.29573	3.35026	3.46358	0.03595
–	S	L	F	S	F	F	E	–	–	3.34649	3.45533	–
–	S	L	F	L	F	F	E	–	–	3.34588	3.45636	–
34	F	L	L	F	F	F	F	3.27052	3.28720	3.35661	3.46916	0.03658
–	F	L	L	S	F	F	F	–	–	3.35868	3.46831	–
–	F	L	L	L	F	F	F	–	–	3.37846	3.51681	–
35	S	L	F	F	F	F	F	3.27125	3.29521	3.35085	3.46854	0.03717
–	S	L	F	S	F	F	F	–	–	3.34754	3.46075	–
–	S	L	F	L	F	F	F	–	–	3.34641	3.46164	–
36	S	F	F	F	F	E	F	3.26899	3.28903	3.34018	3.48861	0.03741
–	S	F	F	S	F	E	F	–	–	3.33946	3.48657	–
–	S	F	F	L	F	E	F	–	–	3.34331	3.50233	–
37	S	L	S	F	F	F	E	3.26963	3.29184	3.35409	3.47266	0.03776
–	S	L	S	S	F	F	E	–	–	3.35244	3.46761	–
–	S	L	S	L	F	F	E	–	–	3.35910	3.48597	–
38	S	S	S	F	F	E	E	3.26882	3.28579	3.35457	3.48086	0.03822
–	S	S	S	S	F	E	E	–	–	3.35648	3.49306	–
–	S	S	S	L	F	E	E	–	–	3.36850	3.53254	–
39	S	S	S	F	F	E	F	3.26944	3.28538	3.35328	3.48524	0.03904
–	S	S	S	S	F	E	F	–	–	3.35512	3.49991	–
–	S	S	S	L	F	E	F	–	–	3.36862	3.54336	–
40	S	L	S	F	F	F	F	3.26972	3.29156	3.35550	3.47691	0.03913
–	S	L	S	S	F	F	F	–	–	3.35300	3.47259	–
–	S	L	S	L	F	F	F	–	–	3.35907	3.48913	–
41	L	F	S	F	F	F	E	3.26973	3.28982	3.35248	3.48820	0.04076
–	L	F	S	S	F	F	E	–	–	3.35167	3.49549	–
–	L	F	S	L	F	F	E	–	–	3.35693	3.50440	–
42	F	L	S	F	F	E	F	3.26886	3.28792	3.35275	3.49108	0.04086
–	F	L	S	S	F	E	F	–	–	3.35574	3.50669	–
–	F	L	S	L	F	E	F	–	–	3.36973	3.54879	–
43	F	F	F	F	F	E	F	3.27201	3.29599	3.35438	3.48458	0.04245
–	F	F	F	S	F	E	F	–	–	3.35470	3.48362	–
–	F	F	F	L	F	E	F	–	–	3.35995	3.50184	–
44	F	F	F	F	F	E	E	3.27266	3.29544	3.35640	3.48384	0.04279
–	F	F	F	S	F	E	E	–	–	3.35634	3.48208	–
–	F	F	F	L	F	E	E	–	–	3.36087	3.49905	–

Continued on the next page

Table 9 continued

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
45	F	L	S	F	F	E	E	3.26955	3.28869	3.35771	3.49426	0.04326
–	F	L	S	S	F	E	E	–	–	3.35708	3.50704	–
–	F	L	S	L	F	E	E	–	–	3.36989	3.54545	–
46	F	S	L	F	F	F	E	3.27026	3.29152	3.36877	3.48052	0.04347
–	F	S	L	S	F	F	E	–	–	3.36815	3.49235	–
–	F	S	L	L	F	F	E	–	–	3.38877	3.54789	–
47	S	F	L	F	F	F	E	3.27082	3.29707	3.36759	3.48050	0.04470
–	S	F	L	S	F	F	E	–	–	3.37299	3.50451	–
–	S	F	L	L	F	F	E	–	–	3.39364	3.55391	–
48	L	F	S	F	F	F	F	3.26847	3.28979	3.35534	3.50281	0.04481
–	L	F	S	S	F	F	F	–	–	3.35375	3.50827	–
–	L	F	S	L	F	F	F	–	–	3.36314	3.52022	–
49	L	F	F	F	F	E	E	3.26990	3.29137	3.35783	3.49926	0.04530
–	L	F	F	S	F	E	E	–	–	3.35391	3.50006	–
–	L	F	F	L	F	E	E	–	–	3.35594	3.51865	–
50	L	F	F	F	F	E	F	3.26999	3.29075	3.35708	3.50485	0.04637
–	L	F	F	S	F	E	F	–	–	3.35488	3.51590	–
–	L	F	F	L	F	E	F	–	–	3.35830	3.53581	–
51	F	S	L	F	F	F	F	3.27089	3.29131	3.36993	3.49315	0.04703
–	F	S	L	S	F	F	F	–	–	3.36818	3.49415	–
–	F	S	L	L	F	F	F	–	–	3.38858	3.54763	–
52	F	S	S	F	F	E	F	3.26865	3.29161	3.36117	3.51000	0.04856
–	F	S	S	S	F	E	F	–	–	3.36472	3.52965	–
–	F	S	S	L	F	E	F	–	–	3.38026	3.57874	–
53	S	F	L	F	F	F	F	3.27072	3.29802	3.37252	3.50341	0.05188
–	S	F	L	S	F	F	F	–	–	3.37660	3.51880	–
–	S	F	L	L	F	F	F	–	–	3.39722	3.56837	–
54	L	S	F	F	F	F	E	3.27364	3.30233	3.36465	3.50630	0.05244
–	L	S	F	S	F	F	E	–	–	3.36498	3.50654	–
–	L	S	F	L	F	F	E	–	–	3.36443	3.50462	–
55	F	S	S	F	F	E	E	3.26924	3.29239	3.36642	3.51918	0.05251
–	F	S	S	S	F	E	E	–	–	3.36725	3.53452	–
–	F	S	S	L	F	E	E	–	–	3.38174	3.58259	–
56	S	L	F	F	F	E	E	3.27353	3.30222	3.36651	3.50676	0.05296
–	S	L	F	S	F	E	E	–	–	3.36681	3.50307	–
–	S	L	F	L	F	E	E	–	–	3.36902	3.51296	–
57	S	L	F	F	F	E	F	3.27386	3.30081	3.36534	3.50990	0.05318
–	S	L	F	S	F	E	F	–	–	3.36524	3.50783	–
–	S	L	F	L	F	E	F	–	–	3.36846	3.52067	–
58	F	F	L	F	F	F	E	3.27178	3.30007	3.38283	3.49826	0.05394
–	F	F	L	S	F	F	E	–	–	3.38260	3.51403	–
–	F	F	L	L	F	F	E	–	–	3.40444	3.57288	–
59	L	F	L	F	F	F	E	3.26990	3.30850	3.36839	3.50619	0.05395
–	L	F	L	S	F	F	E	–	–	3.37534	3.51785	–
–	L	F	L	L	F	F	E	–	–	3.39388	3.57341	–
60	S	L	L	F	F	F	E	3.27336	3.30437	3.37835	3.49820	0.05428
–	S	L	L	S	F	F	E	–	–	3.38176	3.51100	–
–	S	L	L	L	F	F	E	–	–	3.39974	3.55395	–
61	L	S	F	F	F	F	F	3.27364	3.30330	3.36546	3.51414	0.05484
–	L	S	F	S	F	F	F	–	–	3.36490	3.50987	–
–	L	S	F	L	F	F	F	–	–	3.36497	3.51186	–
62	L	S	S	F	F	F	E	3.27204	3.30281	3.37695	3.50829	0.05573
–	L	S	S	S	F	F	E	–	–	3.37692	3.51062	–
–	L	S	S	L	F	F	E	–	–	3.38174	3.52278	–
63	S	L	L	F	F	F	F	3.27323	3.30428	3.37886	3.50430	0.05587
–	S	L	L	S	F	F	F	–	–	3.38268	3.51423	–
–	S	L	L	L	F	F	F	–	–	3.40069	3.55947	–
64	L	S	S	F	F	F	F	3.27340	3.30231	3.37747	3.51327	0.05732
–	L	S	S	S	F	F	F	–	–	3.37555	3.51676	–
–	L	S	S	L	F	F	F	–	–	3.38186	3.53822	–
65	L	F	L	F	F	F	F	3.26967	3.30862	3.37376	3.52122	0.05902
–	L	F	L	S	F	F	F	–	–	3.37668	3.53280	–
–	L	F	L	L	F	F	F	–	–	3.39632	3.58523	–
66	F	F	L	F	F	F	F	3.27271	3.30070	3.38509	3.51821	0.05988
–	F	F	L	S	F	F	F	–	–	3.38396	3.51879	–
–	F	F	L	L	F	F	F	–	–	3.40436	3.57216	–
67	S	F	S	F	F	E	E	3.26894	3.28897	3.37167	3.54713	0.05988
–	S	F	S	S	F	E	E	–	–	3.37147	3.56243	–
–	S	F	S	L	F	E	E	–	–	3.38724	3.61663	–

Continued on the next page

Table 9 continued

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
68	F	F	S	F	F	E	F	3.27089	3.30021	3.37859	3.53451	0.06176
–	F	F	S	S	F	E	F	–	–	3.38068	3.55512	–
–	F	F	S	L	F	E	F	–	–	3.39769	3.61328	–
69	L	F	S	F	F	E	E	3.26967	3.29642	3.38304	3.53954	0.06287
–	L	F	S	S	F	E	E	–	–	3.38362	3.55066	–
–	L	F	S	L	F	E	E	–	–	3.39721	3.58915	–
70	L	S	L	F	F	F	E	3.27410	3.31873	3.38615	3.51094	0.06319
–	L	S	L	S	F	F	E	–	–	3.39314	3.52416	–
–	L	S	L	L	F	F	E	–	–	3.41121	3.56990	–
71	S	L	S	F	F	E	F	3.27281	3.29880	3.38592	3.53275	0.06328
–	S	L	S	S	F	E	F	–	–	3.38673	3.54664	–
–	S	L	S	L	F	E	F	–	–	3.39755	3.58131	–
72	S	F	S	F	F	E	F	3.26864	3.28828	3.37153	3.56278	0.06351
–	S	F	S	S	F	E	F	–	–	3.37253	3.57494	–
–	S	F	S	L	F	E	F	–	–	3.39115	3.63345	–
73	S	L	S	F	F	E	E	3.27230	3.29994	3.38764	3.53229	0.06375
–	S	L	S	S	F	E	E	–	–	3.38990	3.54270	–
–	S	L	S	L	F	E	E	–	–	3.39896	3.57394	–
74	L	S	F	F	F	E	E	3.27439	3.30424	3.38232	3.53561	0.06485
–	L	S	F	S	F	E	E	–	–	3.37844	3.53977	–
–	L	S	F	L	F	E	E	–	–	3.38031	3.54683	–
75	F	F	S	F	F	E	E	3.27112	3.30024	3.38287	3.54289	0.06499
–	F	F	S	S	F	E	E	–	–	3.38370	3.56084	–
–	F	F	S	L	F	E	E	–	–	3.40034	3.61307	–
76	L	S	F	F	F	E	F	3.27527	3.30478	3.38143	3.54119	0.06637
–	L	S	F	S	F	E	F	–	–	3.38024	3.54597	–
–	L	S	F	L	F	E	F	–	–	3.38078	3.55334	–
77	L	S	L	F	F	F	F	3.27313	3.31790	3.38969	3.52262	0.06654
–	L	S	L	S	F	F	F	–	–	3.39494	3.52776	–
–	L	S	L	L	F	F	F	–	–	3.41292	3.58013	–
78	L	F	S	F	F	E	F	3.26965	3.29774	3.38308	3.55340	0.06667
–	L	F	S	S	F	E	F	–	–	3.38407	3.58811	–
–	L	F	S	L	F	E	F	–	–	3.40805	3.63847	–
79	S	F	F	F	E	F	E	3.27015	3.29547	3.38745	3.56516	0.07026
–	S	F	F	S	E	F	E	–	–	3.38383	3.56421	–
–	S	F	F	L	E	F	E	–	–	3.38653	3.56805	–
80	S	S	F	F	E	F	E	3.27080	3.29562	3.38689	3.56972	0.07146
–	S	S	F	S	E	F	E	–	–	3.38266	3.56881	–
–	S	S	F	L	E	F	E	–	–	3.38367	3.57306	–
81	S	S	L	F	F	E	E	3.26973	3.30190	3.38543	3.56649	0.07159
–	S	S	L	S	F	E	E	–	–	3.39864	3.60327	–
–	S	S	L	L	F	E	E	–	–	3.42474	3.66066	–
82	S	S	F	F	E	F	F	3.27009	3.29522	3.38539	3.57329	0.07171
–	S	S	F	S	E	F	F	–	–	3.38282	3.57221	–
–	S	S	F	L	E	F	F	–	–	3.38487	3.57494	–
83	S	S	L	F	F	E	F	3.27052	3.30178	3.37860	3.57349	0.07180
–	S	S	L	S	F	E	F	–	–	3.39472	3.61570	–
–	S	S	L	L	F	E	F	–	–	3.42203	3.79827	–
84	S	F	F	F	E	F	F	3.26972	3.29574	3.38862	3.57305	0.07249
–	S	F	F	S	E	F	F	–	–	3.38693	3.57149	–
–	S	F	F	L	E	F	F	–	–	3.39022	3.57921	–
85	F	L	L	F	F	E	F	3.27006	3.29976	3.39188	3.57102	0.07389
–	F	L	L	S	F	E	F	–	–	3.39795	3.59657	–
–	F	L	L	L	F	E	F	–	–	3.42935	3.66573	–
86	S	S	S	F	E	F	E	3.27056	3.29685	3.39478	3.57657	0.07540
–	S	S	S	S	E	F	E	–	–	3.39667	3.57644	–
–	S	S	S	L	E	F	E	–	–	3.40565	3.59924	–
87	F	S	F	F	E	F	E	3.27278	3.30364	3.39728	3.57016	0.07667
–	F	S	F	S	E	F	E	–	–	3.39657	3.56606	–
–	F	S	F	L	E	F	E	–	–	3.39808	3.57036	–
88	S	S	S	F	E	F	F	3.27064	3.29712	3.39446	3.58264	0.07692
–	S	S	S	S	E	F	F	–	–	3.39462	3.57953	–
–	S	S	S	L	E	F	F	–	–	3.40620	3.60279	–
89	L	S	S	F	F	E	E	3.27365	3.30908	3.40485	3.55943	0.07746
–	L	S	S	S	F	E	E	–	–	3.40427	3.57649	–
–	L	S	S	L	F	E	E	–	–	3.41857	3.60282	–
90	F	S	F	F	E	F	F	3.27291	3.30363	3.39810	3.57325	0.07768
–	F	S	F	S	E	F	F	–	–	3.39650	3.56898	–
–	F	S	F	L	E	F	F	–	–	3.39807	3.57387	–

Continued on the next page

Table 9 continued

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
91	F	L	F	F	E	F	E	3.27375	3.30073	3.39503	3.58196	0.07858
–	F	L	F	S	E	F	E	–	–	3.39452	3.57614	–
–	F	L	F	L	E	F	E	–	–	3.39482	3.57816	–
92	S	F	S	F	E	F	E	3.27063	3.29884	3.40146	3.58376	0.07938
–	S	F	S	S	E	F	E	–	–	3.40316	3.58572	–
–	S	F	S	L	E	F	E	–	–	3.41534	3.61346	–
93	L	S	S	F	F	E	F	3.27328	3.30758	3.40495	3.56966	0.07957
–	L	S	S	S	F	E	F	–	–	3.40486	3.58304	–
–	L	S	S	L	F	E	F	–	–	3.41792	3.61759	–
94	F	L	F	F	E	F	F	3.27311	3.30094	3.39547	3.58616	0.07963
–	F	L	F	S	E	F	F	–	–	3.39449	3.58048	–
–	F	L	F	L	E	F	F	–	–	3.39509	3.58183	–
95	S	F	S	F	E	F	F	3.27064	3.29988	3.40160	3.59343	0.08210
–	S	F	S	S	E	F	F	–	–	3.40327	3.59396	–
–	S	F	S	L	E	F	F	–	–	3.41426	3.62240	–
96	F	L	S	F	E	F	E	3.27139	3.30217	3.40070	3.59264	0.08243
–	F	L	S	S	E	F	E	–	–	3.40116	3.59583	–
–	F	L	S	L	E	F	E	–	–	3.41093	3.62138	–
97	F	L	L	F	F	E	E	3.27006	3.30021	3.39828	3.59889	0.08257
–	F	L	L	S	F	E	E	–	–	3.40711	3.60678	–
–	F	L	L	L	F	E	E	–	–	3.43381	3.67016	–
98	F	S	S	F	E	F	E	3.27045	3.30556	3.40433	3.59271	0.08397
–	F	S	S	S	E	F	E	–	–	3.40554	3.59852	–
–	F	S	S	L	E	F	E	–	–	3.41643	3.63118	–
99	F	L	S	F	E	F	F	3.27111	3.30256	3.40087	3.60062	0.08450
–	F	L	S	S	E	F	F	–	–	3.40264	3.60554	–
–	F	L	S	L	E	F	F	–	–	3.41190	3.63236	–
100	F	S	S	F	E	F	F	3.27114	3.30560	3.40436	3.59572	0.08491
–	F	S	S	S	E	F	F	–	–	3.40532	3.60322	–
–	F	S	S	L	E	F	F	–	–	3.41618	3.63359	–
101	F	F	F	F	E	F	E	3.27438	3.31208	3.41199	3.58542	0.08667
–	F	F	F	S	E	F	E	–	–	3.41075	3.58197	–
–	F	F	F	L	E	F	E	–	–	3.41273	3.58555	–
102	S	L	L	F	F	E	F	3.27229	3.31356	3.40702	3.59173	0.08686
–	S	L	L	S	F	E	F	–	–	3.42106	3.62514	–
–	S	L	L	L	F	E	F	–	–	3.44674	3.73170	–
103	F	F	F	F	E	F	F	3.27507	3.31179	3.41247	3.58743	0.08740
–	F	F	F	S	E	F	F	–	–	3.41106	3.58329	–
–	F	F	F	L	E	F	F	–	–	3.41392	3.58944	–
104	L	F	F	F	E	F	E	3.27087	3.30218	3.40544	3.62799	0.09233
–	L	F	F	S	E	F	E	–	–	3.40447	3.63644	–
–	L	F	F	L	E	F	E	–	–	3.40579	3.63915	–
105	F	S	L	F	F	E	F	3.27014	3.30341	3.40740	3.63366	0.09436
–	F	S	L	S	F	E	F	–	–	3.41708	3.66810	–
–	F	S	L	L	F	E	F	–	–	3.44683	3.79671	–
106	F	F	S	F	E	F	E	3.27256	3.31398	3.42072	3.60853	0.09465
–	F	F	S	S	E	F	E	–	–	3.42162	3.61711	–
–	F	F	S	L	E	F	E	–	–	3.43339	3.65103	–
107	F	F	S	F	E	F	F	3.27315	3.31411	3.42049	3.61227	0.09571
–	F	F	S	S	E	F	F	–	–	3.42189	3.62022	–
–	F	F	S	L	E	F	F	–	–	3.43341	3.65352	–
108	L	F	F	F	E	F	F	3.27096	3.30204	3.40830	3.63955	0.09592
–	L	F	F	S	E	F	F	–	–	3.40599	3.65423	–
–	L	F	F	L	E	F	F	–	–	3.40633	3.64917	–
109	S	S	L	F	E	F	E	3.27147	3.30675	3.42397	3.62020	0.09630
–	S	S	L	S	E	F	E	–	–	3.43255	3.63662	–
–	S	S	L	L	E	F	E	–	–	3.45249	3.69439	–
110	S	L	L	F	F	E	E	3.27302	3.31636	3.41418	3.62648	0.09822
–	S	L	L	S	F	E	E	–	–	3.42749	3.65546	–
–	S	L	L	L	F	E	E	–	–	3.45125	3.72041	–
111	S	S	F	F	E	E	E	3.27281	3.30878	3.41910	3.63004	0.09839
–	S	S	F	S	E	E	E	–	–	3.42024	3.63153	–
–	S	S	F	L	E	E	E	–	–	3.42463	3.64269	–
112	S	S	L	F	E	F	F	3.27189	3.30614	3.42585	3.62943	0.09903
–	S	S	L	S	E	F	F	–	–	3.43160	3.64592	–
–	S	S	L	L	E	F	F	–	–	3.45217	3.69981	–
113	S	S	F	F	E	E	F	3.27300	3.30760	3.41879	3.63643	0.09966
–	S	S	F	S	E	E	F	–	–	3.41813	3.63471	–
–	S	S	F	L	E	E	F	–	–	3.42405	3.64973	–

Continued on the next page

Table 9 continued

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
114	L	F	S	F	E	F	E	3.26966	3.30198	3.42146	3.64368	0.09990
–	L	F	S	S	E	F	E	–	–	3.42216	3.64950	–
–	L	F	S	L	E	F	E	–	–	3.42922	3.67394	–
115	F	S	L	F	F	E	E	3.26997	3.30432	3.41649	3.64978	0.10085
–	F	S	L	S	F	E	E	–	–	3.42549	3.67368	–
–	F	S	L	L	F	E	E	–	–	3.45707	3.72758	–
116	S	F	F	F	E	E	E	3.27229	3.30865	3.41948	3.64135	0.10115
–	S	F	F	S	E	E	E	–	–	3.42048	3.64086	–
–	S	F	F	L	E	E	E	–	–	3.42599	3.66023	–
117	L	F	L	F	F	E	E	3.27088	3.31336	3.41221	3.64697	0.10156
–	L	F	L	S	F	E	E	–	–	3.42188	3.68835	–
–	L	F	L	L	F	E	E	–	–	3.45055	3.76981	–
118	S	F	L	F	F	E	E	3.26995	3.30685	3.40469	3.66421	0.10213
–	S	F	L	S	F	E	E	–	–	3.42241	3.69750	–
–	S	F	L	L	F	E	E	–	–	3.45336	3.74800	–
119	L	S	L	F	F	E	F	3.27390	3.32261	3.42261	3.62813	0.10252
–	L	S	L	S	F	E	F	–	–	3.43535	3.65948	–
–	L	S	L	L	F	E	F	–	–	3.46585	3.82726	–
120	S	F	F	F	E	E	F	3.27231	3.30850	3.42131	3.64668	0.10291
–	S	F	F	S	E	E	F	–	–	3.42031	3.64387	–
–	S	F	F	L	E	E	F	–	–	3.42611	3.66946	–
121	S	F	L	F	E	F	E	3.27188	3.30909	3.43467	3.63705	0.10388
–	S	F	L	S	E	F	E	–	–	3.44243	3.65909	–
–	S	F	L	L	E	F	E	–	–	3.46451	3.72372	–
122	L	S	L	F	F	E	E	3.27334	3.32433	3.42806	3.62798	0.10413
–	L	S	L	S	F	E	E	–	–	3.43953	3.65493	–
–	L	S	L	L	F	E	E	–	–	3.46804	3.75959	–
123	L	F	S	F	E	F	F	3.26983	3.30120	3.42097	3.66407	0.10472
–	L	F	S	S	E	F	F	–	–	3.42210	3.66599	–
–	L	F	S	L	E	F	F	–	–	3.43189	3.69119	–
124	S	F	L	F	E	F	F	3.27188	3.30941	3.43692	3.64653	0.10689
–	S	F	L	S	E	F	F	–	–	3.44289	3.66911	–
–	S	F	L	L	E	F	F	–	–	3.46710	3.73513	–
125	L	F	F	F	E	E	E	3.27058	3.30693	3.42605	3.66749	0.10847
–	L	F	F	S	E	E	E	–	–	3.42720	3.68012	–
–	L	F	F	L	E	E	E	–	–	3.43057	3.68235	–
126	F	L	L	F	E	F	E	3.27349	3.30426	3.43774	3.66120	0.10988
–	F	L	L	S	E	F	E	–	–	3.45143	3.68510	–
–	F	L	L	L	E	F	E	–	–	3.47586	3.74367	–
127	L	F	F	F	E	E	F	3.26971	3.30464	3.42887	3.67517	0.11030
–	L	F	F	S	E	E	F	–	–	3.42784	3.69107	–
–	L	F	F	L	E	E	F	–	–	3.43473	3.70315	–
128	L	F	L	F	F	E	F	3.27061	3.31252	3.40914	3.68730	0.11060
–	L	F	L	S	F	E	F	–	–	3.42081	3.72276	–
–	L	F	L	L	F	E	F	–	–	3.48281	3.84013	–
129	S	S	S	F	E	E	F	3.27243	3.30754	3.44231	3.66077	0.11147
–	S	S	S	S	E	E	F	–	–	3.44424	3.67149	–
–	S	S	S	L	E	E	F	–	–	3.45887	3.71791	–
130	S	S	S	F	E	E	E	3.27277	3.30880	3.44400	3.66005	0.11211
–	S	S	S	S	E	E	E	–	–	3.44637	3.67028	–
–	S	S	S	L	E	E	E	–	–	3.46200	3.71118	–
131	S	L	S	F	E	F	E	3.27419	3.31153	3.44186	3.66621	0.11415
–	S	L	S	S	E	F	E	–	–	3.44234	3.66575	–
–	S	L	S	L	E	F	E	–	–	3.44910	3.68686	–
132	L	F	L	F	E	F	E	3.27096	3.31703	3.44034	3.66633	0.11437
–	L	F	L	S	E	F	E	–	–	3.44848	3.67749	–
–	L	F	L	L	E	F	E	–	–	3.46911	3.73821	–
133	S	F	L	F	F	E	F	3.27009	3.30588	3.40226	3.71736	0.11461
–	S	F	L	S	F	E	F	–	–	3.43561	3.73386	–
–	S	F	L	L	F	E	F	–	–	3.48369	3.93320	–
134	S	L	S	F	E	F	F	3.27375	3.31170	3.43977	3.67038	0.11461
–	S	L	S	S	E	F	F	–	–	3.44010	3.66718	–
–	S	L	S	L	E	F	F	–	–	3.44945	3.69170	–
135	S	L	F	F	E	F	F	3.27421	3.31009	3.43936	3.67314	0.11491
–	S	L	F	S	E	F	F	–	–	3.43464	3.67541	–
–	S	L	F	L	E	F	F	–	–	3.43427	3.67531	–
136	S	L	F	F	E	F	E	3.27472	3.31052	3.44083	3.67763	0.11663
–	S	L	F	S	E	F	E	–	–	3.43897	3.67499	–
–	S	L	F	L	E	F	E	–	–	3.43826	3.67688	–

Continued on the next page

Table 9 continued

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
137	F	L	L	F	E	F	F	3.27334	3.30404	3.44020	3.68819	0.11715
–	F	L	L	S	E	F	F	–	–	3.45219	3.71055	–
–	F	L	L	L	E	F	F	–	–	3.47926	3.76418	–
138	F	S	F	F	E	E	F	3.27416	3.31641	3.44818	3.67311	0.11867
–	F	S	F	S	E	E	F	–	–	3.44844	3.67740	–
–	F	S	F	L	E	E	F	–	–	3.45472	3.69729	–
139	F	F	L	F	F	E	E	3.27236	3.31249	3.43620	3.69192	0.11895
–	F	F	L	S	F	E	E	–	–	3.44560	3.71703	–
–	F	F	L	L	F	E	E	–	–	3.47779	3.78841	–
140	F	S	F	F	E	E	E	3.27469	3.31607	3.44794	3.67597	0.11937
–	F	S	F	S	E	E	E	–	–	3.44782	3.67648	–
–	F	S	F	L	E	E	E	–	–	3.45452	3.69578	–
141	F	S	L	F	E	F	E	3.27367	3.30810	3.44834	3.68606	0.11975
–	F	S	L	S	E	F	E	–	–	3.46091	3.71551	–
–	F	S	L	L	E	F	E	–	–	3.48721	3.78085	–
142	S	F	S	F	E	E	E	3.27283	3.31151	3.45353	3.68130	0.12050
–	S	F	S	S	E	E	E	–	–	3.45494	3.70118	–
–	S	F	S	L	E	E	E	–	–	3.47449	3.75204	–
143	S	F	S	F	E	E	F	3.27240	3.31159	3.45025	3.68855	0.12140
–	S	F	S	S	E	E	F	–	–	3.45166	3.70948	–
–	S	F	S	L	E	E	F	–	–	3.47467	3.76081	–
144	L	F	L	F	E	F	F	3.26998	3.31693	3.44596	3.69094	0.12166
–	L	F	L	S	E	F	F	–	–	3.45028	3.69400	–
–	L	F	L	L	E	F	F	–	–	3.46977	3.75190	–
145	F	F	L	F	F	E	F	3.27226	3.31370	3.42845	3.71519	0.12311
–	F	F	L	S	F	E	F	–	–	3.44074	3.70264	–
–	F	F	L	L	F	E	F	–	–	3.47637	3.86243	–
146	L	L	S	F	F	F	E	3.28057	3.33531	3.45602	3.66179	0.12413
–	L	L	S	S	F	F	E	–	–	3.45222	3.65561	–
–	L	L	S	L	F	F	E	–	–	3.45912	3.66677	–
147	F	S	L	F	E	F	F	3.27396	3.30878	3.44842	3.70344	0.12436
–	F	S	L	S	E	F	F	–	–	3.46154	3.72633	–
–	F	S	L	L	E	F	F	–	–	3.48964	3.78798	–
148	L	L	F	F	F	F	E	3.28325	3.33974	3.44900	3.66492	0.12493
–	L	L	F	S	F	F	E	–	–	3.44770	3.66267	–
–	L	L	F	L	F	F	E	–	–	3.44770	3.66120	–
149	L	L	S	F	F	F	F	3.28057	3.33473	3.45596	3.66796	0.12551
–	L	L	S	S	F	F	F	–	–	3.45499	3.66643	–
–	L	L	S	L	F	F	F	–	–	3.45777	3.67747	–
150	F	L	F	F	E	E	E	3.27499	3.31386	3.45215	3.70032	0.12604
–	F	L	F	S	E	E	E	–	–	3.45182	3.69974	–
–	F	L	F	L	E	E	E	–	–	3.45521	3.71222	–
151	L	L	F	F	F	F	F	3.28177	3.33938	3.44983	3.67287	0.12667
–	L	L	F	S	F	F	F	–	–	3.45001	3.67024	–
–	L	L	F	L	F	F	F	–	–	3.44793	3.67108	–
152	F	L	F	F	E	E	F	3.27505	3.31193	3.45328	3.70673	0.12745
–	F	L	F	S	E	E	F	–	–	3.45292	3.70642	–
–	F	L	F	L	E	E	F	–	–	3.45713	3.71866	–
153	F	F	F	F	E	E	F	3.27621	3.32520	3.46160	3.68417	0.12750
–	F	F	F	S	E	E	F	–	–	3.46227	3.68970	–
–	F	F	F	L	E	E	F	–	–	3.47035	3.71053	–
154	S	L	L	F	E	F	E	3.27438	3.32243	3.45972	3.69328	0.12816
–	S	L	L	S	E	F	E	–	–	3.46636	3.70922	–
–	S	L	L	L	E	F	E	–	–	3.48499	3.75768	–
155	L	S	S	F	E	F	E	3.27376	3.31755	3.45387	3.70741	0.12885
–	L	S	S	S	E	F	E	–	–	3.45216	3.70887	–
–	L	S	S	L	E	F	E	–	–	3.45956	3.71534	–
156	F	F	F	F	E	E	E	3.27666	3.32552	3.46308	3.68847	0.12914
–	F	F	F	S	E	E	E	–	–	3.46290	3.69242	–
–	F	F	F	L	E	E	E	–	–	3.47041	3.71449	–
157	L	S	F	F	E	F	E	3.27546	3.31782	3.45048	3.71148	0.12952
–	L	S	F	S	E	F	E	–	–	3.44878	3.70728	–
–	L	S	F	L	E	F	E	–	–	3.44999	3.70363	–
158	L	S	S	F	E	F	F	3.27311	3.31576	3.45393	3.71328	0.12973
–	L	S	S	S	E	F	F	–	–	3.45216	3.70535	–
–	L	S	S	L	E	F	F	–	–	3.46068	3.72084	–
159	L	S	F	F	E	F	F	3.27529	3.31690	3.45048	3.71407	0.12989
–	L	S	F	S	E	F	F	–	–	3.44992	3.71078	–
–	L	S	F	L	E	F	F	–	–	3.45054	3.70651	–

Continued on the next page

Table 9 continued

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
160	L	F	S	F	E	E	E	3.27329	3.31045	3.45452	3.72224	0.13083
–	L	F	S	S	E	E	E	–	–	3.45934	3.75047	–
–	L	F	S	L	E	E	E	–	–	3.47622	3.78339	–
161	S	L	L	F	E	F	F	3.27419	3.32201	3.45902	3.70586	0.13098
–	S	L	L	S	E	F	F	–	–	3.46601	3.71884	–
–	S	L	L	L	E	F	F	–	–	3.48657	3.76539	–
162	F	F	L	F	E	F	E	3.27679	3.31899	3.46532	3.70834	0.13307
–	F	F	L	S	E	F	E	–	–	3.47929	3.73527	–
–	F	F	L	L	E	F	E	–	–	3.50652	3.80740	–
163	L	L	L	F	F	F	F	3.28046	3.34950	3.46891	3.67280	0.13362
–	L	L	L	S	F	F	F	–	–	3.47182	3.67710	–
–	L	L	L	L	F	F	F	–	–	3.48798	3.71110	–
164	F	L	S	F	E	E	F	3.27258	3.31371	3.46694	3.72538	0.13536
–	F	L	S	S	E	E	F	–	–	3.47273	3.73883	–
–	F	L	S	L	E	E	F	–	–	3.49251	3.79007	–
165	L	F	S	F	E	E	F	3.27178	3.31161	3.45953	3.73621	0.13549
–	L	F	S	S	E	E	F	–	–	3.46236	3.76367	–
–	L	F	S	L	E	E	F	–	–	3.47786	3.80407	–
166	L	L	F	F	F	E	E	3.28141	3.34009	3.46532	3.69431	0.13599
–	L	L	F	S	F	E	E	–	–	3.46472	3.69492	–
–	L	L	F	L	F	E	E	–	–	3.46631	3.70246	–
167	F	S	S	F	E	E	F	3.27294	3.31856	3.47107	3.72012	0.13638
–	F	S	S	S	E	E	F	–	–	3.47753	3.74457	–
–	F	S	S	L	E	E	F	–	–	3.50041	3.81424	–
168	F	L	S	F	E	E	E	3.27220	3.31500	3.46981	3.72574	0.13639
–	F	L	S	S	E	E	E	–	–	3.47523	3.73804	–
–	F	L	S	L	E	E	E	–	–	3.49374	3.78427	–
169	F	F	L	F	E	F	F	3.27678	3.31948	3.46732	3.72085	0.13681
–	F	F	L	S	E	F	F	–	–	3.48001	3.74396	–
–	F	F	L	L	E	F	F	–	–	3.50451	3.82015	–
170	L	L	F	F	F	E	F	3.28170	3.33920	3.46686	3.69791	0.13712
–	L	L	F	S	F	E	F	–	–	3.46483	3.70464	–
–	L	L	F	L	F	E	F	–	–	3.46786	3.71137	–
171	L	L	L	F	F	F	E	3.28035	3.34908	3.46689	3.69788	0.13926
–	L	L	L	S	F	F	E	–	–	3.46913	3.70048	–
–	L	L	L	L	F	F	E	–	–	3.48279	3.73678	–
172	L	S	L	F	E	F	F	3.27356	3.33050	3.47032	3.72428	0.14037
–	L	S	L	S	E	F	F	–	–	3.47216	3.71792	–
–	L	S	L	L	E	F	F	–	–	3.49011	3.75652	–
173	L	S	L	F	E	F	E	3.27342	3.32998	3.47079	3.72986	0.14172
–	L	S	L	S	E	F	E	–	–	3.47337	3.72380	–
–	L	S	L	L	E	F	E	–	–	3.49226	3.76161	–
174	F	S	S	F	E	E	E	3.27290	3.32026	3.47638	3.73543	0.14195
–	F	S	S	S	E	E	E	–	–	3.48262	3.75697	–
–	F	S	S	L	E	E	E	–	–	3.50524	3.81965	–
175	L	S	F	F	E	E	F	3.27381	3.32195	3.47029	3.74573	0.14365
–	L	S	F	S	E	E	F	–	–	3.46994	3.75161	–
–	L	S	F	L	E	E	F	–	–	3.47404	3.75351	–
176	L	S	F	F	E	E	E	3.27529	3.32347	3.47035	3.74398	0.14398
–	L	S	F	S	E	E	E	–	–	3.47069	3.74551	–
–	L	S	F	L	E	E	E	–	–	3.47110	3.74524	–
177	L	L	S	F	F	E	E	3.28045	3.34313	3.48926	3.70357	0.14481
–	L	L	S	S	F	E	E	–	–	3.48850	3.71303	–
–	L	L	S	L	F	E	E	–	–	3.49683	3.74080	–
178	L	L	S	F	F	E	F	3.28153	3.34230	3.48614	3.70984	0.14566
–	L	L	S	S	F	E	F	–	–	3.48761	3.72240	–
–	L	L	S	L	F	E	F	–	–	3.49496	3.75610	–
179	S	S	L	F	E	E	F	3.27338	3.31935	3.47308	3.75595	0.14615
–	S	S	L	S	E	E	F	–	–	3.48630	3.79034	–
–	S	S	L	L	E	E	F	–	–	3.51783	3.89971	–
180	F	F	S	F	E	E	F	3.27488	3.32946	3.48702	3.74315	0.14933
–	F	F	S	S	E	E	F	–	–	3.49436	3.76875	–
–	F	F	S	L	E	E	F	–	–	3.52125	3.84165	–
181	S	S	L	F	E	E	E	3.27327	3.32133	3.47891	3.76683	0.15079
–	S	S	L	S	E	E	E	–	–	3.49123	3.80051	–
–	S	S	L	L	E	E	E	–	–	3.52222	3.85686	–
182	S	L	F	F	E	E	F	3.27628	3.32348	3.48579	3.76112	0.15237
–	S	L	F	S	E	E	F	–	–	3.48274	3.76502	–
–	S	L	F	L	E	E	F	–	–	3.48556	3.77537	–

Continued on the next page

Table 9 continued

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
183	S	L	F	F	E	E	E	3.27616	3.32475	3.48869	3.76151	0.15348
–	S	L	F	S	E	E	E	–	–	3.48776	3.76288	–
–	S	L	F	L	E	E	E	–	–	3.48780	3.77413	–
184	L	S	S	F	E	E	E	3.27654	3.32687	3.48602	3.76219	0.15361
–	L	S	S	S	E	E	E	–	–	3.48735	3.77778	–
–	L	S	S	L	E	E	E	–	–	3.50006	3.79845	–
185	L	S	S	F	E	E	F	3.27632	3.32494	3.48441	3.77066	0.15479
–	L	S	S	S	E	E	F	–	–	3.48790	3.78622	–
–	L	S	S	L	E	E	F	–	–	3.49818	3.80926	–
186	S	L	S	F	E	E	F	3.27536	3.32431	3.49027	3.76967	0.15561
–	S	L	S	S	E	E	F	–	–	3.49290	3.77975	–
–	S	L	S	L	E	E	F	–	–	3.50386	3.80983	–
187	F	F	S	F	E	E	E	3.27487	3.33013	3.49580	3.75956	0.15580
–	F	F	S	S	E	E	E	–	–	3.50041	3.78376	–
–	F	F	S	L	E	E	E	–	–	3.52449	3.85082	–
188	S	L	S	F	E	E	E	3.27563	3.32597	3.49370	3.76678	0.15623
–	S	L	S	S	E	E	E	–	–	3.49470	3.77230	–
–	S	L	S	L	E	E	E	–	–	3.50623	3.80002	–
189	L	L	L	F	F	E	F	3.28214	3.35631	3.50663	3.74184	0.16244
–	L	L	L	S	F	E	F	–	–	3.51462	3.77385	–
–	L	L	L	L	F	E	F	–	–	3.53873	3.86558	–
190	S	F	L	F	E	E	F	3.27300	3.32327	3.49135	3.82403	0.16862
–	S	F	L	S	E	E	F	–	–	3.50819	3.87440	–
–	S	F	L	L	E	E	F	–	–	3.54431	4.01003	–
191	S	F	L	F	E	E	E	3.27328	3.32428	3.49792	3.83446	0.17319
–	S	F	L	S	E	E	E	–	–	3.51369	3.86950	–
–	S	F	L	L	E	E	E	–	–	3.54796	3.91111	–
192	L	L	L	F	F	E	E	3.28106	3.35802	3.51011	3.79451	0.17663
–	L	L	L	S	F	E	E	–	–	3.52112	3.82225	–
–	L	L	L	L	F	E	E	–	–	3.54042	3.88468	–
193	F	L	L	F	E	E	E	3.27455	3.32535	3.51262	3.83321	0.17714
–	F	L	L	S	E	E	E	–	–	3.52989	3.88175	–
–	F	L	L	L	E	E	E	–	–	3.56206	3.93371	–
194	F	L	L	F	E	E	F	3.27462	3.32262	3.51223	3.83671	0.17725
–	F	L	L	S	E	E	F	–	–	3.52470	3.87503	–
–	F	L	L	L	E	E	F	–	–	3.55950	3.93954	–
195	S	L	L	F	E	E	F	3.27553	3.33793	3.51074	3.83254	0.17989
–	S	L	L	S	E	E	F	–	–	3.52073	3.85258	–
–	S	L	L	L	E	E	F	–	–	3.54850	3.92491	–
196	S	L	L	F	E	E	E	3.27551	3.34082	3.51441	3.83131	0.18122
–	S	L	L	S	E	E	E	–	–	3.52480	3.84528	–
–	S	L	L	L	E	E	E	–	–	3.54962	3.91748	–
197	L	S	L	F	E	E	F	3.27641	3.34240	3.50677	3.88256	0.19274
–	L	S	L	S	E	E	F	–	–	3.51779	3.87472	–
–	L	S	L	L	E	E	F	–	–	3.54134	3.96453	–
198	F	S	L	F	E	E	F	3.27443	3.32783	3.52648	3.88161	0.19329
–	F	S	L	S	E	E	F	–	–	3.54036	3.93581	–
–	F	S	L	L	E	E	F	–	–	3.58304	3.98721	–
199	F	S	L	F	E	E	E	3.27458	3.32988	3.52969	3.87967	0.19416
–	F	S	L	S	E	E	E	–	–	3.54768	3.94862	–
–	F	S	L	L	E	E	E	–	–	3.58677	3.99559	–
200	L	F	L	F	E	E	F	3.27210	3.32752	3.48568	3.93940	0.19688
–	L	F	L	S	E	E	F	–	–	3.50109	3.92486	–
–	L	F	L	L	E	E	F	–	–	3.53571	4.03128	–
201	L	F	L	F	E	E	E	3.27216	3.32795	3.49146	3.93707	0.19787
–	L	F	L	S	E	E	E	–	–	3.50423	3.91266	–
–	L	F	L	L	E	E	E	–	–	3.53223	3.99328	–
202	L	S	L	F	E	E	E	3.27732	3.34363	3.50994	3.89995	0.19842
–	L	S	L	S	E	E	E	–	–	3.52046	3.87996	–
–	L	S	L	L	E	E	E	–	–	3.54320	3.94605	–
203	F	F	L	F	E	E	F	3.27661	3.33987	3.54414	3.90612	0.20739
–	F	F	L	S	E	E	F	–	–	3.56202	3.98728	–
–	F	F	L	L	E	E	F	–	–	3.60874	4.06019	–
204	F	F	L	F	E	E	E	3.27690	3.34180	3.55145	3.90959	0.21064
–	F	F	L	S	E	E	E	–	–	3.57330	3.99057	–
–	F	F	L	L	E	E	E	–	–	3.61278	4.04875	–
205	L	L	S	F	E	F	E	3.28110	3.35586	3.56005	3.93956	0.22485
–	L	L	S	S	E	F	E	–	–	3.55599	3.93241	–
–	L	L	S	L	E	F	E	–	–	3.56043	3.94336	–

Continued on the next page

Table 9 continued

Rank	Parameter scaling				EMA scaling			Terminal validation loss				Mean BSZ regret
	η_{mat}	λ_{mat}	η_{aux}	λ_{aux}	μ	β_1	β_2	256K	512K	1M	2M	$\overline{\Delta}$
206	L	L	L	F	E	F	E	3.28136	3.36883	3.56283	3.92717	0.22575
–	L	L	L	S	E	F	E	–	–	3.56768	3.93701	–
–	L	L	L	L	E	F	E	–	–	3.58181	3.98613	–
207	L	L	S	F	E	F	F	3.27988	3.35621	3.55967	3.96047	0.22976
–	L	L	S	S	E	F	F	–	–	3.55844	3.96583	–
–	L	L	S	L	E	F	F	–	–	3.56230	3.97255	–
208	L	L	L	F	E	F	F	3.28122	3.36787	3.56466	3.99377	0.24259
–	L	L	L	S	E	F	F	–	–	3.57125	3.97084	–
–	L	L	L	L	E	F	F	–	–	3.58213	4.00610	–
209	L	L	F	F	E	F	E	3.28338	3.36024	3.57922	3.98963	0.24382
–	L	L	F	S	E	F	E	–	–	3.57705	3.99148	–
–	L	L	F	L	E	F	E	–	–	3.57428	3.98978	–
210	L	L	F	F	E	F	F	3.28227	3.35854	3.57680	3.99957	0.24500
–	L	L	F	S	E	F	F	–	–	3.57941	3.99618	–
–	L	L	F	L	E	F	F	–	–	3.57805	3.98443	–
211	L	L	S	F	E	E	E	3.28397	3.37028	3.60205	4.03314	0.26307
–	L	L	S	S	E	E	E	–	–	3.60149	4.03060	–
–	L	L	S	L	E	E	E	–	–	3.60745	4.04821	–
212	L	L	F	F	E	E	E	3.28304	3.36687	3.61016	4.04059	0.26587
–	L	L	F	S	E	E	E	–	–	3.60645	4.04374	–
–	L	L	F	L	E	E	E	–	–	3.60957	4.04215	–
213	L	L	S	F	E	E	F	3.28399	3.36915	3.59994	4.05086	0.26669
–	L	L	S	S	E	E	F	–	–	3.60024	4.05637	–
–	L	L	S	L	E	E	F	–	–	3.60775	4.08402	–
214	L	L	F	F	E	E	F	3.28400	3.36549	3.61166	4.05216	0.26903
–	L	L	F	S	E	E	F	–	–	3.60325	4.05198	–
–	L	L	F	L	E	E	F	–	–	3.60848	4.05007	–
215	L	L	L	F	E	E	E	3.28394	3.38503	3.61280	4.08244	0.28176
–	L	L	L	S	E	E	E	–	–	3.61831	4.11066	–
–	L	L	L	L	E	E	E	–	–	3.63715	4.14367	–
216	L	L	L	F	E	E	F	3.28336	3.38368	3.61308	4.12617	0.29228
–	L	L	L	S	E	E	F	–	–	3.61931	4.13409	–
–	L	L	L	L	E	E	F	–	–	3.63827	4.18445	–

F Language-Model Optimizer Crossover Experiments

We compare Adam, Lion, Muon, KL-SOAP, and Shampoo in the language-model setting of Section 4 at batch sizes $B \in \{128\text{K}, 512\text{K}, 1\text{M}, 2\text{M}\}$ tokens and a training budget of approximately 1.7B tokens. Each optimizer is tuned independently at every batch size under both decoupled weight decay and HyperBall norm control. We rank the selected configurations by terminal validation loss, with lower values indicating better performance. The SOAP variant we use throughout is KL-SOAP (Lin et al., 2026), which the main text calls SOAP for brevity. An optimizer crossover is a reversal of this ordering across batch sizes after retuning.

F.1 Implementations and common protocol

We follow the notation of Section E.1: $\eta_{\text{M}}, \lambda_{\text{M}}$ are the hidden-matrix learning rate and weight decay, and $\eta_{\text{A}}, \lambda_{\text{A}}$ are their auxiliary-Adam counterparts. Here the subscript M applies to whichever hidden-matrix optimizer is being compared; in Section E.1 it is MuonW. The auxiliary learning rates are $\eta_{\text{Emb.}}, \eta_{\text{Out.}}, \eta_{\text{Scalar}}$ for embeddings, the output projection, and scalar parameters, respectively. The search coordinate η_{A} scales all three together. As in Section E.1, μ is Muon momentum and β_1, β_2 are the auxiliary Adam retention coefficients. We distinguish the matrix optimizer's coefficients as $\beta_1^{\text{M}}, \beta_2^{\text{M}}$ and write β_{sh} for KL-SOAP's preconditioner retention coefficient. The auxiliary learning-rate cooldown occupies the fraction f_{cd} of training. We write terminal validation loss as L_{val}. Learning rates and weight decays in the tables are peak values; an additional subscript t denotes their values at update t.

All methods use the same model, token stream, validation set, precision, distributed training code, and learning-rate schedule shape. For a hidden matrix parameter W, the WD family applies decoupled weight decay as

$$W_{t+1} = (1 - \eta_{\text{M},t}\lambda_{\text{M},t})W_t - \eta_{\text{M},t}U_t,$$

where U_t is the update produced by the named optimizer. The HyperBall family instead applies the corresponding norm-preserving matrix projection and uses no matrix weight decay. We do not tune the preconditioner frequency, preconditioner initialization, or schedule family.

F.2 Optimizer definitions

For a hidden matrix $W \in \mathbb{R}^{m\times n}$ with gradient G_t, the boxes define its raw direction U_t and wrapper; scalar operations are entrywise and distributed details are omitted. Other parameter groups use bias-corrected Adam with their tabulated learning rates and absolute decay λ_{A}. HyperBall acts only on hidden matrices.

Algorithm 1 WD and HyperBall matrix wrappers

Let $(U_t, S_{t+1}) = \mathcal{D}_{\mathsf{O}}(G_t, S_t)$ and $\epsilon_H = 10^{-10}$. The two matrix families differ only in the final map:

$$\text{WD:}\quad W_{t+1} = (1 - \eta_{\text{M},t}\lambda_{\text{M},t})W_t - \eta_{\text{M},t}U_t,$$

$$\text{HyperBall:}\quad r_t = \|W_t\|_F, \qquad Z_t = W_t - \eta_{\text{M},t} r_t \frac{U_t}{\max(\|U_t\|_F, \epsilon_H)},$$

$$W_{t+1} = r_t \frac{Z_t}{\max(\|Z_t\|_F, \epsilon_H)}.$$

Algorithm 2 Adam and Lion raw directions

Adam ($M_0 = V_0 = 0$):

$$M_t = \beta_1^{\mathrm{M}} M_{t-1} + (1-\beta_1^{\mathrm{M}}) G_t, \qquad V_t = \beta_2^{\mathrm{M}} V_{t-1} + (1-\beta_2^{\mathrm{M}}) G_t^{\odot 2},$$
$$U_t = \frac{M_t/(1-(\beta_1^{\mathrm{M}})^t)}{\sqrt{V_t/(1-(\beta_2^{\mathrm{M}})^t)} + \epsilon}.$$

Lion ($M_0 = 0$):

$$U_t = \operatorname{sign}\Big(\beta_1^{\mathrm{M}} M_{t-1} + (1-\beta_1^{\mathrm{M}}) G_t\Big), \qquad M_t = \beta_2^{\mathrm{M}} M_{t-1} + (1-\beta_2^{\mathrm{M}}) G_t.$$

Algorithm 3 Muon raw direction with the Newton–Schulz map

1. Set $\omega = \mu$ and $M_0 = 0$. Form $M_t = \mu M_{t-1} + (1-\mu) G_t$ and $X = \omega M_t + (1-\omega) G_t$.
2. Cast X to bfloat16, transpose if $m > n$, and normalize $X \leftarrow X/(\|X\|_F + 10^{-7})$.
3. For 12 iterations with $(a, b, c) = (2, -1.5, 0.5)$, set $A = XX^\top$, $B = bA + cA^2$, and $X \leftarrow aX + BX$.
4. Undo the transpose and return $U_t = \sqrt{\max(1, m/n)}\, X$.

MuonW and MuonH share this Nesterov–Newton–Schulz direction (Jordan et al., 2024).

Algorithm 4 KL-SOAP raw direction

Initialize from G_1 and set $U_1 = 0$: $M_1 = V_1 = 0$, $C_L = G_1 G_1^\top/n$, $C_R = G_1^\top G_1/m$, and $(Q_s, \rho_s) = (\mathrm{eigvec}_{\mathrm{rev}}(C_s), 0.1^{-1/2}\mathbf{1})$ for $s \in \{L, R\}$.
Direction for $t \geq 2$:

$$\widehat{G}_t = Q_L^\top G_t Q_R,$$
$$M_t = \beta_1^{\mathrm{M}} M_{t-1} + (1-\beta_1^{\mathrm{M}}) \widehat{G}_t,$$
$$V_t = \beta_2^{\mathrm{M}} V_{t-1} + (1-\beta_2^{\mathrm{M}}) \widehat{G}_t^{\odot 2},$$
$$U_t = Q_L \frac{M_t}{\sqrt{V_t} + \epsilon} Q_R^\top \qquad \text{(no bias correction)}.$$

Refresh. After forming U_t, update C_L, C_R from the right- and left-whitened gradient covariances and ρ_L, ρ_R from $\mathrm{diag}(\widehat{G}_t)$; then set $Q_s \leftarrow \mathrm{qr}(C_s Q_s)$ and re-express M_t in the new basis for $s \in \{L, R\}$. This refresh is performed every step.

Algorithm 5 Distributed Shampoo raw direction

For each distributed matrix block, let A_t be the library's Adam-preconditioned graft direction. The Shampoo direction is

$$M_t = \beta_1^{\mathrm{M}} M_{t-1} + (1-\beta_1^{\mathrm{M}}) G_t,$$
$$C_L \leftarrow \beta_2^{\mathrm{M}} C_L + (1-\beta_2^{\mathrm{M}}) G_t G_t^\top/n, \qquad C_R \leftarrow \beta_2^{\mathrm{M}} C_R + (1-\beta_2^{\mathrm{M}}) G_t^\top G_t/m,$$
$$S_t = C_L^{-1/4} M_t C_R^{-1/4}, \qquad U_t = \frac{\|A_t\|_F}{\|S_t\|_F} S_t.$$

Here $C_s^{-1/4}$ is the eigen-based pseudoinverse fourth root. The fixed Shampoo implementation hyperparameters are

$$\texttt{precondition\_frequency} = 1, \qquad \epsilon_{\text{factor}} = 0,$$
$$\texttt{start\_preconditioning\_step} = -1, \qquad \epsilon_{\text{graft}} = 10^{-15},$$
$$\texttt{max\_preconditioner\_dim} = 8192.$$

F.3 Coordinate-descent protocol

The immediate-neighbor audit below applies to the WD campaign. Let $B_0 = 512\text{K}$ and $\kappa = B/B_0$, as in Section E.1. For each learning-rate coordinate η with reference value η_0, the initial transfer uses $\eta(B) = \eta_0\sqrt{\kappa}$. First-moment coefficients are held constant under this initial transfer, while second-moment and preconditioner coefficients, denoted generically by q with reference value $q_0 = q(B_0)$, use the time-scale anchor $q(B) = q_0^\kappa$. After this transfer, a one-coordinate sweep of an ordinary positive multiplicative coordinate x, such as a learning rate, evaluates

$$x' \in x\{1/2, 1/\sqrt{2}, 1, \sqrt{2}, 2\}.$$

The factors $1/\sqrt{2}$ and $\sqrt{2}$ are the adjacent grid points, while $1/2$ and 2 provide a wider search around the same center. For any momentum or EMA coefficient q (including μ and β_{sh}), the same factors act on its refresh rate rather than on q directly:

$$1 - q' = m(1 - q), \qquad m \in \{1/2, 1/\sqrt{2}, 1, \sqrt{2}, 2\}.$$

The cooldown coordinate instead uses the absolute grid

$$f_{\text{cd}} \in \{0, 0.2, 0.4, 0.6, 0.8\},$$

and the matrix weight-decay grid is

$$\lambda_{\text{M}} \in \{0, 0.01, 0.025, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, 1.0, 1.2,$$
$$1.4, 1.6, 1.8, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 4.4, 4.8, 5.2, 5.6, 6.0, 6.4, 6.8, 7.2, 8.0\}.$$

The grid is deliberately denser near zero. Weight decay is searched in these absolute units rather than multiplicatively.

At each iteration, all coordinates except one are fixed at the current hyperparameter configuration. A candidate replaces the current value only if its median validation loss improves by more than $\Delta = 0.002$. Gains in $[0.002, 0.006)$ require three matching repeats; larger gains require two, and every selected center has at least two repeats. After the last accepted move, we evaluate the immediate lower and upper grid neighbors of every active coordinate. We call an entry *locally closed* when none of these neighbors exceeds the acceptance threshold under the multi-seed repeat policy.

Method	Tuned coordinates	Count
MuonW	$\eta_{\text{M}}, \mu, \lambda_{\text{M}}, \eta_{\text{A}}, \beta_1, \beta_2, f_{\text{cd}}$	7
KL-SOAP-W	$\eta_{\text{M}}, \beta_1^{\text{M}}, \beta_2^{\text{M}}, \beta_{\text{sh}}, \lambda_{\text{M}}, \eta_{\text{A}}, \beta_1, \beta_2, f_{\text{cd}}$	9
ShampooW	$\eta_{\text{M}}, \lambda_{\text{M}}, \eta_{\text{A}}, \beta_1, \beta_2, f_{\text{cd}}$	6
AdamW	$\eta_{\text{M}}, \beta_1^{\text{M}}, \beta_2^{\text{M}}, \lambda_{\text{M}}, \eta_{\text{A}}, \beta_1, \beta_2, f_{\text{cd}}$	8
LionW	$\eta_{\text{M}}, \beta_1^{\text{M}}, \beta_2^{\text{M}}, \lambda_{\text{M}}, \eta_{\text{A}}, \beta_1, \beta_2, f_{\text{cd}}$	8

Table 10: Coordinates tuned in the WD campaign.

The complete one-coordinate audit is collected in Section F.7.

HyperBall search schedule. HyperBall uses the same coordinate-descent move criterion, with optimizer-specific learning rates, momenta, and auxiliary cooldown swept around each center.

F.4 FINAL WEIGHT-DECAY HYPERPARAMETERS

The lowest-loss optimizer changes from KL-SOAP to Shampoo as batch size increases. Under decoupled weight decay, KL-SOAP-W attains the lowest terminal validation loss at 128K, 3.2536, compared with 3.2650 for ShampooW. At 2M, ShampooW attains the lowest loss, 3.3622, while KL-SOAP-W reaches 3.3833. This reversal persists despite independent hyperparameter tuning at each batch size (Figure 1 and table 11).

Tables 11 and 12 together specify the selected configurations in absolute hyperparameter units. The losses are terminal validation losses for the selected configurations, and boldface marks the numerical minimum at each batch size. Shampoo's matrix moment coefficients are fixed rather than tuned; blank entries denote coefficients that are not part of an optimizer. Hyperparameters are shown to three significant figures and losses to four decimal places.

				Momentum / preconditioner					
B	Method	Steps	η_{M}	μ	β_1^{M}	β_2^{M}	β_{sh}	λ_{M}	L_{val}
128K	MuonW	13000	0.0177	0.95	–	–	–	0.025	3.2656
128K	KL-SOAP-W	13000	0.0045	–	0.965	0.974	0.948	0.1	**3.2536**
128K	ShampooW	13000	0.005	–	0.9	0.9	–	0.05	3.2650
128K	AdamW	13000	0.00075	–	0.94	0.949	–	0.3	3.3151
128K	LionW	13000	5×10^{-5}	–	0.9	0.99	–	8.6	3.3176
512K	MuonW	3250	0.025	0.95	–	–	–	0.05	3.2774
512K	KL-SOAP-W	3250	0.0127	–	0.9	0.859	0.859	0.15	**3.2761**
512K	ShampooW	3250	0.00707	–	0.9	0.9	–	0.15	3.2809
512K	AdamW	3250	0.0015	–	0.9	0.859	–	0.6	3.3392
512K	LionW	3250	0.0001	–	0.95	0.96	–	0.05	3.3907
1M	MuonW	1625	0.025	0.95	–	–	–	0.1	3.3108
1M	KL-SOAP-W	1625	0.00636	–	0.929	0.866	0.866	0.6	**3.2974**
1M	ShampooW	1625	0.00707	–	0.9	0.9	–	0.3	3.3090
1M	AdamW	1625	0.0015	–	0.859	0.931	–	0.8	3.4028
1M	LionW	1625	0.000141	–	0.95	0.944	–	0.15	3.4452
2M	MuonW	813	0.025	0.95	–	–	–	0.1	3.3659
2M	KL-SOAP-W	813	0.036	–	0.929	0.828	0.878	0.4	3.3833
2M	ShampooW	813	0.01	–	0.9	0.9	–	0.5	**3.3622**
2M	AdamW	813	0.00212	–	0.9	0.869	–	1.8	3.4781
2M	LionW	813	0.0002	–	0.9	0.889	–	0.3	3.5562

Table 11: Final WD hidden-matrix hyperparameters and terminal validation loss.

		Auxiliary learning rates			Adam coefficients			
B	Method	$\eta_{\text{Emb.}}$	$\eta_{\text{Out.}}$	η_{Scalar}	β_1	β_2	f_{cd}	λ_{A}
128K	MuonW	0.495	0.00283	0.0106	0.8	0.994	0.8	0.001
128K	KL-SOAP-W	0.15	0.00156	0.005	0.434	0.995	0.4	0
128K	ShampooW	1.2	0.0125	0.04	0.52	0.995	0.6	0
128K	AdamW	0.849	0.00884	0.0283	0.717	0.995	0.4	0
128K	LionW	0.287	0.00299	0.00958	0.859	0.995	0.6	0
512K	MuonW	0.7	0.004	0.015	0.8	0.965	0.7	0.001
512K	KL-SOAP-W	0.849	0.00884	0.0283	0.8	0.975	0.4	0
512K	ShampooW	0.849	0.00884	0.0283	0.8	0.982	0.6	0
512K	AdamW	0.849	0.00884	0.0283	0.859	0.975	0.4	0
512K	LionW	0.212	0.00221	0.00707	0.717	0.95	0.2	0
1M	MuonW	0.99	0.00566	0.0212	0.8	0.931	0.8	0.001
1M	KL-SOAP-W	0.6	0.00625	0.02	0.8	0.951	0.6	0
1M	ShampooW	0.849	0.00884	0.0283	0.8	0.971	0.6	0
1M	AdamW	0.6	0.00625	0.02	0.859	0.902	0.4	0
1M	LionW	0.3	0.00313	0.01	0.6	0.902	0.2	0
2M	MuonW	0.7	0.004	0.015	0.859	0.907	0.4	0.001
2M	KL-SOAP-W	0.6	0.00625	0.02	0.8	0.934	0.4	0
2M	ShampooW	0.6	0.00625	0.02	0.8	0.944	0.6	0
2M	AdamW	3.39	0.0354	0.113	0.929	0.967	0.6	0
2M	LionW	0.3	0.00313	0.01	0.8	0.954	0.2	0

Table 12: Final WD auxiliary-Adam hyperparameters.

F.5 Optimizer crossover under HyperBall

The change in winner also occurs under HyperBall. With the norm-preserving map of Algorithm 1 and independent retuning, KL-SOAP-H has the lowest loss at 128K, 3.2585, whereas ShampooH has the lowest loss at 2M, 3.3390. At 1M, the two are nearly tied, with reported losses of 3.2962 and 3.2965, respectively. Thus the same endpoint reversal appears under both weight decay and HyperBall (Figure 16 and table 13).

Other optimizer pairs also reverse their ordering. MuonH has lower loss than ShampooH at 128K, but ShampooH has lower loss at all three larger batches. Their difference at 512K is only about 0.0007. LionH has lower loss than AdamH at 128K, while AdamH has lower loss at every larger batch. Both remain above MuonH, KL-SOAP-H, and ShampooH in validation loss at every tested batch size.

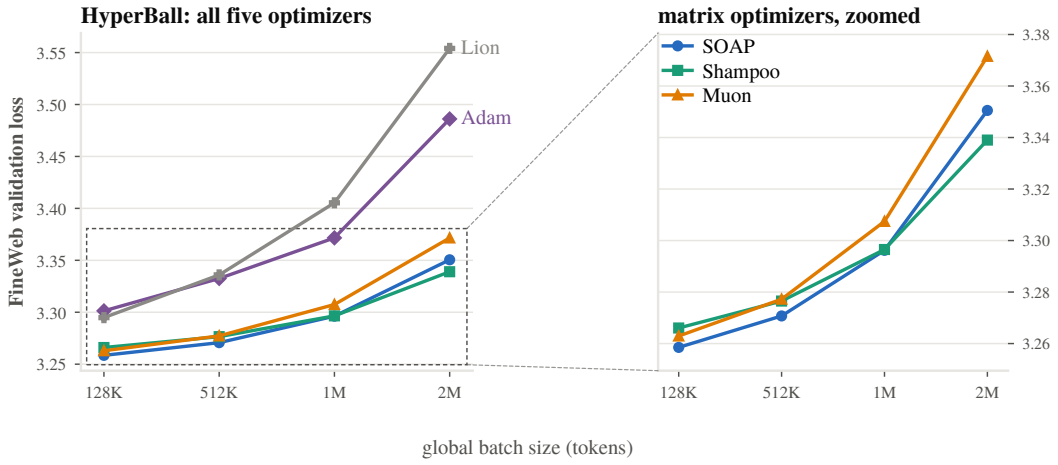

Figure 16: **The lowest-loss optimizer changes with batch size under HyperBall.** Language-model pretraining with an approximately 1.7B-token budget and independent tuning of every optimizer at every batch size. Left: all five optimizers. Right: Muon, KL-SOAP (labeled SOAP), and Shampoo on a zoomed loss axis. KL-SOAP has the lowest reported loss from 128K through 1M, with a gap of only about 0.0003 to Shampoo at 1M; Shampoo has the lowest loss at 2M.

F.6 Final HyperBall hyperparameters

Tables 13 and 14 specify the selected HyperBall configurations in absolute hyperparameter units. The loss column gives the terminal validation losses plotted in Figure 16. These configurations use no matrix or auxiliary weight decay. As in the weight-decay tables, hyperparameters are shown to three significant figures and losses to four decimal places, with boldface marking the numerical minimum at each batch size.

				Momentum / preconditioner				
B	Method	Steps	η_{M}	μ	β_1^{M}	β_2^{M}	β_{sh}	L_{val}
128K	MuonH	13000	0.009	0.95	–	–	–	3.2629
128K	KL-SOAP-H	13000	0.009	–	0.929	0.974	0.948	**3.2585**
128K	ShampooH	13000	0.009	–	0.922	0.853	–	3.2660
128K	AdamH	13000	0.009	–	0.94	0.949	–	3.3013
128K	LionH	13000	0.00636	–	0.929	0.987	–	3.2948
512K	MuonH	3250	0.018	0.95	–	–	–	3.2772
512K	KL-SOAP-H	3250	0.018	–	0.929	0.859	0.9	**3.2707**
512K	ShampooH	3250	0.018	–	0.929	0.9	–	3.2765
512K	AdamH	3250	0.018	–	0.929	0.9	–	3.3325
512K	LionH	3250	0.018	–	0.929	0.95	–	3.3360
1M	MuonH	1625	0.0255	0.95	–	–	–	3.3074
1M	KL-SOAP-H	1625	0.0255	–	0.929	0.866	0.866	**3.2962**
1M	ShampooH	1625	0.0255	–	0.929	0.848	–	3.2965
1M	AdamH	1625	0.018	–	0.965	0.902	–	3.3716
1M	LionH	1625	0.036	–	0.9	0.902	–	3.4053
2M	MuonH	813	0.036	0.95	–	–	–	3.3715
2M	KL-SOAP-H	813	0.036	–	0.929	0.656	0.878	3.3505
2M	ShampooH	813	0.036	–	0.929	0.725	–	**3.3390**
2M	AdamH	813	0.0255	–	0.965	0.907	–	3.4861
2M	LionH	813	0.036	–	0.9	0.907	–	3.5540

Table 13: Final HyperBall hidden-matrix hyperparameters and terminal validation loss.

		Auxiliary learning rates			Adam coefficients		
B	Method	$\eta_{\text{Emb.}}$	$\eta_{\text{Out.}}$	η_{Scalar}	β_1	β_2	f_{cd}
128K	MuonH	0.3	0.00313	0.01	0.8	0.994	0.4
128K	KL-SOAP-H	0.3	0.00313	0.01	0.434	0.994	0.4
128K	ShampooH	1.2	0.0125	0.04	0.52	0.995	0.4
128K	AdamH	0.424	0.00442	0.0141	0.717	0.995	0.4
128K	LionH	0.3	0.00313	0.01	0.717	0.994	0.4
512K	MuonH	0.6	0.00625	0.02	0.859	0.975	0.4
512K	KL-SOAP-H	0.6	0.00625	0.02	0.8	0.975	0.8
512K	ShampooH	0.849	0.00884	0.0283	0.8	0.982	0.4
512K	AdamH	0.6	0.00625	0.02	0.8	0.975	0.4
512K	LionH	0.424	0.00442	0.0141	0.8	0.983	0.4
1M	MuonH	0.6	0.00625	0.02	0.9	0.951	0.4
1M	KL-SOAP-H	0.6	0.00625	0.02	0.8	0.951	0.6
1M	ShampooH	0.849	0.00884	0.0283	0.8	0.971	0.4
1M	AdamH	0.3	0.00313	0.01	0.8	0.959	0.25
1M	LionH	0.6	0.00625	0.02	0.9	0.976	0.4
2M	MuonH	0.6	0.00625	0.02	0.9	0.934	0.4
2M	KL-SOAP-H	0.6	0.00625	0.02	0.8	0.907	0.6
2M	ShampooH	0.6	0.00625	0.02	0.8	0.944	0.4
2M	AdamH	0.424	0.00442	0.0141	0.8	0.921	0.25
2M	LionH	0.6	0.00625	0.02	0.9	0.954	0.2

Table 14: Final HyperBall auxiliary-Adam hyperparameters.

F.7 Complete One-Coordinate Sweep Results

The 492-run final-configuration audit is supplemented here with 643 terminal arms recovered from the coordinate-descent ledgers, covering 40 optimizer–batch settings. A historical arm is retained only when every active coordinate other than the swept coordinate matches the accepted final hyperparameter configuration (up to ledger serialization precision). We use the latest such sweep background and resolve duplicate retries by terminal-evidence recency without consulting validation loss. All included arms reached their configured final validation step. This snapshot includes 152 strictly terminal arms from the boundary-completion campaign: 76 initial missing-neighbor arms and 76 subsequent outward boundary extensions. Incomplete arms without a persisted exact-terminal marker are omitted. The first row of each optimizer–batch block gives the accepted center in absolute hyperparameter units. Every later row changes only the displayed coordinate while freezing every other coordinate at that accepted center; a dash means unchanged from the first row. The machine-readable source records each historical arm's full frozen background, campaign stamp, source ledger, and job ID.

The three auxiliary learning rates are tied during search, so the tables show the absolute embedding rate $\eta_{\text{Emb.}}$ while the output-projection and scalar rates, $\eta_{\text{Out.}}$ and η_{Scalar}, change by the same factor. The reported quantity is terminal validation loss, not the best loss observed earlier in training.

These tables are a local sensitivity record rather than a new configuration-selection rule. In particular, a numerically lower single-seed arm is not accepted unless it also satisfies the multi-seed threshold policy in Section F.3.

F.7.1 HyperBall sweeps

MuonH

Table 15: Hyperparameter ablation for MuonH at batch size $B = 128K$.

η_{M}	μ	$\eta_{\text{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.009	0.95	0.3	0.8	0.99363	0.4	3.26346
0.0045	–	–	–	–	–	3.29906
0.006364	–	–	–	–	–	3.27465
0.012728	–	–	–	–	–	3.27016
0.018	–	–	–	–	–	3.29629
–	0.975	–	–	–	–	3.27140
–	0.96464	–	–	–	–	3.26435
–	0.92929	–	–	–	–	3.26798
–	0.9	–	–	–	–	3.27793
–	–	0.15	–	–	–	3.26878
–	–	0.21213	–	–	–	3.26640
–	–	0.42426	–	–	–	3.26341
–	–	0.6	–	–	–	3.26543
–	–	0.84853	–	–	–	3.26400
–	–	–	0.9	–	–	3.26636
–	–	–	0.85858	–	–	3.26464
–	–	–	0.71716	–	–	3.26304
–	–	–	0.6	–	–	3.26358
–	–	–	–	0.99681	–	3.26598
–	–	–	–	0.9955	–	3.26300
–	–	–	–	0.99099	–	3.26375
–	–	–	–	0.98726	–	3.26441
–	–	–	–	0.98198	–	3.26634
–	–	–	–	0.97452	–	3.27010
–	–	–	–	–	0	3.29470
–	–	–	–	–	0.2	3.26584
–	–	–	–	–	0.6	3.26324
–	–	–	–	–	0.8	3.26470

Table 16: Hyperparameter ablation for MuonH at batch size $B = 512K$.

η_{M}	μ	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.018	0.95	0.6	0.85858	0.975	0.4	3.27768
0.009	–	–	–	–	–	3.31502
0.012728	–	–	–	–	–	3.28879
0.025456	–	–	–	–	–	3.28412
0.036	–	–	–	–	–	3.31009
–	0.975	–	–	–	–	3.28664
–	0.96464	–	–	–	–	3.27986
–	0.92929	–	–	–	–	3.28260
–	0.9	–	–	–	–	3.29317
–	–	0.3	–	–	–	3.28083
–	–	0.42426	–	–	–	3.27904
–	–	0.84853	–	–	–	3.27802
–	–	1.2	–	–	–	3.28040
–	–	–	0.95	–	–	3.27875
–	–	–	0.92929	–	–	3.27751
–	–	–	0.9	–	–	3.27760
–	–	–	0.8	–	–	3.27826
–	–	–	0.71716	–	–	3.28116
–	–	–	–	0.9875	–	3.27784
–	–	–	–	0.98232	–	3.27735
–	–	–	–	0.96464	–	3.27817
–	–	–	–	0.95	–	3.27851
–	–	–	–	0.92929	–	3.28065
–	–	–	–	0.9	–	3.28232
–	–	–	–	–	0	3.30830
–	–	–	–	–	0.2	3.28022
–	–	–	–	–	0.6	3.27805
–	–	–	–	–	0.8	3.27882

Table 17: Hyperparameter ablation for MuonH at batch size $B = 1M$.

η_{M}	μ	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.025456	0.95	0.6	0.9	0.95125	0.4	3.30679
0.012728	–	–	–	–	–	3.34188
0.018	–	–	–	–	–	3.31511
0.036	–	–	–	–	–	3.31483
0.050912	–	–	–	–	–	3.33885
–	0.975	–	–	–	–	3.31714
–	0.96464	–	–	–	–	3.31027
–	0.92929	–	–	–	–	3.31138
–	0.9	–	–	–	–	3.32204
–	–	0.21213	–	–	–	3.31597
–	–	0.3	–	–	–	3.31402
–	–	0.42426	–	–	–	3.30766
–	–	0.84853	–	–	–	3.30833
–	–	1.2	–	–	–	3.31302
–	–	–	0.95	–	–	3.31025
–	–	–	0.92929	–	–	3.30805
–	–	–	0.85858	–	–	3.30808
–	–	–	0.8	–	–	3.31023
–	–	–	–	0.98276	–	3.30706
–	–	–	–	0.97562	–	3.30668
–	–	–	–	0.96553	–	3.30731
–	–	–	–	0.93106	–	3.30810

MuonH, $B = 1M$

η_{M}	μ	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	–	0.9025	–	3.30767
–	–	–	–	–	0	3.33445
–	–	–	–	–	0.2	3.30956
–	–	–	–	–	0.6	3.30758
–	–	–	–	–	0.8	3.31045

Table 18: Hyperparameter ablation for MuonH at batch size $B = 2M$.

η_{M}	μ	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.036	0.95	0.6	0.9	0.93442	0.4	3.37098
0.018	–	–	–	–	–	3.40605
0.025456	–	–	–	–	–	3.38018
0.050912	–	–	–	–	–	3.38112
0.072	–	–	–	–	–	3.40408
–	0.96464	–	–	–	–	3.37684
–	0.92929	–	–	–	–	3.37373
–	–	0.42426	–	–	–	3.37348
–	–	0.84853	–	–	–	3.37688
–	–	–	0.92929	–	–	3.37308
–	–	–	0.85858	–	–	3.37239
–	–	–	–	0.95363	–	3.37119
–	–	–	–	0.90725	–	3.37120
–	–	–	–	0.86884	–	3.37315
–	–	–	–	–	0.2	3.37237
–	–	–	–	–	0.6	3.37317

KL-SOAP-H

Table 19: Hyperparameter ablation for KL-SOAP-H at batch size $B = 128K$.

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.009	0.92929	0.974	0.94801	0.3	0.43431	0.99363	0.4	3.25728
0.0045	–	–	–	–	–	–	–	3.29786
0.006364	–	–	–	–	–	–	–	3.27114
0.012728	–	–	–	–	–	–	–	3.26005
0.018	–	–	–	–	–	–	–	3.27850
–	0.96464	–	–	–	–	–	–	3.25890
–	0.95	–	–	–	–	–	–	3.25503
–	0.9	–	–	–	–	–	–	3.26334
–	–	0.98162	–	–	–	–	–	3.25779
–	–	0.96324	–	–	–	–	–	3.25726
–	–	0.94801	–	–	–	–	–	3.25816
–	–	–	0.96324	–	–	–	–	3.25842
–	–	–	0.92647	–	–	–	–	3.25684
–	–	–	0.89601	–	–	–	–	3.25809
–	–	–	–	0.21213	–	–	–	3.25777
–	–	–	–	0.42426	–	–	–	3.25725
–	–	–	–	0.6	–	–	–	3.25634
–	–	–	–	0.84853	–	–	–	3.25754
–	–	–	–	–	0.85858	–	–	3.26071
–	–	–	–	–	0.8	–	–	3.25735
–	–	–	–	–	0.6	–	–	3.25681
–	–	–	–	–	0.2	–	–	3.25923

KL-SOAP-H, $B = 128K$

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	–	–	–	0.99681	–	3.25960
–	–	–	–	–	–	0.9955	–	3.25682
–	–	–	–	–	–	0.99099	–	3.25801
–	–	–	–	–	–	0.98726	–	3.25760
–	–	–	–	–	–	–	0	3.28843
–	–	–	–	–	–	–	0.2	3.26051
–	–	–	–	–	–	–	0.6	3.25730
–	–	–	–	–	–	–	0.8	3.25928

Table 20: Hyperparameter ablation for KL-SOAP-H at batch size $B = 512K$.

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.018	0.92929	0.85858	0.9	0.6	0.8	0.975	0.8	3.26964
0.009	–	–	–	–	–	–	–	3.30243
0.012728	–	–	–	–	–	–	–	3.28109
0.025456	–	–	–	–	–	–	–	3.27408
0.036	–	–	–	–	–	–	–	3.29737
–	0.975	–	–	–	–	–	–	3.29182
–	0.96464	–	–	–	–	–	–	3.27817
–	0.95	–	–	–	–	–	–	3.27292
–	0.9	–	–	–	–	–	–	3.27516
–	–	0.9	–	–	–	–	–	3.27167
–	–	0.8	–	–	–	–	–	3.27101
–	–	–	0.92929	–	–	–	–	3.27126
–	–	–	0.85858	–	–	–	–	3.27135
–	–	–	–	0.42426	–	–	–	3.27290
–	–	–	–	0.84853	–	–	–	3.27163
–	–	–	–	–	0.9	–	–	3.26989
–	–	–	–	–	0.85858	–	–	3.26911
–	–	–	–	–	0.71716	–	–	3.27080
–	–	–	–	–	–	0.98232	–	3.27077
–	–	–	–	–	–	0.96464	–	3.27250
–	–	–	–	–	–	–	0.4	3.27116
–	–	–	–	–	–	–	0.6	3.26846
–	–	–	–	–	–	–	1	3.27039

Table 21: Hyperparameter ablation for KL-SOAP-H at batch size $B = 1M$.

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.025456	0.92929	0.86565	0.86565	0.6	0.8	0.95125	0.6	3.29536
0.012728	–	–	–	–	–	–	–	3.32611
0.018	–	–	–	–	–	–	–	3.30198
0.036	–	–	–	–	–	–	–	3.30162
0.050912	–	–	–	–	–	–	–	3.32237
–	0.95	–	–	–	–	–	–	3.30005
–	0.9	–	–	–	–	–	–	3.29841
–	–	0.93282	–	–	–	–	–	3.29737
–	–	0.905	–	–	–	–	–	3.29432
–	–	0.81	–	–	–	–	–	3.29574
–	–	0.62	–	–	–	–	–	3.29691
–	–	0.4626	–	–	–	–	–	3.29764
–	–	–	0.905	–	–	–	–	3.29452
–	–	–	0.81	–	–	–	–	3.29379

KL-SOAP-H, $B = 1M$

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	0.7313	–	–	–	–	3.30078
–	–	–	–	0.3	–	–	–	3.29809
–	–	–	–	0.42426	–	–	–	3.29490
–	–	–	–	0.84853	–	–	–	3.30146
–	–	–	–	–	0.9	–	–	3.29544
–	–	–	–	–	0.85858	–	–	3.29404
–	–	–	–	–	0.71716	–	–	3.29805
–	–	–	–	–	–	0.97562	–	3.29541
–	–	–	–	–	–	0.96553	–	3.29478
–	–	–	–	–	–	0.93106	–	3.29639
–	–	–	–	–	–	–	0.4	3.29679
–	–	–	–	–	–	–	0.8	3.29528
–	–	–	–	–	–	–	1	3.29743

Table 22: Hyperparameter ablation for KL-SOAP-H at batch size $B = 2M$.

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.036	0.92929	0.6561	0.87841	0.6	0.8	0.90725	0.6	3.35062
0.018	–	–	–	–	–	–	–	3.38592
0.025456	–	–	–	–	–	–	–	3.36252
0.050912	–	–	–	–	–	–	–	3.35583
0.072	–	–	–	–	–	–	–	3.38261
–	0.95	–	–	–	–	–	–	3.35416
–	0.9	–	–	–	–	–	–	3.35290
–	–	0.75683	–	–	–	–	–	3.34943
–	–	0.51365	–	–	–	–	–	3.34819
–	–	0.3122	–	–	–	–	–	3.35210
–	–	–	0.93921	–	–	–	–	3.35539
–	–	–	0.91402	–	–	–	–	3.35029
–	–	–	0.82805	–	–	–	–	3.35193
–	–	–	–	0.42426	–	–	–	3.35165
–	–	–	–	0.84853	–	–	–	3.35283
–	–	–	–	–	0.9	–	–	3.34920
–	–	–	–	–	0.85858	–	–	3.34857
–	–	–	–	–	0.71716	–	–	3.35331
–	–	–	–	–	–	0.93442	–	3.34938
–	–	–	–	–	–	0.86884	–	3.34752
–	–	–	–	–	–	0.81451	–	3.35052
–	–	–	–	–	–	–	0.2	3.34980
–	–	–	–	–	–	–	0.4	3.34885
–	–	–	–	–	–	–	0.8	3.35197

ShampooH

Table 23: Hyperparameter ablation for ShampooH at batch size $B = 128K$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.009	0.92222	0.85294	1.2	0.52	0.9955	0.4	3.26617
0.0045	–	–	–	–	–	–	3.30292
0.006364	–	–	–	–	–	–	3.27724
0.012728	–	–	–	–	–	–	3.26965
0.018	–	–	–	–	–	–	3.29102

ShampooH, $B = 128K$

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	0.9725	–	–	–	–	–	3.27934
–	0.96111	–	–	–	–	–	3.27001
–	0.945	–	–	–	–	–	3.26581
–	0.89	–	–	–	–	–	3.27140
–	–	0.92647	–	–	–	–	3.26603
–	–	0.89601	–	–	–	–	3.26571
–	–	0.79203	–	–	–	–	3.26656
–	–	–	0.84853	–	–	–	3.26672
–	–	–	1.6971	–	–	–	3.26749
–	–	–	–	0.66059	–	–	3.26792
–	–	–	–	0.32118	–	–	3.26924
–	–	–	–	–	0.99681	–	3.26945
–	–	–	–	–	0.99363	–	3.26595
–	–	–	–	–	0.99099	–	3.26633
–	–	–	–	–	–	0.2	3.26838
–	–	–	–	–	–	0.6	3.26544
–	–	–	–	–	–	0.8	3.26565

Table 24: Hyperparameter ablation for ShampooH at batch size $B = 512K$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.018	0.92929	0.9	0.84853	0.8	0.98232	0.4	3.27578
0.009	–	–	–	–	–	–	3.31081
0.012728	–	–	–	–	–	–	3.28723
0.025456	–	–	–	–	–	–	3.28195
0.036	–	–	–	–	–	–	3.30230
–	0.96464	–	–	–	–	–	3.28247
–	0.95	–	–	–	–	–	3.27802
–	0.9	–	–	–	–	–	3.28012
–	0.85858	–	–	–	–	–	3.28888
–	–	0.95	–	–	–	–	3.28006
–	–	0.92929	–	–	–	–	3.27877
–	–	0.915	–	–	–	–	3.27786
–	–	0.885	–	–	–	–	3.27714
–	–	0.85858	–	–	–	–	3.27577
–	–	0.8	–	–	–	–	3.27737
–	–	–	0.42426	–	–	–	3.27871
–	–	–	0.6	–	–	–	3.27734
–	–	–	1.0182	–	–	–	3.27825
–	–	–	1.2	–	–	–	3.27865
–	–	–	1.6971	–	–	–	3.28087
–	–	–	–	0.9	–	–	3.27947
–	–	–	–	0.85858	–	–	3.27885
–	–	–	–	0.71716	–	–	3.27602
–	–	–	–	0.6	–	–	3.27953
–	–	–	–	–	0.99116	–	3.27649
–	–	–	–	–	0.9875	–	3.27634
–	–	–	–	–	0.98586	–	3.27741
–	–	–	–	–	0.985	–	3.27589
–	–	–	–	–	0.98	–	3.27763
–	–	–	–	–	0.97879	–	3.27786
–	–	–	–	–	0.975	–	3.27610
–	–	–	–	–	0.96464	–	3.27666
–	–	–	–	–	0.92929	–	3.28011

ShampooH, $B = 512K$

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	–	–	–	0	3.30509
–	–	–	–	–	–	0.2	3.27948
–	–	–	–	–	–	0.6	3.27620
–	–	–	–	–	–	0.8	3.27758

Table 25: Hyperparameter ablation for ShampooH at batch size $B = 1M$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.025456	0.92929	0.848	0.84853	0.8	0.97075	0.4	3.29641
0.012728	–	–	–	–	–	–	3.33305
0.018	–	–	–	–	–	–	3.30733
0.036	–	–	–	–	–	–	3.30217
0.050912	–	–	–	–	–	–	3.32518
–	0.96464	–	–	–	–	–	3.30594
–	0.95	–	–	–	–	–	3.29847
–	0.9	–	–	–	–	–	3.30112
–	0.85858	–	–	–	–	–	3.30898
–	–	0.89252	–	–	–	–	3.29759
–	–	0.867	–	–	–	–	3.29550
–	–	0.8575	–	–	–	–	3.29623
–	–	0.85275	–	–	–	–	3.29619
–	–	0.8385	–	–	–	–	3.29599
–	–	0.78504	–	–	–	–	3.29664
–	–	–	0.42426	–	–	–	3.29788
–	–	–	0.6	–	–	–	3.29851
–	–	–	1.2	–	–	–	3.29751
–	–	–	1.6971	–	–	–	3.29925
–	–	–	–	0.9	–	–	3.29644
–	–	–	–	0.85858	–	–	3.29723
–	–	–	–	0.71716	–	–	3.29753
–	–	–	–	0.6	–	–	3.30094
–	–	–	–	–	0.97932	–	3.29649
–	–	–	–	–	0.97562	–	3.29584
–	–	–	–	–	0.97319	–	3.29559
–	–	–	–	–	0.96831	–	3.29591
–	–	–	–	–	0.96588	–	3.29572
–	–	–	–	–	0.96553	–	3.29506
–	–	–	–	–	0.961	–	3.29544
–	–	–	–	–	0.95863	–	3.29598
–	–	–	–	–	0.95125	–	3.29590
–	–	–	–	–	–	0.2	3.29782
–	–	–	–	–	–	0.6	3.29663
–	–	–	–	–	–	0.8	3.29755

Table 26: Hyperparameter ablation for ShampooH at batch size $B = 2M$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.036	0.92929	0.72488	0.6	0.8	0.94435	0.4	3.33928
0.018	–	–	–	–	–	–	3.37605
0.025456	–	–	–	–	–	–	3.34886
0.050912	–	–	–	–	–	–	3.34763
0.072	–	–	–	–	–	–	3.37374
–	0.95	–	–	–	–	–	3.34432
–	0.9	–	–	–	–	–	3.34161

ShampooH, $B = 2M$

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	0.86244	–	–	–	–	3.34115
–	–	0.80546	–	–	–	–	3.34013
–	–	0.61092	–	–	–	–	3.34285
–	–	–	0.42426	–	–	–	3.34033
–	–	–	0.84853	–	–	–	3.34150
–	–	–	1.6971	–	–	–	3.36828
–	–	–	2.4	–	–	–	3.35629
–	–	–	–	0.9	–	–	3.34651
–	–	–	–	0.85858	–	–	3.34282
–	–	–	–	0.71716	–	–	3.34073
–	–	–	–	–	0.97218	–	3.33963
–	–	–	–	–	0.96065	–	3.33901
–	–	–	–	–	0.9213	–	3.33973
–	–	–	–	–	–	0.2	3.34120
–	–	–	–	–	–	0.6	3.34076

AdamH

Table 27: Hyperparameter ablation for AdamH at batch size $B = 128K$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.009	0.94	0.94903	0.42426	0.71716	0.9955	0.4	3.30144
0.0045	–	–	–	–	–	–	3.32743
0.006364	–	–	–	–	–	–	3.30696
0.012728	–	–	–	–	–	–	3.31867
0.018	–	–	–	–	–	–	3.37667
–	0.96464	–	–	–	–	–	3.30978
–	0.95757	–	–	–	–	–	3.30506
–	0.955	–	–	–	–	–	3.30335
–	0.95	–	–	–	–	–	3.30173
–	0.9475	–	–	–	–	–	3.30244
–	0.945	–	–	–	–	–	3.30143
–	0.9425	–	–	–	–	–	3.30122
–	0.9375	–	–	–	–	–	3.30219
–	0.935	–	–	–	–	–	3.30232
–	0.91515	–	–	–	–	–	3.30470
–	0.85858	–	–	–	–	–	3.31708
–	–	0.96396	–	–	–	–	3.30185
–	–	0.92792	–	–	–	–	3.30117
–	–	0.89807	–	–	–	–	3.30243
–	–	–	0.3	–	–	–	3.30220
–	–	–	0.6	–	–	–	3.30327
–	–	–	0.84853	–	–	–	3.30713
–	–	–	–	0.85858	–	–	3.30895
–	–	–	–	0.8	–	–	3.30489
–	–	–	–	0.6	–	–	3.30310
–	–	–	–	0.43431	–	–	3.30172
–	–	–	–	–	0.99681	–	3.30688
–	–	–	–	–	0.99363	–	3.30115
–	–	–	–	–	0.99099	–	3.30242
–	–	–	–	–	–	0.2	3.30295
–	–	–	–	–	–	0.6	3.30273

Table 28: Hyperparameter ablation for AdamH at batch size $B = 512K$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.018	0.92929	0.9	0.6	0.8	0.975	0.4	3.33227
0.009	–	–	–	–	–	–	3.36880
0.012728	–	–	–	–	–	–	3.34160
0.025456	–	–	–	–	–	–	3.34476
0.036	–	–	–	–	–	–	3.38899
–	0.96464	–	–	–	–	–	3.35050
–	0.95	–	–	–	–	–	3.33188

AdamH, $B = 512K$

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	0.9	–	–	–	–	–	3.34178
–	0.85858	–	–	–	–	–	3.35278
–	0.8	–	–	–	–	–	3.37392
–	–	0.95	–	–	–	–	3.33508
–	–	0.92929	–	–	–	–	3.33403
–	–	0.92	–	–	–	–	3.33574
–	–	0.91	–	–	–	–	3.33441
–	–	0.88	–	–	–	–	3.33438
–	–	0.85858	–	–	–	–	3.33293
–	–	0.8	–	–	–	–	3.33727
–	–	–	0.15	–	–	–	3.34334
–	–	–	0.21213	–	–	–	3.33908
–	–	–	0.3	–	–	–	3.33450
–	–	–	0.42426	–	–	–	3.33273
–	–	–	0.48	–	–	–	3.33297
–	–	–	0.84853	–	–	–	3.33310
–	–	–	1.2	–	–	–	3.33941
–	–	–	–	0.9	–	–	3.33600
–	–	–	–	0.85858	–	–	3.33366
–	–	–	–	0.71716	–	–	3.33481
–	–	–	–	0.6	–	–	3.33843
–	–	–	–	–	0.9875	–	3.33413
–	–	–	–	–	0.98625	–	3.33435
–	–	–	–	–	0.985	–	3.33434
–	–	–	–	–	0.9825	–	3.33289
–	–	–	–	–	0.98232	–	3.33338
–	–	–	–	–	0.98125	–	3.33371
–	–	–	–	–	0.98	–	3.33402
–	–	–	–	–	0.97	–	3.33461
–	–	–	–	–	0.96464	–	3.33401
–	–	–	–	–	0.95	–	3.33696
–	–	–	–	–	–	0	3.35882
–	–	–	–	–	–	0.2	3.33429
–	–	–	–	–	–	0.6	3.33440
–	–	–	–	–	–	0.8	3.33778

Table 29: Hyperparameter ablation for AdamH at batch size $B = 1M$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.018	0.96464	0.9025	0.3	0.8	0.95863	0.25	3.37114
0.006364	–	–	–	–	–	–	3.45697
0.009	–	–	–	–	–	–	3.41310
0.012728	–	–	–	–	–	–	3.38258
0.025456	–	–	–	–	–	–	3.37975
0.036	–	–	–	–	–	–	3.41154
–	0.98232	–	–	–	–	–	3.54534
–	0.975	–	–	–	–	–	3.38105
–	0.95	–	–	–	–	–	3.37619
–	0.92929	–	–	–	–	–	3.38756
–	–	0.95125	–	–	–	–	3.37308
–	–	0.93106	–	–	–	–	3.37198
–	–	0.86211	–	–	–	–	3.38260
–	–	0.805	–	–	–	–	3.43956
–	–	–	0.15	–	–	–	3.38125
–	–	–	0.21213	–	–	–	3.37582
–	–	–	0.42426	–	–	–	3.37405
–	–	–	0.6	–	–	–	3.37512
–	–	–	–	0.9	–	–	3.38120
–	–	–	–	0.85858	–	–	3.37483
–	–	–	–	0.71716	–	–	3.37220
–	–	–	–	0.6	–	–	3.37908
–	–	–	–	–	0.9842	–	3.37242
–	–	–	–	–	0.98069	–	3.37241
–	–	–	–	–	0.97932	–	3.37340
–	–	–	–	–	0.97806	–	3.37127
–	–	–	–	–	0.97562	–	3.37203
–	–	–	–	–	0.97075	–	3.37093
–	–	–	–	–	0.96898	–	3.37210
–	–	–	–	–	0.96691	–	3.37094
–	–	–	–	–	0.96588	–	3.37241
–	–	–	–	–	0.96553	–	3.37175

AdamH, $B = 1M$

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	–	–	0.96344	–	3.37270
–	–	–	–	–	0.961	–	3.37246
–	–	–	–	–	0.95613	–	3.37196
–	–	–	–	–	0.9545	–	3.37201
–	–	–	–	–	0.95417	–	3.37164
–	–	–	–	–	0.95369	–	3.37186
–	–	–	–	–	0.95125	–	3.37211
–	–	–	–	–	0.95036	–	3.37168
–	–	–	–	–	0.94638	–	3.37235
–	–	–	–	–	0.9415	–	3.37144
–	–	–	–	–	0.93906	–	3.37127
–	–	–	–	–	0.93106	–	3.37125
–	–	–	–	–	0.922	–	3.37241
–	–	–	–	–	0.91727	–	3.37517
–	–	–	–	–	0.9025	–	3.37397
–	–	–	–	–	0.883	–	3.37502
–	–	–	–	–	–	0	3.41948
–	–	–	–	–	–	0.2	3.37244
–	–	–	–	–	–	0.4	3.37297
–	–	–	–	–	–	0.6	3.37766
–	–	–	–	–	–	0.8	3.38208

Table 30: Hyperparameter ablation for AdamH at batch size $B = 2M$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.025456	0.96464	0.90725	0.42426	0.8	0.9213	0.25	3.48723
0.012728	–	–	–	–	–	–	3.52999
0.018	–	–	–	–	–	–	3.49427
0.036	–	–	–	–	–	–	3.50275
0.050912	–	–	–	–	–	–	3.54708
–	0.975	–	–	–	–	–	3.50189
–	0.95	–	–	–	–	–	3.49129
–	–	0.93442	–	–	–	–	3.48752
–	–	0.86884	–	–	–	–	3.50311
–	–	0.73767	–	–	–	–	3.57150
–	–	0.62901	–	–	–	–	3.63153
–	–	–	0.3	–	–	–	3.48943
–	–	–	0.6	–	–	–	3.49050
–	–	–	–	0.9	–	–	3.48878
–	–	–	–	0.85858	–	–	3.48518
–	–	–	–	0.71716	–	–	3.49649
–	–	–	–	–	0.98033	–	3.49149
–	–	–	–	–	0.97218	–	3.48463
–	–	–	–	–	0.94435	–	3.48546
–	–	–	–	–	0.8887	–	3.48618
–	–	–	–	–	–	0	3.50407
–	–	–	–	–	–	0.2	3.48478
–	–	–	–	–	–	0.4	3.48505

LionH

Table 31: Hyperparameter ablation for LionH at batch size $B = 128K$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.006364	0.92929	0.98726	0.3	0.71716	0.99363	0.4	3.29369
0.003182	–	–	–	–	–	–	3.33450
0.0045	–	–	–	–	–	–	3.30646
0.009	–	–	–	–	–	–	3.31381
0.012728	–	–	–	–	–	–	3.35250

LionH, $B = 128K$

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	0.96464	–	–	–	–	–	3.30461
–	0.95	–	–	–	–	–	3.29967
–	0.9	–	–	–	–	–	3.30068
–	0.85858	–	–	–	–	–	3.31608
–	–	0.99363	–	–	–	–	3.30089
–	–	0.99099	–	–	–	–	3.29744
–	–	0.98198	–	–	–	–	3.29758
–	–	0.97452	–	–	–	–	3.30083
–	–	–	0.15	–	–	–	3.30024
–	–	–	0.21213	–	–	–	3.30087
–	–	–	0.42426	–	–	–	3.29772
–	–	–	0.6	–	–	–	3.29969
–	–	–	–	0.9	–	–	3.29848
–	–	–	–	0.85858	–	–	3.29766
–	–	–	–	0.8	–	–	3.29477
–	–	–	–	0.6	–	–	3.29743
–	–	–	–	0.43431	–	–	3.29948
–	–	–	–	–	0.99775	–	3.30025
–	–	–	–	–	0.99681	–	3.29933
–	–	–	–	–	0.99618	–	3.29991
–	–	–	–	–	0.9955	–	3.29609
–	–	–	–	–	0.9949	–	3.29601
–	–	–	–	–	0.99427	–	3.29557
–	–	–	–	–	0.99395	–	3.29657
–	–	–	–	–	0.99331	–	3.29658
–	–	–	–	–	0.99299	–	3.29635
–	–	–	–	–	0.99267	–	3.29658
–	–	–	–	–	0.99236	–	3.29669
–	–	–	–	–	0.99099	–	3.29483
–	–	–	–	–	–	0	3.32282
–	–	–	–	–	–	0.2	3.29707
–	–	–	–	–	–	0.6	3.29601
–	–	–	–	–	–	0.8	3.29884

Table 32: Hyperparameter ablation for LionH at batch size $B = 512K$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.018	0.92929	0.95	0.42426	0.8	0.98333	0.4	3.33575
0.009	–	–	–	–	–	–	3.37584
0.012728	–	–	–	–	–	–	3.34782
0.025456	–	–	–	–	–	–	3.34516
0.036	–	–	–	–	–	–	3.38208
–	0.96464	–	–	–	–	–	3.37497
–	0.95	–	–	–	–	–	3.33810
–	0.9	–	–	–	–	–	3.34398
–	0.85858	–	–	–	–	–	3.35917
–	0.8	–	–	–	–	–	3.38589
–	–	0.975	–	–	–	–	3.34189
–	–	0.96464	–	–	–	–	3.33838
–	–	0.92929	–	–	–	–	3.34374
–	–	0.9	–	–	–	–	3.38908
–	–	–	0.21213	–	–	–	3.34113
–	–	–	0.3	–	–	–	3.33591
–	–	–	0.50912	–	–	–	3.33594

LionH, $B = 512K$

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	0.6	–	–	–	3.33932
–	–	–	0.84853	–	–	–	3.33961
–	–	–	–	0.9	–	–	3.33676
–	–	–	–	0.85858	–	–	3.33719
–	–	–	–	0.71716	–	–	3.33825
–	–	–	–	0.6	–	–	3.34386
–	–	–	–	–	0.99167	–	3.33778
–	–	–	–	–	0.98821	–	3.33654
–	–	–	–	–	0.9875	–	3.33544
–	–	–	–	–	0.98667	–	3.33810
–	–	–	–	–	0.985	–	3.33570
–	–	–	–	–	0.98232	–	3.33654
–	–	–	–	–	0.98	–	3.33765
–	–	–	–	–	0.979	–	3.33757
–	–	–	–	–	0.97643	–	3.33675
–	–	–	–	–	0.975	–	3.33611
–	–	–	–	–	0.96667	–	3.33734
–	–	–	–	–	0.95	–	3.33678
–	–	–	–	–	0.92929	–	3.33806
–	–	–	–	–	–	0	3.36221
–	–	–	–	–	–	0.2	3.33746
–	–	–	–	–	–	0.6	3.33777
–	–	–	–	–	–	0.8	3.34169

Table 33: Hyperparameter ablation for LionH at batch size $B = 1M$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.036	0.9	0.9025	0.6	0.9	0.97562	0.4	3.40475
0.018	–	–	–	–	–	–	3.43203
0.025456	–	–	–	–	–	–	3.40845
0.050912	–	–	–	–	–	–	3.43559
0.072	–	–	–	–	–	–	3.48983
–	0.95	–	–	–	–	–	3.66610
–	0.92929	–	–	–	–	–	3.49354
–	0.85858	–	–	–	–	–	3.43192
–	0.8	–	–	–	–	–	3.45733
–	–	0.95125	–	–	–	–	3.43273
–	–	0.93106	–	–	–	–	3.41619
–	–	0.86211	–	–	–	–	3.48391
–	–	0.805	–	–	–	–	3.63295
–	–	–	0.3	–	–	–	3.41612
–	–	–	0.42426	–	–	–	3.41057
–	–	–	0.84853	–	–	–	3.41163
–	–	–	1.2	–	–	–	3.41552
–	–	–	1.6971	–	–	–	3.42273
–	–	–	–	0.95	–	–	3.40775
–	–	–	–	0.92929	–	–	3.40667
–	–	–	–	0.85858	–	–	3.40907
–	–	–	–	0.8	–	–	3.41053
–	–	–	–	–	0.98781	–	3.40605
–	–	–	–	–	0.98538	–	3.40507
–	–	–	–	–	0.98276	–	3.40538
–	–	–	–	–	0.9805	–	3.40559
–	–	–	–	–	0.96553	–	3.40448
–	–	–	–	–	0.95125	–	3.40362

LionH, $B = 1M$

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	–	–	0.93106	–	3.40504
–	–	–	–	–	0.9025	–	3.40683
–	–	–	–	–	0.86211	–	3.40711
–	–	–	–	–	0.805	–	3.40759
–	–	–	–	–	–	0	3.42772
–	–	–	–	–	–	0.2	3.40325
–	–	–	–	–	–	0.6	3.40817

Table 34: Hyperparameter ablation for LionH at batch size $B = 2M$.

η_{M}	β_1^{M}	β_2^{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.036	0.9	0.90725	0.6	0.9	0.95363	0.2	3.55651
0.018	–	–	–	–	–	–	3.60350
0.025456	–	–	–	–	–	–	3.56984
0.050912	–	–	–	–	–	–	3.57392
0.072	–	–	–	–	–	–	3.62350
–	0.92929	–	–	–	–	–	3.64818
–	0.85858	–	–	–	–	–	3.61206
–	–	0.93442	–	–	–	–	3.59923
–	–	0.86884	–	–	–	–	3.64545
–	–	–	0.42426	–	–	–	3.56105
–	–	–	0.84853	–	–	–	3.56483
–	–	–	–	0.92929	–	–	3.55956
–	–	–	–	0.85858	–	–	3.56042
–	–	–	–	–	0.97681	–	3.55751
–	–	–	–	–	0.96721	–	3.55283
–	–	–	–	–	0.93442	–	3.55904
–	–	–	–	–	–	0	3.57606
–	–	–	–	–	–	0.4	3.55957

F.7.2 WEIGHT DECAY SWEEPS

MuonW

Table 35: Hyperparameter ablation for MuonW at batch size $B = 128K$.

η_{M}	μ	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.017678	0.95	0.025	0.49497	0.8	0.99363	0.8	3.26547
0.00625	–	–	–	–	–	–	3.27885
0.0088388	–	–	–	–	–	–	3.27196
0.0125	–	–	–	–	–	–	3.26909
0.025	–	–	–	–	–	–	3.27012
0.035355	–	–	–	–	–	–	3.28074
–	0.975	–	–	–	–	–	3.26698
–	0.96464	–	–	–	–	–	3.26617
–	0.92929	–	–	–	–	–	3.26707
–	0.9	–	–	–	–	–	3.27014
–	–	0.01	–	–	–	–	3.26771
–	–	0.05	–	–	–	–	3.27446
–	–	–	0.35	–	–	–	3.26966
–	–	–	0.7	–	–	–	3.26812
–	–	–	0.98995	–	–	–	3.27181
–	–	–	–	0.9	–	–	3.26904

MuonW, $B = 128K$

η_{M}	μ	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	–	0.85858	–	–	3.26685
–	–	–	–	0.71716	–	–	3.26623
–	–	–	–	0.6	–	–	3.26888
–	–	–	–	–	0.99681	–	3.26637
–	–	–	–	–	0.9955	–	3.26512
–	–	–	–	–	0.99099	–	3.26567
–	–	–	–	–	0.98726	–	3.26673
–	–	–	–	–	–	0.6	3.26940
–	–	–	–	–	–	1	3.26561

Table 36: Hyperparameter ablation for MuonW at batch size $B = 512K$.

η_{M}	μ	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.025	0.95	0.05	0.7	0.8	0.96464	0.7	3.27788
0.0125	–	–	–	–	–	–	3.28710
0.017678	–	–	–	–	–	–	3.28196
0.035355	–	–	–	–	–	–	3.28059
0.05	–	–	–	–	–	–	3.28835
–	0.96464	–	–	–	–	–	3.27811
–	0.92929	–	–	–	–	–	3.27926
–	–	0.025	–	–	–	–	3.28061
–	–	0.1	–	–	–	–	3.28498
–	–	–	0.49497	–	–	–	3.27872
–	–	–	0.98995	–	–	–	3.27890
–	–	–	–	0.9	–	–	3.27871
–	–	–	–	0.85858	–	–	3.27724
–	–	–	–	0.71716	–	–	3.27936
–	–	–	–	–	0.98232	–	3.27671
–	–	–	–	–	0.975	–	3.27618
–	–	–	–	–	0.95	–	3.27863
–	–	–	–	–	0.92929	–	3.28128
–	–	–	–	–	0.9	–	3.28350
–	–	–	–	–	–	0.6	3.27873
–	–	–	–	–	–	0.8	3.27792

Table 37: Hyperparameter ablation for MuonW at batch size $B = 1M$.

η_{M}	μ	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.025	0.95	0.1	0.98995	0.8	0.93106	0.8	3.31108
0.0125	–	–	–	–	–	–	3.32240
0.017678	–	–	–	–	–	–	3.31407
0.035355	–	–	–	–	–	–	3.31274
0.05	–	–	–	–	–	–	3.32043
0.070711	–	–	–	–	–	–	3.33872
–	0.96464	–	–	–	–	–	3.31200
–	0.92929	–	–	–	–	–	3.31255
–	–	0.05	–	–	–	–	3.31761
–	–	0.15	–	–	–	–	3.31150
–	–	–	0.49497	–	–	–	3.30991
–	–	–	0.7	–	–	–	3.30942
–	–	–	1.4	–	–	–	3.31786
–	–	–	–	0.9	–	–	3.31144
–	–	–	–	0.85858	–	–	3.30962
–	–	–	–	0.71716	–	–	3.31435

MuonW, $B = 1M$

η_{M}	μ	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	–	–	0.97562	–	3.31019
–	–	–	–	–	0.96553	–	3.30994
–	–	–	–	–	0.95125	–	3.31033
–	–	–	–	–	0.9025	–	3.31181
–	–	–	–	–	–	0.4	3.31609
–	–	–	–	–	–	0.6	3.30963
–	–	–	–	–	–	1	3.31617

Table 38: Hyperparameter ablation for MuonW at batch size $B = 2M$.

η_{M}	μ	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.025	0.95	0.1	0.7	0.85858	0.90725	0.4	3.36617
0.0125	–	–	–	–	–	–	3.37566
0.017678	–	–	–	–	–	–	3.36828
0.035355	–	–	–	–	–	–	3.37356
0.05	–	–	–	–	–	–	3.38618
–	0.975	–	–	–	–	–	3.37805
–	0.96464	–	–	–	–	–	3.36938
–	0.92929	–	–	–	–	–	3.36714
–	0.9	–	–	–	–	–	3.37494
–	–	0.05	–	–	–	–	3.36880
–	–	0.070711	–	–	–	–	3.36811
–	–	0.075	–	–	–	–	3.36815
–	–	0.14142	–	–	–	–	3.36836
–	–	0.15	–	–	–	–	3.36782
–	–	0.2	–	–	–	–	3.37323
–	–	–	0.35	–	–	–	3.37690
–	–	–	0.49497	–	–	–	3.36928
–	–	–	0.98995	–	–	–	3.37011
–	–	–	1.4	–	–	–	3.38212
–	–	–	–	0.92929	–	–	3.37232
–	–	–	–	0.9	–	–	3.36761
–	–	–	–	0.8	–	–	3.36755
–	–	–	–	0.71716	–	–	3.37453
–	–	–	–	–	0.95363	–	3.36652
–	–	–	–	–	0.93442	–	3.36564
–	–	–	–	–	0.86884	–	3.36584
–	–	–	–	–	0.81451	–	3.36714
–	–	–	–	–	–	0	3.50834
–	–	–	–	–	–	0.2	3.38019
–	–	–	–	–	–	0.6	3.36769
–	–	–	–	–	–	0.8	3.37831

KL-SOAP-W

Table 39: Hyperparameter ablation for KL-SOAP-W at batch size $B = 128K$.

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.0045	0.96464	0.974	0.94801	0.1	0.15	0.43431	0.9955	0.4	3.25388
0.00225	–	–	–	–	–	–	–	–	3.26253
0.003182	–	–	–	–	–	–	–	–	3.25685
0.006364	–	–	–	–	–	–	–	–	3.25702
0.009	–	–	–	–	–	–	–	–	3.26629

KL-SOAP-W, $B = 128K$

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	0.975	–	–	–	–	–	–	–	3.25474
–	0.95	–	–	–	–	–	–	–	3.25607
–	0.92929	–	–	–	–	–	–	–	3.25725
–	–	0.98162	–	–	–	–	–	–	3.25606
–	–	0.96324	–	–	–	–	–	–	3.25523
–	–	0.94801	–	–	–	–	–	–	3.25426
–	–	–	0.974	–	–	–	–	–	3.25568
–	–	–	0.96324	–	–	–	–	–	3.25599
–	–	–	0.92647	–	–	–	–	–	3.25708
–	–	–	0.89601	–	–	–	–	–	3.25503
–	–	–	–	0	–	–	–	–	3.27457
–	–	–	–	0.05	–	–	–	–	3.25839
–	–	–	–	0.15	–	–	–	–	3.25699
–	–	–	–	0.2	–	–	–	–	3.26210
–	–	–	–	–	0.075	–	–	–	3.25869
–	–	–	–	–	0.10607	–	–	–	3.25464
–	–	–	–	–	0.21213	–	–	–	3.25268
–	–	–	–	–	0.42426	–	–	–	3.25694
–	–	–	–	–	0.6	–	–	–	3.25690
–	–	–	–	–	–	0.71716	–	–	3.25486
–	–	–	–	–	–	0.6	–	–	3.25562
–	–	–	–	–	–	0.2	–	–	3.25365
–	–	–	–	–	–	0	–	–	3.25409
–	–	–	–	–	–	–	0.99775	–	3.25740
–	–	–	–	–	–	–	0.99681	–	3.25816
–	–	–	–	–	–	–	0.99363	–	3.25516
–	–	–	–	–	–	–	0.99099	–	3.25588
–	–	–	–	–	–	–	–	0	3.28109
–	–	–	–	–	–	–	–	0.2	3.25560
–	–	–	–	–	–	–	–	0.6	3.25541
–	–	–	–	–	–	–	–	0.8	3.25522

Table 40: Hyperparameter ablation for KL-SOAP-W at batch size $B = 512K$.

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.012728	0.9	0.85858	0.85858	0.15	0.84853	0.8	0.975	0.4	3.27594
0.006364	–	–	–	–	–	–	–	–	3.28188
0.009	–	–	–	–	–	–	–	–	3.27597
0.018	–	–	–	–	–	–	–	–	3.28161
0.025456	–	–	–	–	–	–	–	–	3.28798
–	0.96464	–	–	–	–	–	–	–	3.96353
–	0.95	–	–	–	–	–	–	–	3.29349
–	0.92929	–	–	–	–	–	–	–	3.27927
–	0.85858	–	–	–	–	–	–	–	3.28136
–	–	0.9	–	–	–	–	–	–	3.27871
–	–	0.8	–	–	–	–	–	–	3.27551
–	–	0.71716	–	–	–	–	–	–	3.27970
–	–	–	0.9	–	–	–	–	–	3.27673
–	–	–	0.8	–	–	–	–	–	3.27583
–	–	–	0.71716	–	–	–	–	–	3.27840
–	–	–	–	0.1	–	–	–	–	3.28024
–	–	–	–	0.2	–	–	–	–	3.27657
–	–	–	–	–	0.42426	–	–	–	3.27784
–	–	–	–	–	0.6	–	–	–	3.27569

KL-SOAP-W, $B = 512K$

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	–	–	1.2	–	–	–	3.27908
–	–	–	–	–	–	0.9	–	–	3.27876
–	–	–	–	–	–	0.85858	–	–	3.27572
–	–	–	–	–	–	0.71716	–	–	3.27833
–	–	–	–	–	–	–	0.9875	–	3.27830
–	–	–	–	–	–	–	0.98232	–	3.27428
–	–	–	–	–	–	–	0.96464	–	3.27787
–	–	–	–	–	–	–	–	0.2	3.27776
–	–	–	–	–	–	–	–	0.6	3.27423
–	–	–	–	–	–	–	–	0.8	3.27605

Table 41: Hyperparameter ablation for KL-SOAP-W at batch size $B = 1M$.

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.006364	0.92929	0.86565	0.86565	0.6	0.6	0.8	0.95125	0.6	3.29629
0.003182	–	–	–	–	–	–	–	–	3.30629
0.0045	–	–	–	–	–	–	–	–	3.30860
0.009	–	–	–	–	–	–	–	–	3.30143
0.012728	–	–	–	–	–	–	–	–	3.30874
–	0.96464	–	–	–	–	–	–	–	3.32143
–	0.95	–	–	–	–	–	–	–	3.31439
–	0.9	–	–	–	–	–	–	–	3.29697
–	–	0.93282	–	–	–	–	–	–	3.30751
–	–	0.905	–	–	–	–	–	–	3.30046
–	–	0.81	–	–	–	–	–	–	3.29852
–	–	0.7313	–	–	–	–	–	–	3.30636
–	–	–	0.905	–	–	–	–	–	3.29693
–	–	–	0.81	–	–	–	–	–	3.29712
–	–	–	0.7313	–	–	–	–	–	3.30174
–	–	–	–	0.3	–	–	–	–	3.31047
–	–	–	–	0.5	–	–	–	–	3.29937
–	–	–	–	0.8	–	–	–	–	3.29648
–	–	–	–	–	0.3	–	–	–	3.30234
–	–	–	–	–	0.42426	–	–	–	3.31750
–	–	–	–	–	0.84853	–	–	–	3.29991
–	–	–	–	–	1.2	–	–	–	3.30911
–	–	–	–	–	–	0.9	–	–	3.29952
–	–	–	–	–	–	0.85858	–	–	3.29690
–	–	–	–	–	–	0.71716	–	–	3.29863
–	–	–	–	–	–	–	0.97562	–	3.29949
–	–	–	–	–	–	–	0.96553	–	3.29591
–	–	–	–	–	–	–	0.93106	–	3.29848
–	–	–	–	–	–	–	0.9025	–	3.30074
–	–	–	–	–	–	–	–	0	3.32420
–	–	–	–	–	–	–	–	0.2	3.29897
–	–	–	–	–	–	–	–	0.4	3.29730
–	–	–	–	–	–	–	–	0.8	3.29812

Table 42: Hyperparameter ablation for KL-SOAP-W at batch size $B = 2M$.

η_{M}	β_1^{M}	β_2^{M}	β_{sh}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.036	0.92929	0.82805	0.87841	0.4	0.6	0.8	0.93442	0.4	3.38357
0.018	–	–	–	–	–	–	–	–	3.39916
0.025456	–	–	–	–	–	–	–	–	3.41082
0.050912	–	–	–	–	–	–	–	–	3.41118
0.072	–	–	–	–	–	–	–	–	3.43941
–	0.95	–	–	–	–	–	–	–	3.41115
–	0.9	–	–	–	–	–	–	–	3.40327
–	–	0.87841	–	–	–	–	–	–	3.38760
–	–	0.75683	–	–	–	–	–	–	3.39691
–	–	–	0.95701	–	–	–	–	–	3.39885
–	–	–	0.93921	–	–	–	–	–	3.38206
–	–	–	0.91402	–	–	–	–	–	3.39036
–	–	–	0.82805	–	–	–	–	–	3.39183
–	–	–	–	0.3	–	–	–	–	3.39107
–	–	–	–	0.5	–	–	–	–	3.38952
–	–	–	–	–	0.42426	–	–	–	3.38793
–	–	–	–	–	0.84853	–	–	–	3.42713
–	–	–	–	–	–	0.9	–	–	3.38595
–	–	–	–	–	–	0.85858	–	–	3.37747
–	–	–	–	–	–	0.71716	–	–	3.39432
–	–	–	–	–	–	–	0.95363	–	3.38445
–	–	–	–	–	–	–	0.90725	–	3.38303
–	–	–	–	–	–	–	0.86884	–	3.39009
–	–	–	–	–	–	–	–	0.2	3.38529
–	–	–	–	–	–	–	–	0.6	3.38907

ShampooW

Table 43: Hyperparameter ablation for ShampooW at batch size $B = 128K$.

η_{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.005	0.05	1.2	0.52	0.9955	0.6	3.26453
0.0025	–	–	–	–	–	3.26990
0.0035355	–	–	–	–	–	3.26949
0.0070711	–	–	–	–	–	3.26794
0.01	–	–	–	–	–	3.27269
–	0.025	–	–	–	–	3.26934
–	0.1	–	–	–	–	3.27064
–	–	0.6	–	–	–	3.26864
–	–	0.84853	–	–	–	3.26383
–	–	1.6971	–	–	–	3.26994
–	–	–	0.66059	–	–	3.26867
–	–	–	0.32118	–	–	3.26695
–	–	–	–	0.99681	–	3.27083
–	–	–	–	0.99363	–	3.26594
–	–	–	–	–	0	3.46776
–	–	–	–	–	0.2	3.29150
–	–	–	–	–	0.4	3.27501
–	–	–	–	–	0.8	3.26451
–	–	–	–	–	1	3.26738

Table 44: Hyperparameter ablation for ShampooW at batch size $B = 512K$.

η_{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.0070711	0.15	0.84853	0.8	0.98232	0.6	3.28179
0.0035355	–	–	–	–	–	3.28292
0.005	–	–	–	–	–	3.28384
0.01	–	–	–	–	–	3.28310
0.014142	–	–	–	–	–	3.28916
–	0.075	–	–	–	–	3.28211
–	0.1	–	–	–	–	3.28383
–	0.10607	–	–	–	–	3.28269
–	0.2	–	–	–	–	3.28183
–	0.21213	–	–	–	–	3.28258
–	0.3	–	–	–	–	3.28561
–	–	0.42426	–	–	–	3.28095
–	–	0.6	–	–	–	3.27898
–	–	1.2	–	–	–	3.28437
–	–	1.6971	–	–	–	3.28486
–	–	–	0.92929	–	–	3.28336
–	–	–	0.9	–	–	3.28064
–	–	–	0.85858	–	–	3.28221
–	–	–	0.71716	–	–	3.28098
–	–	–	0.6	–	–	3.28077
–	–	–	–	0.99375	–	3.28108
–	–	–	–	0.99116	–	3.27941
–	–	–	–	0.9875	–	3.28162
–	–	–	–	0.975	–	3.28231
–	–	–	–	0.96464	–	3.28236
–	–	–	–	–	0	3.47668
–	–	–	–	–	0.2	3.30494
–	–	–	–	–	0.4	3.28826
–	–	–	–	–	0.8	3.28314

Table 45: Hyperparameter ablation for ShampooW at batch size $B = 1M$.

η_{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.0070711	0.3	0.84853	0.8	0.97075	0.6	3.30873
0.0035355	–	–	–	–	–	3.31717
0.005	–	–	–	–	–	3.31121
0.01	–	–	–	–	–	3.31060
0.014142	–	–	–	–	–	3.31547
–	0.2	–	–	–	–	3.31070
–	0.4	–	–	–	–	3.30676
–	0.5	–	–	–	–	3.31051
–	–	0.42426	–	–	–	3.30973
–	–	0.6	–	–	–	3.30635
–	–	1.2	–	–	–	3.31106
–	–	–	0.85858	–	–	3.30935
–	–	–	0.71716	–	–	3.31020
–	–	–	–	0.97932	–	3.30906
–	–	–	–	0.95863	–	3.30866
–	–	–	–	0.9415	–	3.30865
–	–	–	–	0.91727	–	3.30964
–	–	–	–	–	0	3.49861
–	–	–	–	–	0.2	3.33490
–	–	–	–	–	0.4	3.31392
–	–	–	–	–	0.8	3.31119

Table 46: Hyperparameter ablation for ShampooW at batch size $B = 2M$.

η_{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.01	0.5	0.6	0.8	0.94435	0.6	3.35964
0.005	–	–	–	–	–	3.36559
0.0070711	–	–	–	–	–	3.36361
0.014142	–	–	–	–	–	3.37268
0.02	–	–	–	–	–	3.38900
–	0.4	–	–	–	–	3.36664
–	0.6	–	–	–	–	3.36263
–	–	0.42426	–	–	–	3.36246
–	–	0.84853	–	–	–	3.36678
–	–	–	0.85858	–	–	3.36125
–	–	–	0.71716	–	–	3.36096
–	–	–	–	0.96065	–	3.36368
–	–	–	–	0.9213	–	3.36152
–	–	–	–	–	0	3.57333
–	–	–	–	–	0.2	3.39395
–	–	–	–	–	0.4	3.36951
–	–	–	–	–	0.8	3.36683

AdamW

Table 47: Hyperparameter ablation for AdamW at batch size $B = 128K$.

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.00075	0.94	0.94903	0.3	0.84853	0.71716	0.9955	0.4	3.31446
0.000375	–	–	–	–	–	–	–	3.33480
0.00053033	–	–	–	–	–	–	–	3.32473
0.0010607	–	–	–	–	–	–	–	3.31801
0.0015	–	–	–	–	–	–	–	3.32305
–	0.97	–	–	–	–	–	–	3.32536
–	0.95757	–	–	–	–	–	–	3.31820
–	0.91515	–	–	–	–	–	–	3.31543
–	0.88	–	–	–	–	–	–	3.31679
–	–	0.97452	–	–	–	–	–	3.31679
–	–	0.96396	–	–	–	–	–	3.31381
–	–	0.92792	–	–	–	–	–	3.31817
–	–	0.89807	–	–	–	–	–	3.31638
–	–	–	0.15	–	–	–	–	3.31748
–	–	–	0.2	–	–	–	–	3.31721
–	–	–	0.21213	–	–	–	–	3.31633
–	–	–	0.4	–	–	–	–	3.31524
–	–	–	0.42426	–	–	–	–	3.31598
–	–	–	–	0.21213	–	–	–	3.32787
–	–	–	–	0.3	–	–	–	3.32338
–	–	–	–	0.42426	–	–	–	3.32150
–	–	–	–	0.6	–	–	–	3.31635
–	–	–	–	1.2	–	–	–	3.31952
–	–	–	–	1.6971	–	–	–	3.31667
–	–	–	–	–	0.85858	–	–	3.32043
–	–	–	–	–	0.8	–	–	3.31978
–	–	–	–	–	0.6	–	–	3.32092
–	–	–	–	–	0.43432	–	–	3.31469
–	–	–	–	–	–	0.99775	–	3.32381
–	–	–	–	–	–	0.99681	–	3.32411
–	–	–	–	–	–	0.99363	–	3.31930
–	–	–	–	–	–	0.99099	–	3.31854
–	–	–	–	–	–	–	0	3.48153
–	–	–	–	–	–	–	0.2	3.32944
–	–	–	–	–	–	–	0.6	3.31252
–	–	–	–	–	–	–	0.8	3.31610

Table 48: Hyperparameter ablation for AdamW at batch size $B = 512K$.

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.0015	0.9	0.85858	0.6	0.84853	0.85858	0.975	0.4	3.33876
0.00075	–	–	–	–	–	–	–	3.36489
0.0010607	–	–	–	–	–	–	–	3.34867
0.0021213	–	–	–	–	–	–	–	3.35025
0.003	–	–	–	–	–	–	–	3.39202
–	0.92929	–	–	–	–	–	–	3.41100
–	0.85858	–	–	–	–	–	–	3.34886
–	–	0.9	–	–	–	–	–	3.34161
–	–	0.8	–	–	–	–	–	3.36918
–	–	–	0.4	–	–	–	–	3.34101
–	–	–	0.5	–	–	–	–	3.33740
–	–	–	0.8	–	–	–	–	3.34298
–	–	–	–	0.6	–	–	–	3.34492
–	–	–	–	1.2	–	–	–	3.34030
–	–	–	–	–	0.9	–	–	3.34408
–	–	–	–	–	0.8	–	–	3.34039
–	–	–	–	–	–	0.9875	–	3.33920
–	–	–	–	–	–	0.98232	–	3.33875
–	–	–	–	–	–	0.96464	–	3.34558
–	–	–	–	–	–	–	0.2	3.35213
–	–	–	–	–	–	–	0.6	3.33901

Table 49: Hyperparameter ablation for AdamW at batch size $B = 1M$.

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.0015	0.85858	0.93106	0.8	0.6	0.85858	0.9025	0.4	3.40136
0.00075	–	–	–	–	–	–	–	3.44254
0.0010607	–	–	–	–	–	–	–	3.41807
0.0021213	–	–	–	–	–	–	–	3.41238
0.003	–	–	–	–	–	–	–	3.50500
–	0.9	–	–	–	–	–	–	3.41518
–	0.8	–	–	–	–	–	–	3.43137
–	–	0.95125	–	–	–	–	–	3.40474
–	–	0.9025	–	–	–	–	–	3.40842
–	–	–	0.5	–	–	–	–	3.40968
–	–	–	0.6	–	–	–	–	3.40047
–	–	–	1	–	–	–	–	3.40526
–	–	–	–	0.21213	–	–	–	3.43876
–	–	–	–	0.3	–	–	–	3.41884
–	–	–	–	0.42426	–	–	–	3.40718
–	–	–	–	0.84853	–	–	–	3.40356
–	–	–	–	–	0.9	–	–	3.40841
–	–	–	–	–	0.8	–	–	3.41039
–	–	–	–	–	–	0.93106	–	3.40439
–	–	–	–	–	–	0.86211	–	3.41144
–	–	–	–	–	–	–	0.2	3.41017
–	–	–	–	–	–	–	0.6	3.40936

Table 50: Hyperparameter ablation for AdamW at batch size $B = 2M$.

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.0021213	0.9	0.86884	1.8	3.3941	0.92929	0.96721	0.6	3.47554
0.0010607	–	–	–	–	–	–	–	3.52875
0.0015	–	–	–	–	–	–	–	3.49515
0.003	–	–	–	–	–	–	–	3.49297
0.0042426	–	–	–	–	–	–	–	3.52116
–	0.95	–	–	–	–	–	–	3.69887
–	0.92929	–	–	–	–	–	–	3.52118
–	0.85858	–	–	–	–	–	–	3.49168
–	0.8	–	–	–	–	–	–	3.51213
–	–	0.93442	–	–	–	–	–	3.48525
–	–	0.90725	–	–	–	–	–	3.47930
–	–	0.81451	–	–	–	–	–	3.50016
–	–	0.73767	–	–	–	–	–	3.72114
–	–	–	0.9	–	–	–	–	3.48466
–	–	–	1.6	–	–	–	–	3.47878

AdamW, $B = 2M$

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	1.7	–	–	–	–	3.47979
–	–	–	2	–	–	–	–	3.47858
–	–	–	2.2	–	–	–	–	3.47810
–	–	–	2.5456	–	–	–	–	3.48380
–	–	–	3.6	–	–	–	–	3.49084
–	–	–	–	1.6971	–	–	–	3.48095
–	–	–	–	2.4	–	–	–	3.48309
–	–	–	–	4.8	–	–	–	3.48230
–	–	–	–	6.7882	–	–	–	3.49032
–	–	–	–	–	0.96464	–	–	3.50568
–	–	–	–	–	0.95	–	–	3.48589
–	–	–	–	–	0.9	–	–	3.48366
–	–	–	–	–	0.85858	–	–	3.48957
–	–	–	–	–	–	0.9836	–	3.48005
–	–	–	–	–	–	0.97681	–	3.47830
–	–	–	–	–	–	0.95363	–	3.48116
–	–	–	–	–	–	0.93442	–	3.48505
–	–	–	–	–	–	–	0	3.68744
–	–	–	–	–	–	–	0.2	3.49286
–	–	–	–	–	–	–	0.4	3.47851
–	–	–	–	–	–	–	0.8	3.48741

LionW

Table 51: Hyperparameter ablation for LionW at batch size $B = 128K$.

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
5×10^{-5}	0.9	0.98996	8.6	0.28728	0.85858	0.9955	0.6	3.32190
2.5×10^{-5}	–	–	–	–	–	–	–	3.36648
3.5355×10^{-5}	–	–	–	–	–	–	–	3.33947
7.0711×10^{-5}	–	–	–	–	–	–	–	3.33920
0.0001	–	–	–	–	–	–	–	3.37269
–	0.92929	–	–	–	–	–	–	3.32372
–	0.85858	–	–	–	–	–	–	3.32665
–	–	0.9929	–	–	–	–	–	3.32357
–	–	0.9858	–	–	–	–	–	3.32438
–	–	–	5.6	–	–	–	–	3.32302
–	–	–	6.4	–	–	–	–	3.32195
–	–	–	–	0.20314	–	–	–	3.33232
–	–	–	–	0.40628	–	–	–	3.32345
–	–	–	–	–	0.92929	–	–	3.32510
–	–	–	–	–	0.9	–	–	3.32200
–	–	–	–	–	0.8	–	–	3.32525
–	–	–	–	–	–	0.99972	–	3.32575
–	–	–	–	–	–	0.9996	–	3.32653
–	–	–	–	–	–	0.99944	–	3.32676
–	–	–	–	–	–	0.9992	–	3.32544
–	–	–	–	–	–	0.99887	–	3.32472
–	–	–	–	–	–	0.99841	–	3.31860
–	–	–	–	–	–	0.99775	–	3.31984
–	–	–	–	–	–	0.99681	–	3.32084
–	–	–	–	–	–	0.99363	–	3.32486
–	–	–	–	–	–	–	0.2	3.34165
–	–	–	–	–	–	–	0.8	3.32735

Table 52: Hyperparameter ablation for LionW at batch size $B = 512K$.

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.0001	0.95	0.96	0.05	0.21213	0.71716	0.95	0.2	3.39255
5×10^{-5}	–	–	–	–	–	–	–	3.44386
7.0711×10^{-5}	–	–	–	–	–	–	–	3.41694
0.00014142	–	–	–	–	–	–	–	3.39765
0.0002	–	–	–	–	–	–	–	3.44060
–	0.975	–	–	–	–	–	–	3.41967
–	0.96464	–	–	–	–	–	–	3.39877
–	0.92929	–	–	–	–	–	–	3.39527
–	0.9	–	–	–	–	–	–	3.40335
–	–	0.97172	–	–	–	–	–	3.39925
–	–	0.94343	–	–	–	–	–	3.40607
–	–	–	0.025	–	–	–	–	3.39471
–	–	–	0.1	–	–	–	–	3.39504
–	–	–	–	0.10607	–	–	–	3.39889
–	–	–	–	0.15	–	–	–	3.39469
–	–	–	–	0.3	–	–	–	3.39468
–	–	–	–	–	0.85858	–	–	3.40081
–	–	–	–	–	0.8	–	–	3.39596
–	–	–	–	–	0.6	–	–	3.39377
–	–	–	–	–	0.43431	–	–	3.39068
–	–	–	–	–	0.2	–	–	3.40208
–	–	–	–	–	–	0.98232	–	3.39259
–	–	–	–	–	–	0.975	–	3.38962
–	–	–	–	–	–	0.96464	–	3.39202
–	–	–	–	–	–	0.92929	–	3.39782
–	–	–	–	–	–	0.9	–	3.39909
–	–	–	–	–	–	–	0	3.45243
–	–	–	–	–	–	–	0.4	3.39615
–	–	–	–	–	–	–	0.6	3.41057
–	–	–	–	–	–	–	0.8	3.43095

Table 53: Hyperparameter ablation for LionW at batch size $B = 1M$.

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.00014142	0.95	0.94371	0.15	0.3	0.6	0.9025	0.2	3.44463
7.0711×10^{-5}	–	–	–	–	–	–	–	3.50421
10×10^{-5}	–	–	–	–	–	–	–	3.47023
0.0002	–	–	–	–	–	–	–	3.48006
0.00028284	–	–	–	–	–	–	–	3.55216
–	0.96464	–	–	–	–	–	–	3.48061
–	0.92929	–	–	–	–	–	–	3.45325
–	0.9	–	–	–	–	–	–	3.47114
–	–	0.97186	–	–	–	–	–	3.48901
–	–	0.9602	–	–	–	–	–	3.47619
–	–	0.9204	–	–	–	–	–	3.48948
–	–	0.88743	–	–	–	–	–	3.54382
–	–	–	0.1	–	–	–	–	3.45037
–	–	–	0.2	–	–	–	–	3.44365
–	–	–	0.3	–	–	–	–	3.44293
–	–	–	0.4	–	–	–	–	3.44494
–	–	–	–	0.15	–	–	–	3.45337
–	–	–	–	0.21213	–	–	–	3.46349
–	–	–	–	0.42426	–	–	–	3.45615

LionW, $B = 1M$

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
–	–	–	–	–	0.85858	–	–	3.46795
–	–	–	–	–	0.8	–	–	3.44993
–	–	–	–	–	0.71716	–	–	3.45528
–	–	–	–	–	0.43431	–	–	3.45558
–	–	–	–	–	0.2	–	–	3.47755
–	–	–	–	–	–	0.95125	–	3.45198
–	–	–	–	–	–	0.93106	–	3.44620
–	–	–	–	–	–	0.86211	–	3.45001
–	–	–	–	–	–	–	0	3.50569
–	–	–	–	–	–	–	0.4	3.45409
–	–	–	–	–	–	–	0.6	3.47429
–	–	–	–	–	–	–	0.8	3.49909

Table 54: Hyperparameter ablation for LionW at batch size $B = 2M$.

η_{M}	β_1^{M}	β_2^{M}	λ_{M}	$\eta_{\mathrm{Emb.}}$	β_1	β_2	f_{cd}	L_{val}
0.0002	0.9	0.88855	0.3	0.3	0.8	0.95363	0.2	3.55681
0.0001	–	–	–	–	–	–	–	3.64900
0.00014142	–	–	–	–	–	–	–	3.59357
0.00028284	–	–	–	–	–	–	–	3.55522
0.0004	–	–	–	–	–	–	–	3.63287
–	0.92929	–	–	–	–	–	–	3.64348
–	0.85858	–	–	–	–	–	–	3.57476
–	–	0.92119	–	–	–	–	–	3.55755
–	–	0.84238	–	–	–	–	–	3.65813
–	–	–	0.2	–	–	–	–	3.55596
–	–	–	0.4	–	–	–	–	3.55535
–	–	–	0.5	–	–	–	–	3.55518
–	–	–	0.6	–	–	–	–	3.55319
–	–	–	0.8	–	–	–	–	3.55844
–	–	–	–	0.21213	–	–	–	3.56433
–	–	–	–	0.42426	–	–	–	3.57157
–	–	–	–	0.6	–	–	–	3.55747
–	–	–	–	–	0.85858	–	–	3.56840
–	–	–	–	–	0.71716	–	–	3.56414
–	–	–	–	–	–	0.96721	–	3.56378
–	–	–	–	–	–	0.93442	–	3.55550
–	–	–	–	–	–	0.90725	–	3.55934
–	–	–	–	–	–	–	0	3.62200
–	–	–	–	–	–	–	0.4	3.57274

G DIRECTIONAL BRANCHING IN THE LANGUAGE MODEL

This appendix describes how the branches of Section 6 are constructed and evaluated, and lists every endpoint behind Figure 5. Section G.1 establishes a local noisy-quadratic correspondence for the polar-Jacobian choice of diagnostic geometry.

Base run and anchors. The base run is the tuned 128K MuonW–AdamW configuration of Section F, trained for 13,000 steps of 128K tokens. Its matrix learning rate is constant until step 2,601 and then decays linearly to zero at step 13,000. Anchors are placed every 1,000 steps from 1,000 to 12,000. A branch restores the full optimizer state at its anchor, sees the same data in the same order as the base run, and follows the base run's learning-rate schedule by global step. Every branch trains for 1,024 base steps (134M tokens).

Gauss–Newton matrix. Let θ denote the vector of hidden-matrix parameters, with all other parameters fixed at the anchor. For a batch of m prediction tokens, let $z_i(\theta)$ be the vocabulary logits for token i, $p_i = \mathrm{softmax}(z_i(\theta))$ its predicted probability vector, and $J_i = \partial z_i/\partial\theta$ the logit Jacobian. For the mean token cross-entropy loss, the generalized Gauss–Newton matrix is

$$G = \frac{1}{m}\sum_{i=1}^{m} J_i^\top \left(\mathrm{diag}(p_i) - p_i p_i^\top\right) J_i,$$

where $\mathrm{diag}(p_i) - p_i p_i^\top$ is the Hessian of the token cross-entropy with respect to the logits, and all quantities are evaluated at the anchor. This positive-semidefinite matrix retains the loss curvature through the logit Jacobian and omits the terms involving second derivatives of the logits with respect to the parameters. We refer to it as the Gauss–Newton matrix below.

Held subspace. Let Ψ be Muon's ideal direction map on the flattened hidden matrices, including the fixed shape factors of Section B.3. For matrix block ℓ with compact singular value decomposition $M_\ell = U_\ell \Sigma_\ell V_\ell^\top$, this map returns $a_{\mathrm{shape},\ell} U_\ell V_\ell^\top$. Let q denote the flattened momentum input to this map, including the Nesterov correction of Equation (10). For anchor parameters $\theta_\star$ and momentum buffer $m_\star$, let $q_\star$ be the reference input formed using the mean gradient $g(\theta_\star)$:

$$q_\star = \omega\mu m_\star + (1-\omega\mu) g(\theta_\star).$$

Assume that all matrix blocks of this reference input have full rank. On the full-rank domain, define the local preconditioner $P(q)$ as the derivative of the direction map with respect to this input, and freeze it at the anchor:

$$P(q) = D\Psi(q), \qquad P = P(q_\star). \tag{48}$$

Thus $P\,\delta q$ is the first-order change in the update direction caused by an input perturbation δq at the anchor. For full-rank matrix inputs, P is symmetric positive semidefinite; Section G.1 proves this property and the resulting local noisy-quadratic correspondence. Empirically, we evaluate P at the anchor's momentum buffer $m_\star$. We also damp every singular value σ of the input to $\sigma + 10^{-3}$ in the closed form of $D\Psi$; the damped operator remains symmetric positive semidefinite. At each anchor we compute the top 768 positive-eigenvalue modes of the preconditioned Gauss–Newton matrix $P^{1/2}GP^{1/2}$ restricted to the hidden matrices. The eigenvectors are computed iteratively from Gauss–Newton–vector products on 128K-token batches. An eigenvector e with positive eigenvalue h corresponds to the parameter-space direction $P^{1/2}e$, since $PG(P^{1/2}e) = hP^{1/2}e$. This mapping is the coordinate transformation of the local noisy-quadratic model, whose momentum dynamics are given in Section G.1. The rank-r construction requires at least r positive eigenvalues. We map the top r corresponding eigenvectors this way and orthonormalize the images with two Cholesky–QR passes, giving the basis $V_r \in \mathbb{R}^{N\times r}$ of the top-r held subspace, where N is the number of hidden-matrix parameters and all hidden matrices are flattened into one vector. The random control instead uses an orthonormal basis of 768 Gaussian directions, drawn once per anchor.

Branch update. Each branch forms two Muon updates from the same stream of gradients. The *base-clock* update takes one step per 128K-token base step at the scheduled learning rate. The *large-batch* update accumulates sixteen base gradients and takes one 2M step with $\sqrt{16}$ times the mean scheduled learning rate over those sixteen steps; the momentum coefficient per update is unchanged. The applied update is the base-clock update projected onto the held subspace plus the large-batch update projected onto its orthogonal complement. Matrix weight decay is applied once per base step to the current parameters, independently of the two projected updates, so the decay per token matches

the base run. The fully scaled branch applies the large-batch update everywhere, and the control branch applies the base-clock update everywhere, which recovers the unmodified optimizer. In every branch the auxiliary AdamW parameters stay on the base clock at 128K. Algorithm 6 summarizes one branch.

Algorithm 6 Directional branching

1. *Basis.* Freeze $P = D\Psi(q_\star)$, the derivative of Muon's ideal direction map at the anchor momentum input, as in Equation (48). In practice P is evaluated at $m_\star$ with singular values damped by 10^{-3}. Compute the top-r positive-eigenvalue modes $e_1, \dots, e_r$ of $P^{1/2}GP^{1/2}$ over the flattened hidden matrices and set $V = \mathrm{orth}(P^{1/2}e_1, \dots, P^{1/2}e_r)$. For the random control, V is an orthonormal basis of 768 Gaussian directions; the fully scaled branch uses $V = 0$ and the control branch $V = I$.
2. *State.* Restore the anchor's parameters and optimizer state, and make two copies of the Muon state: a base-clock copy and a large-batch copy.
3. For base steps $t = 1, \dots, 1{,}024$, with the base run's batch t, learning rate η_t, and matrix weight-decay coefficient λ_t:
   (a) Compute the 128K gradient g_t of the hidden matrices at the current parameters.
   (b) Base clock: $\Delta_{128\mathrm{K}} = \eta_t \,\mathrm{Muon}_{\mathrm{base}}(g_t)$.
   (c) Large batch: add g_t to an accumulator; when t is a multiple of 16, set $\Delta_{2\mathrm{M}} = 4\bar{\eta}\,\mathrm{Muon}_{\mathrm{large}}(\bar{g})$ with $\bar{g}$ the mean of the sixteen accumulated gradients and $\bar{\eta}$ their mean learning rate, and clear the accumulator; otherwise $\Delta_{2\mathrm{M}} = 0$.
   (d) Apply $W \leftarrow (1 - \eta_t\lambda_t)W - VV^\top\Delta_{128\mathrm{K}} - (I - VV^\top)\Delta_{2\mathrm{M}}$.
   (e) Update the auxiliary AdamW parameters on the base clock, as in the base run.
4. Evaluate the validation loss on 10.5M held-out tokens; the branch penalty is this loss minus the control branch's.

Evaluation. The branch penalty is the validation loss on 10.5M held-out tokens at the end of a branch, minus the same quantity for the control branch from the same anchor. The loss curves in Figure 5 are instead evaluated on 1M held-out tokens every 32 base steps. To estimate run-to-run noise, we ran one branch twice (anchor 3,000, 512K, top-768 held); the two penalties were $+18.9$ and $+13.5 \times 10^{-3}$ nats. Table 55 lists all endpoints.

anchor	fully scaled	top-16	top-64	top-128	top-256	top-768	random-768
1,000	+68.7	+58.9	+55.9	+53.7	+42.2	+41.6	+70.4
2,000	+50.9	–	–	–	–	+30.5	+50.8
3,000	+36.1	+34.3	+27.6	+28.6	+21.9	+18.3	+36.0
4,000	+40.3	–	–	–	–	+19.0	+39.7
5,000	+28.8	+21.2	+18.4	+17.0	+14.8	+11.7	+28.6
6,000	+25.1	–	–	–	–	+10.2	+24.6
7,000	+25.4	+20.8	+16.7	–	+14.0	+12.5	+25.5
8,000	+20.6	–	–	–	–	+11.7	+20.6
9,000	+16.6	+13.5	+12.5	–	+10.9	+10.8	+16.6
10,000	+12.6	–	–	–	–	+11.1	+12.6
11,000	+14.3	+14.3	+14.1	–	+14.1	+14.1	+14.3
12,000	+12.9	–	–	–	–	+12.9	+12.9

Table 55: **Branch penalty at 2M for every anchor and held subspace**, in units of 10^{-3} nats: the validation loss at the end of a 1,024-step branch minus that of the 128K control branch from the same anchor. A dash marks a branch that was not run.

G.1 A LOCAL NOISY-QUADRATIC MODEL FOR MUON

Setup. We use the objective L, mean gradient g, and minibatch oracle $\widehat{g}_B$ of Equation (8), and apply the optimizer recursion in Equations (9) to (11). The parameter vector θ contains the flattened hidden matrices, with auxiliary parameters fixed, and steps are indexed from the anchor by $t = 0, 1, \dots$. We allow schedules $\eta_t, \lambda_t \geq 0$ and retain $\mu \in [0, 1)$ and $\omega \in [0, 1]$. Specialize the direction map Ψ to the ideal blockwise polar map of Section B.3, including its fixed positive shape factors $a_{\mathrm{shape},\ell}$ for matrix

block ℓ. Use the local preconditioner $P(q) = D\Psi(q)$ defined in Equation (48). The results below are local to full-rank polar inputs and do not assume that the ideal polar map is smooth everywhere. All vector norms are Euclidean, matrix blocks have the equivalent Frobenius norm, and operator norms are induced norms.

Use the additive oracle in Equation (8), with independent scaled noises z_t of mean zero and fixed single-example covariance $\Sigma_0 \succeq 0$. Thus the minibatch noise is $B^{-1/2}z_t$. The initial state and frozen operators below are fixed independently of these noises. Write q_t for the Nesterov input denoted $\widetilde{m}_{t+1}$ in Equation (10). We reserve tildes below for the preconditioned coordinates of Section C.1.2. Let $(\bar{\theta}_t, \bar{m}_t)$ be the noiseless reference trajectory with the same initial state and schedules, and let $\bar{q}_t$ be its Nesterov input, so $\bar{q}_0 = q_\star$ from the subspace construction. For each of θ, m, q, use δ to denote the deviation from its reference value. In particular, $\delta\theta_0 = \delta m_0 = 0$.

Assume that all matrix blocks of the anchor input $\bar{q}_0$ have full rank. Freeze the Gauss–Newton matrix $G_0 = G$ defined above and the polar Jacobian $P_0 = P(\bar{q}_0)$. The frozen linear model has deviations $(\delta\theta_t^{\mathrm{lin}}, \delta m_t^{\mathrm{lin}})$, zero initial state, and the same noises z_t. Its linearized gradient and Nesterov-input deviations, $\delta g_t^{\mathrm{lin}}$ and $\delta q_t^{\mathrm{lin}}$, are specified by

$$\begin{aligned} \delta g_t^{\mathrm{lin}} &= G_0\delta\theta_t^{\mathrm{lin}} + B^{-1/2}z_t, \\ \delta m_{t+1}^{\mathrm{lin}} &= \mu\delta m_t^{\mathrm{lin}} + (1-\mu)\delta g_t^{\mathrm{lin}}, \\ \delta q_t^{\mathrm{lin}} &= \omega\delta m_{t+1}^{\mathrm{lin}} + (1-\omega)\delta g_t^{\mathrm{lin}}, \\ \delta\theta_{t+1}^{\mathrm{lin}} &= (1-\eta_t\lambda_t)\delta\theta_t^{\mathrm{lin}} - \eta_t P_0\delta q_t^{\mathrm{lin}}. \end{aligned} \tag{49}$$

This model approximates centered trajectory deviations.

Theorem G.1 (Local noisy-quadratic correspondence for ideal Muon)**.** *Under the setup above, P_0 is positive semidefinite. Writing $P_0^{\dagger/2}$ for the positive square root of its Moore–Penrose pseudoinverse, the coordinates*

$$\widetilde{\theta}_t = P_0^{\dagger/2}\delta\theta_t^{\mathrm{lin}}, \qquad \widetilde{m}_t = P_0^{1/2}\delta m_t^{\mathrm{lin}}$$

transform the frozen model in Equation (49) exactly into momentum/Nesterov SGD on $\operatorname{range} P_0$*, with the same schedules and decoupled weight decay, quadratic curvature $\widetilde{H}$, and minibatch noise covariance $\widetilde{\Sigma}/B$, where*

$$\widetilde{H} = P_0^{1/2}G_0P_0^{1/2}, \qquad \widetilde{\Sigma} = P_0^{1/2}\Sigma_0P_0^{1/2}.$$

Proof. We verify the geometry of the ideal update Jacobian and then transform the frozen dynamics. For a full-rank matrix block with compact singular value decomposition $M = U\Sigma V^\top$, the differential of its nuclear norm in a perturbation E is $\operatorname{tr}(U^\top EV) = \langle UV^\top, E\rangle_F$. Thus its nuclear-norm gradient is its polar factor. The ideal direction map Ψ is therefore the gradient of the convex function given by the sum of the block nuclear norms weighted by $a_{\mathrm{shape},\ell}$. This function is smooth on the full-rank neighborhood, so P_0 is symmetric positive semidefinite.

Every parameter increment in Equation (49) is a scalar shrinkage of the current deviation plus a vector in $\operatorname{range} P_0$. Zero initialization implies that $\delta\theta_t^{\mathrm{lin}}$ stays in this range and equals $P_0^{1/2}\widetilde{\theta}_t$. The momentum buffer averages unpreconditioned gradients, so its coordinate transformation multiplies by $P_0^{1/2}$. Define the transformed scaled noise by $\widetilde{z}_t = P_0^{1/2}z_t$. Multiply the linear momentum recursion by $P_0^{1/2}$ and the parameter recursion by $P_0^{\dagger/2}$. The identity $P_0^{\dagger/2}P_0 = P_0^{1/2}$ then gives

$$\begin{aligned} \widetilde{m}_{t+1} &= \mu\widetilde{m}_t + (1-\mu)(\widetilde{H}\widetilde{\theta}_t + B^{-1/2}\widetilde{z}_t), \\ \widetilde{\theta}_{t+1} &= (1-\eta_t\lambda_t)\widetilde{\theta}_t - \eta_t\big[\omega\widetilde{m}_{t+1} + (1-\omega)(\widetilde{H}\widetilde{\theta}_t + B^{-1/2}\widetilde{z}_t)\big]. \end{aligned} \tag{50}$$

These operations establish the claimed coordinate correspondence, which is invertible for parameter deviations in $\operatorname{range} P_0$. The quadratic objective is $\widetilde{\theta}^\top\widetilde{H}\widetilde{\theta}/2$. Momentum components in $\ker P_0$ do not affect the frozen parameter updates and are omitted by this reduction. If $P_0 = 0$, the parameter deviations stay zero and the reduced model has dimension zero.

A fixed linear transformation preserves the independence and zero means of z_t, and its covariance is $\operatorname{Cov}(\widetilde{z}_t) = P_0^{1/2}\Sigma_0P_0^{1/2}$. This covariance identity holds for the unconditioned frozen process. □

Remark G.2 (Curvature modes and CNR). For a unit eigenvector e_i of $\widetilde{H}$ with positive eigenvalue h_i, the noise variance is $c_i = e_i^\top \widetilde{\Sigma} e_i$ and the parameter-space direction is $P_0^{1/2} e_i$. The CNR convention of Definition 5 applies with these h_i, c_i. If the transformed noises are Gaussian and $\widetilde{H}, \widetilde{\Sigma}$ are simultaneously diagonalizable on $\operatorname{range} P_0$, their common eigenbasis gives the independent-coordinate NQM of Definition 4 on its positive-curvature modes. Without these conditions, the coordinate noises can be correlated. With $P = P_0$ at the anchor, this is the eigenvector-to-parameter construction in Algorithm 6. Large h_i alone does not imply large CNR, since c_i also enters its definition.

Approximation error. Let $H_0 = \nabla^2 L(\bar{\theta}_0)$ be the anchor Hessian. Define the Gauss–Newton discrepancy and the two reference-trajectory drifts by

$$\varepsilon_G = \|H_0 - G_0\|, \qquad d_{H,t} = \|\nabla^2 L(\bar{\theta}_t) - H_0\|, \qquad d_{P,t} = \|P(\bar{q}_t) - P_0\|.$$

Finite-batch estimation error in G_0 is included in ε_G. Assume that the Hessian is Lipschitz with constant L_H on the parameter segments $[\bar{\theta}_t, \theta_t]$. Assume that every matrix block remains full rank on the input segments $[\bar{q}_t, q_t]$, where P is Lipschitz with constant L_P. These assumptions are local conditions on a noise realization over the S steps being compared. They do not assert that unbounded noises stay in such a neighborhood. The noise distribution of the frozen linear model is not conditioned on this local event.

Let the full-state error be $\mathcal{E}_t = (\delta\theta_t - \delta\theta_t^{\mathrm{lin}}, \delta m_t - \delta m_t^{\mathrm{lin}})$, let F_t be the frozen state-transition matrix, and define the one-step error bound ϵ_t by

$$F_t = \begin{pmatrix} (1-\eta_t\lambda_t)I - \eta_t(1-\omega\mu)P_0G_0 & -\eta_t\omega\mu P_0 \\ (1-\mu)G_0 & \mu I \end{pmatrix},$$

$$\epsilon_t = \left[\eta_t(1-\omega\mu)\|P_0\| + 1 - \mu\right]\left[(\varepsilon_G + d_{H,t})\|\delta\theta_t\| + \frac{L_H}{2}\|\delta\theta_t\|^2\right] + \eta_t\left[d_{P,t}\|\delta q_t\| + \frac{L_P}{2}\|\delta q_t\|^2\right].$$

Theorem G.3 (Finite-horizon approximation error for ideal Muon)**.** *For every integer $S \geq 1$ and every noise realization satisfying the local segment assumptions for $0 \leq t < S$, the original Muon deviations and the frozen model satisfy*

$$\|\mathcal{E}_S\| \leq \sum_{t=0}^{S-1}\left(\prod_{j=t+1}^{S-1}\|F_j\|\right)\epsilon_t, \tag{51}$$

where an empty product is one.

Proof. Let the gradient deviation be $\delta g_t = \widehat{g}_B(\theta_t) - g(\bar{\theta}_t)$. Define the gradient and update remainders by

$$\begin{aligned} r_t^g &= \delta g_t - G_0\delta\theta_t - B^{-1/2}z_t, \\ r_t^u &= \Psi(q_t) - \Psi(\bar{q}_t) - P_0\delta q_t. \end{aligned} \tag{52}$$

For the error comparison, subtract the reference recursion from the noisy recursion. The definitions of the remainders give the exact identities

$$\begin{aligned} \delta g_t &= G_0\delta\theta_t + B^{-1/2}z_t + r_t^g, \\ \delta m_{t+1} &= \mu\delta m_t + (1-\mu)\delta g_t, \\ \delta q_t &= \omega\delta m_{t+1} + (1-\omega)\delta g_t, \\ \delta\theta_{t+1} &= (1-\eta_t\lambda_t)\delta\theta_t - \eta_t P_0\delta q_t - \eta_t r_t^u. \end{aligned}$$

Reference subtraction cancels the baseline update $\Psi(\bar{q}_t)$. This is essential because positive-scale invariance implies $P(q)q = 0$ for the ideal polar map, whereas $\Psi(q)$ need not vanish. The theorem therefore concerns perturbations, not the replacement of the absolute update by $P_0 q_t$.

On each realization satisfying the local segment assumptions, the nonlinear remainders satisfy

$$\begin{aligned} \|r_t^g\| &\leq (\varepsilon_G + d_{H,t})\|\delta\theta_t\| + \frac{L_H}{2}\|\delta\theta_t\|^2, \\ \|r_t^u\| &\leq d_{P,t}\|\delta q_t\| + \frac{L_P}{2}\|\delta q_t\|^2. \end{aligned} \tag{53}$$

Integrating the Hessian along the parameter segment yields

$$\left\|g(\theta_t) - g(\bar{\theta}_t) - \nabla^2 L(\bar{\theta}_t)\delta\theta_t\right\| \le \frac{L_H}{2}\|\delta\theta_t\|^2.$$

Adding and subtracting $H_0\delta\theta_t$ and $G_0\delta\theta_t$ proves the gradient bound in Equation (53). In particular, the Gauss–Newton discrepancy contributes a first-order error, not a quadratic Taylor remainder. The Lipschitz assumption on P likewise gives

$$\|\Psi(q_t) - \Psi(\bar{q}_t) - P(\bar{q}_t)\delta q_t\| \le \frac{L_P}{2}\|\delta q_t\|^2.$$

Adding and subtracting $P_0\delta q_t$ proves the update bound.

Let b_t denote the remainder forcing in the full-state error recursion. Since $\delta q_t = \omega\mu\delta m_t + (1-\omega\mu)\delta g_t$, subtracting the frozen recursion gives $\mathcal{E}_{t+1} = F_t\mathcal{E}_t + b_t$, where

$$b_t = \begin{pmatrix} -\eta_t\left[(1-\omega\mu)P_0 r_t^g + r_t^u\right] \\ (1-\mu)r_t^g \end{pmatrix}, \qquad \|b_t\| \le \epsilon_t.$$

Starting from $\mathcal{E}_0 = 0$, iterate this identity and take operator norms to obtain Equation (51). The bound makes no contraction assumption. If the transition norms admit a uniform upper bound, their product is bounded by the corresponding power of that bound. Small one-step errors alone do not imply a uniform-in-time approximation without further stability control. The comparison uses the original state coordinates because nonlinear remainders need not remain in $\mathrm{range}\, P_0$.

For the finite Newton–Schulz variant, let Ψ_{NS} denote its direction map and let its map defect be $d = \Psi_{\mathrm{NS}} - \Psi$. If both trajectories use Ψ_{NS}, retain the ideal-map Jacobian $P_0 = P(\bar{q}_0)$ along the resulting reference trajectory. Replacing Ψ by $\Psi + d$ in both trajectories adds exactly $d(q_t) - d(\bar{q}_t)$ to the update remainder. The triangle inequality adds its norm to the update-remainder bound and $\eta_t\|d(q_t) - d(\bar{q}_t)\|$ to ϵ_t. A quantitative implementation guarantee requires a bound on that defect. The quadratic model describes centered gradient responses, not the full loss difference from a nonstationary reference, which also has a linear term. Moving auxiliary parameters would introduce additional gradient-coupling terms. The local correspondence consequently motivates the diagnostic geometry without proving a branch-penalty prediction or transferring the SignSGD scaling exponents of Section 5.1 to Muon. □

H Batch Size and Optimizer Compute Efficiency

The optimizer crossovers in Section 4 show that batch size changes which optimizer achieves the lowest loss at a fixed training budget. Here we measure the compute efficiency cost of increasing batch size in a separate nanochat experiment with MuonW, KL-SOAP-W, ShampooW, AdamW, and LionW. We tune each optimizer at a reference batch of 256K tokens and transfer its hyperparameters to larger batches. We compare each result with the compute needed to reach the same loss using a 256K batch, first with the same optimizer and then with MuonW as a shared reference.

H.1 Model, data, and training budgets

This experiment uses the January 7 nanochat snapshot. The decoder has depth 8, width 512, sequence length 2,048, and vocabulary size 32,768. The scaling parameter count, used to set the training token budget, is 58,720,256. Training uses FineWeb-Edu shards 0 through 238 (Penedo et al., 2024). Shard 239 is reserved for validation. Validation bits per byte (BPB) are computed on 10,485,760 tokens. Every result uses seed 1. This setting differs from the preceding MuonW scaling-rule sweep (Section E.3) and the independently retuned optimizer comparison (Section 4). Those experiments use a 12-layer, 768-wide model on FineWeb with approximately 1.7B training tokens.

The reference global batch contains 262,144 tokens, which we abbreviate as 256K. Each optimizer is tuned at a horizon of eight training tokens per scaling parameter. This horizon contains 469,762,048 tokens and takes 1,792 optimization steps at the reference batch. We call each optimizer's selected configuration at this batch and horizon its anchor. The selected hyperparameter configurations are shown in Table 56.

We follow the notation of Sections E.1 and F. The matrix learning rate and weight decay are η_{M} and λ_{M}, and auxiliary weight decay is λ_{A}. Muon momentum is μ; the other matrix optimizers use $\beta_1^{\mathrm{M}}, \beta_2^{\mathrm{M}}$, while auxiliary Adam uses β_1, β_2. KL-SOAP's preconditioner coefficient is β_{sh}; ShampooW uses β_2^{M} for its preconditioner.

The embeddings and output projection use AdamW for every optimizer. Their learning rates $\eta_{\mathrm{Emb.}}$ and $\eta_{\mathrm{Out.}}$ have base values 0.3 and 0.004. Both rates are multiplied by the auxiliary multiplier in Table 56 and by the common width factor $(512/768)^{-1/2}$. Auxiliary AdamW uses reference coefficients $\beta_1 = 0.8$ and $\beta_2 = 0.95$, $\epsilon = 10^{-10}$, and weight decay $\lambda_{\mathrm{A}} = 0$. The cooldown fraction is $f_{\mathrm{cd}} = 0.4$; all learning rates decay linearly to zero over this final fraction of each run.

	Matrix hyperparameter configuration					
Optimizer	η_{M}	λ_{M}	μ / β_1^{M}	β_2^{M}	β_{sh}	Aux. LR mult.
MuonW	0.02	0.0473004	0.95	–	–	1.41421
KL-SOAP-W	0.0127279	0.05	0.95	0.9	0.9	1.41421
ShampooW	0.005	0.2	0.9	0.9	–	2
AdamW	0.00178381	0.6	0.9	0.95	–	1
LionW	0.000336359	12	0.9	0.99	–	1.41421

Table 56: Final anchor hyperparameter configurations at a global batch of 256K tokens and eight tokens per scaling parameter. The first-moment column uses μ for MuonW and β_1^{M} for the other optimizers. The coefficient β_{sh} applies only to KL-SOAP-W; ShampooW uses β_2^{M} for its preconditioner. The auxiliary learning rate multiplier acts on both $\eta_{\mathrm{Emb.}}$ and $\eta_{\mathrm{Out.}}$.

H.2 Anchor search

To select each optimizer's anchor, we first search a grid of matrix learning rates and matrix weight decays at 256K tokens and eight training tokens per scaling parameter. We then vary one hyperparameter at a time, extending a search range when its best observed value lies at a boundary. This single-seed search produced 178 unique completed observations with the same training implementation and data. Table 57 reports the number of observations for each optimizer and the improvement from the initial matrix grid to the selected anchor. These search-stage observations are distinct from the

observations used for the reference fits and batch comparisons in Table 61, including their 256K, eight-token-per-parameter measurements.

Optimizer	Completed points	Initial BPB	Final BPB	Improvement
MuonW	47	1.0730	1.0711	0.0019
KL-SOAP-W	23	1.0715	1.0712	0.0003
ShampooW	40	1.0777	1.0750	0.0027
AdamW	25	1.1094	1.1080	0.0014
LionW	43	1.1510	1.1369	0.0141

Table 57: Anchor search observations. Initial BPB is the selected result after the first matrix learning rate and weight decay grid. Final BPB is the selected result after the subsequent coordinate searches and boundary extensions.

H.3 Transfer across the token horizon

To estimate a reference loss–compute curve for each optimizer, we vary the training token budget while holding the global batch at 256K tokens. Let $N = 58{,}720{,}256$ denote the parameter count used for this budget and let τ denote the number of training tokens. We evaluate

$$\frac{\tau}{N} \in \{1, 2, 4, 8, 12, 16, 24\}.$$

The anchor has $\tau/N = 8$. For the relative horizon $r = (\tau/N)/8$, we multiply every peak learning rate by $r^{-0.32}$ and matrix weight decay by $r^{-0.16}$.

We obtain this scaling by combining two empirical scaling results. For peak learning rate η, Bjorck et al. (2025) fit $\eta \propto \tau^{-0.32}$ at fixed model size in their large-model experiments. For batch size B, matrix peak learning rate η_{M}, and matrix weight decay λ_{M}, the nominal parameter-averaging timescale relative to the training horizon is $\tau_{\text{Power Lines}} = B/(\eta_{\mathrm{M}}\lambda_{\mathrm{M}}\tau)$ (Equation (23)). Bergsma et al. (2025) fit this timescale as proportional to $(\tau/N)^{-0.520}$ for AdamW. Combining these two prescriptions at fixed N and B gives

$$\lambda_{\mathrm{M}} \propto \frac{1}{\eta_{\mathrm{M}}\tau\tau_{\text{Power Lines}}} \propto \tau^{0.32-1+0.520} = \tau^{-0.16}.$$

We have not established that their combination is the best scaling rule across token horizons or optimizers in our setting. The reference curves therefore measure performance under this prescribed transfer, without independent retuning at each token budget.

H.4 Scaling moving-average coefficients across batch sizes

KL-SOAP-W and ShampooW favor scaling only auxiliary AdamW β_2. For both optimizers, the best tested coefficient-transfer rule shared across 512K, 1M, and 2M scales auxiliary AdamW β_2 while keeping all other moving-average coefficients fixed. We select this common rule by its mean validation-loss gap to the best tested rule at each batch. The selected pattern matches our earlier MuonW sweep, which keeps Muon momentum and auxiliary Adam β_1 fixed and scales auxiliary Adam β_2 (Section E.3). We describe the comparison and the resulting transfer protocol below.

Batch comparison. We fix $\tau/N = 8$ and compare global batches of 256K, 512K, 1M, and 2M tokens. Let B denote the global batch in tokens, $B_0 = 262{,}144$ the reference batch, and $\kappa = B/B_0$ their ratio. The update count $S(B) = \lceil \tau/B \rceil$ is 1,792, 896, 448, and 224, respectively. For every matrix or auxiliary peak learning rate $\eta \in \{\eta_{\mathrm{M}}, \eta_{\mathrm{Emb.}}, \eta_{\mathrm{Out.}}\}$, we use

$$\eta(B) = \eta(B_0), \qquad \lambda_{\mathrm{M}}(B) = \kappa\lambda_{\mathrm{M}}(B_0).$$

Thus, peak learning rates remain fixed across batches and matrix weight decay increases linearly. Learning rates still decay to zero over the final 40 percent of each run.

These two choices follow the common rule with the smallest mean batch-size regret in the earlier MuonW sweep (Section E.3). That rule attains the grid minimum at 1M and 2M; the individual

winners at 256K and 512K instead use square-root learning-rate scaling. The fixed-learning-rate, linear-weight-decay choice also matches the corresponding Power Lines prescription (Bergsma et al., 2025). We apply these two choices to all five optimizers in the different nanochat setting described above, without independently retuning learning rates or weight decay at each batch.

Coefficient-transfer search. For a moving-average coefficient q_0 at the reference batch, let $q(B)$ denote its value at the target batch. We test two choices,

$$q(B) = q_0 \qquad \text{or} \qquad q(B) = q_0^{\kappa},$$

called fixed and EMA transfer, respectively. EMA transfer preserves the moving average's decay over the same number of processed tokens. For KL-SOAP-W, we independently choose between these transfers for β_1^{M}, β_2^{M}, β_{sh}, β_1, and β_2. Its 2^5 choices at three target batches give 96 runs. ShampooW shares β_2^{M} between its matrix second moment and preconditioner, so its 2^4 choices give 48 runs. The search therefore contains 144 runs in total.

For each optimizer, a rule's batch-size (BSZ) regret is its validation BPB minus the lowest BPB among that optimizer's tested rules at the same batch. We select one rule for all three target batches by minimizing mean BSZ regret across 512K, 1M, and 2M. Both optimizers select *auxiliary-only EMA transfer*, which applies EMA transfer only to auxiliary AdamW β_2. Its mean BSZ regret is 0.001233 BPB for KL-SOAP-W and 0.000400 BPB for ShampooW.

Transferring matrix coefficients as well gives higher loss at 2M. We compare the selected rule with an invariant-preserving rule that also applies EMA transfer to β_2^{M} and, for KL-SOAP-W, β_{sh}. Both rules keep β_1^{M} and auxiliary β_1 fixed. The invariant-preserving rule maintains the decay of second-moment and preconditioner moving averages in token units. Nevertheless, auxiliary-only EMA transfer gives lower loss for both optimizers at 2M (Table 58). Table 62 reports all 48 coefficient-transfer rules.

Optimizer	Transfer rule	512K BPB	1M BPB	2M BPB
KL-SOAP-W	**Auxiliary-only EMA**	**1.0812**	**1.1035**	**1.1614**
	Invariant-preserving	1.0867	1.1594	1.3735
ShampooW	**Auxiliary-only EMA**	**1.0847**	**1.1039**	**1.1525**
	Invariant-preserving	1.0844	1.1044	1.6558

Table 58: Coefficient transfer across batch sizes. Auxiliary-only EMA transfers auxiliary Adam β_2 and keeps all other coefficients fixed; it minimizes mean BSZ regret across the three target batches for each optimizer. The invariant-preserving rule also transfers matrix β_2^{M} and KL-SOAP-W's preconditioner coefficient β_{sh}, preserving their decay in token units. Both rules keep β_1^{M} and β_1 fixed. EMA transfer means $q(B) = q_0^{\kappa}$, where $\kappa = B/B_0$, with $B_0 = 256\mathrm{K}$ tokens.

Coefficient transfers for the remaining optimizers. For MuonW, we keep μ and auxiliary Adam β_1 fixed and apply EMA transfer to auxiliary Adam β_2. For AdamW and LionW, we keep β_1^{M} and auxiliary β_1 fixed and apply EMA transfer to both β_2^{M} and auxiliary β_2. These are prescribed choices for the remaining optimizers; the exhaustive coefficient-transfer search above covers KL-SOAP-W and ShampooW.

H.5 Compute efficiency metric

We measure training compute as the number of training tokens multiplied by 352,321,536 floating point operations per token. This common compute proxy excludes optimizer-specific update costs and does not measure wall-clock time. As in Section F, L_{val} denotes terminal validation loss, measured here in BPB. Let C denote compute and $L_{\mathrm{val},o}(C)$ the fitted validation loss for optimizer o at a batch of 256K. With a fitted asymptotic loss E_o, positive amplitude A_o, and positive exponent α_o, the reference curve is

$$L_{\mathrm{val},o}(C) = E_o + A_o \left(\frac{C}{10^{18}} \right)^{-\alpha_o}.$$

The fitted values are reported in Table 59. Each fit uses that optimizer's seven reference observations, whose losses decrease monotonically with compute.

Optimizer	Irreducible BPB	Amplitude	Exponent	RMSE
MuonW	0.983714	0.025145	0.672383	0.002624
KL-SOAP-W	0.992348	0.018555	0.805057	0.001106
ShampooW	1.011685	0.011108	0.907607	0.006164
AdamW	0.987890	0.030050	0.778845	0.005971
LionW	0.918750	0.095158	0.492865	0.013589

Table 59: Parameters of the fitted compute loss curves.

For a larger batch result with loss L_{val}, equivalent compute is the compute on the fitted reference curve that reaches the same loss. We denote this quantity by $C_{o,\mathrm{eq}}(L_{\mathrm{val}})$ and obtain it by inverting the fitted curve for $L_{\mathrm{val}} > E_o$,

$$C_{o,\mathrm{eq}}(L_{\mathrm{val}}) = 10^{18} \left(\frac{A_o}{L_{\mathrm{val}} - E_o} \right)^{1/\alpha_o}.$$

Let C_{actual} be the compute used by the observed run. We define its compute efficiency CE_o as the ratio of equivalent compute to actual compute, and its penalty P_o as one minus this ratio,

$$\mathrm{CE}_o(L_{\mathrm{val}}) = \frac{C_{o,\mathrm{eq}}(L_{\mathrm{val}})}{C_{\mathrm{actual}}}, \qquad P_o(L_{\mathrm{val}}) = 1 - \mathrm{CE}_o(L_{\mathrm{val}}).$$

A penalty of zero means that the larger batch is as efficient as the 256K reference at the same loss. A positive penalty means that the larger batch uses more compute to reach that loss. For example, a penalty of 0.6 means that the reference curve reaches the observed loss with 40 percent of the run's compute. In the reported comparisons, each optimizer's own 256K anchor is assigned a penalty of zero by convention; the fitted curve need not pass exactly through that observation.

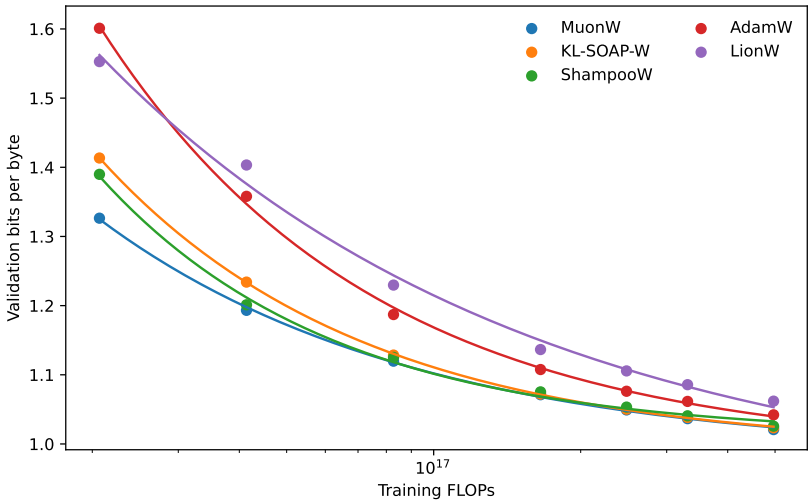

Figure 17: Validation losses across token budgets when the batch isze is 256K and the fitted loss–compute curve for each optimizer. Each curve is fit only to that optimizer's seven observations.

Using each optimizer's own 256K curve measures the efficiency lost when transferring that optimizer to a larger batch. Because the reference changes with the optimizer, these penalties do not rank absolute performance across optimizers. For a common comparison, we instead invert every observed loss through the MuonW 256K curve and divide by the run's actual compute. At the fixed budget used here, this shared reference preserves the ordering of absolute losses. Only the MuonW 256K anchor is assigned zero penalty by convention in the shared comparison; the other optimizers' 256K results are evaluated through the fitted MuonW curve.

H.6 RESULTS

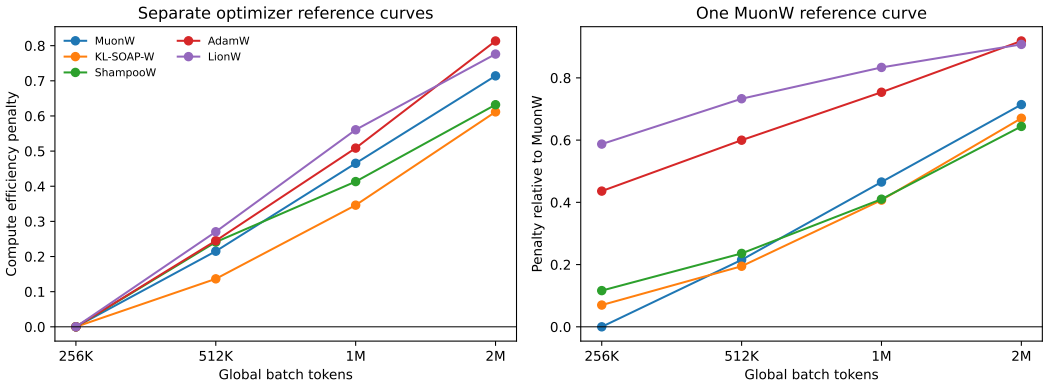

Figure 18: Compute efficiency penalties at a fixed horizon of eight tokens per scaling parameter. The left panel fits and uses a separate reference curve for every optimizer. The right panel maps every loss through the MuonW curve and therefore retains absolute loss differences. Lower is better.

KL-SOAP-W loses the least efficiency relative to its own reference. Its penalties are the smallest at every larger batch, reaching 0.1365 at 512K, 0.3462 at 1M, and 0.6116 at 2M. ShampooW is close at 2M with a penalty of 0.6320. MuonW reaches 0.7141, LionW reaches 0.7761, and AdamW reaches 0.8133 at the same batch. The fitted KL-SOAP-W reference therefore reaches its 2M-batch loss using about 39 percent of the compute spent by that run.

ShampooW achieves the lowest loss at the largest batch. KL-SOAP-W has the smallest common reference penalty at 512K and 1M. At 2M, ShampooW has penalty 0.6440 and KL-SOAP-W has penalty 0.6702. Their absolute validation losses are 1.1525 and 1.1614. Thus, retaining more efficiency relative to one's own reference does not imply achieving the lowest absolute loss.

This change in winner agrees qualitatively with the KL-SOAP-to-Shampoo crossover in Section 4. The present experiment provides evidence in a different setting under the specified transfer protocol. Its penalties measure the combined outcome of changing batch size and transferring hyperparameters, rather than the efficiency achievable after independent retuning at every batch.

B	Optimizer	L_{val} (BPB)	Compute-efficiency penalty	
			Own reference	Muon reference
256K	MuonW	1.0712	0.0000	0.0000
	KL-SOAP-W	1.0722	0.0000	0.0699
	ShampooW	1.0753	0.0000	0.1163
	AdamW	1.1076	0.0000	0.4361
	LionW	1.1365	0.0000	0.5872
512K	MuonW	1.0829	0.2151	0.2151
	KL-SOAP-W	1.0812	0.1365	0.1947
	ShampooW	1.0847	0.2412	0.2358
	AdamW	1.1397	0.2449	0.5997
	LionW	1.1885	0.2705	0.7330
1M	MuonW	1.1121	0.4653	0.4653
	KL-SOAP-W	1.1035	0.3462	0.4072
	ShampooW	1.1039	0.4133	0.4101
	AdamW	1.2000	0.5086	0.7538
	LionW	1.2651	0.5607	0.8335
2M	MuonW	1.1793	0.7141	0.7141
	KL-SOAP-W	1.1614	0.6116	0.6702
	ShampooW	1.1525	0.6320	0.6440
	AdamW	1.4387	0.8133	0.9185
	LionW	1.4016	0.7761	0.9076

Table 60: Compute efficiency penalties at the eight token per parameter horizon. Own curve penalty uses the compute loss curve fitted separately for each optimizer. Muon curve penalty maps every observed loss through the single MuonW curve.

All optimizer specific inversions in Table 60 fall within the loss range covered by their respective seven reference observations. Under the common MuonW curve, the AdamW and LionW results at 2M lie outside the directly observed MuonW loss range and are extrapolations. The remaining common reference values are interpolations.

H.7 Complete results

Table 61 contains the 50 observations used for the reference fits and batch comparisons. Table 62 gives the complete ranking of coefficient-transfer rules for KL-SOAP-W and ShampooW.

Table 61: Complete observations across token budgets and batch sizes used in the compute efficiency analysis.

Optimizer	τ/N	B (tokens)	$S(B)$	L_{val} (BPB)
MuonW	1	262,144	224	1.3264
	2	262,144	448	1.1932
	4	262,144	896	1.1197
	8	262,144	1,792	1.0712
	12	262,144	2,688	1.0492
	16	262,144	3,584	1.0366
	24	262,144	5,376	1.0208
	8	524,288	896	1.0829
	8	1,048,576	448	1.1121
	8	2,097,152	224	1.1793
KL-SOAP-W	1	262,144	224	1.4134
	2	262,144	448	1.2340
	4	262,144	896	1.1285
	8	262,144	1,792	1.0722
	12	262,144	2,688	1.0503
	16	262,144	3,584	1.0384
	24	262,144	5,376	1.0234
	8	524,288	896	1.0812
	8	1,048,576	448	1.1035
	8	2,097,152	224	1.1614
ShampooW	1	262,144	224	1.3898
	2	262,144	448	1.2012
	4	262,144	896	1.1251
	8	262,144	1,792	1.0753
	12	262,144	2,688	1.0533
	16	262,144	3,584	1.0407
	24	262,144	5,376	1.0257
	8	524,288	896	1.0847
	8	1,048,576	448	1.1039
	8	2,097,152	224	1.1525
AdamW	1	262,144	224	1.6010
	2	262,144	448	1.3580
	4	262,144	896	1.1871
	8	262,144	1,792	1.1076
	12	262,144	2,688	1.0762
	16	262,144	3,584	1.0615
	24	262,144	5,376	1.0421
	8	524,288	896	1.1397
	8	1,048,576	448	1.2000
	8	2,097,152	224	1.4387
LionW	1	262,144	224	1.5528
	2	262,144	448	1.4033
	4	262,144	896	1.2297
	8	262,144	1,792	1.1365
	12	262,144	2,688	1.1056
	16	262,144	3,584	1.0857
	24	262,144	5,376	1.0619
	8	524,288	896	1.1885
	8	1,048,576	448	1.2651
	8	2,097,152	224	1.4016

Table 62: Complete beta-transfer rule rankings. F keeps a coefficient fixed, and E applies EMA transfer. For ShampooW, the preconditioner shares the matrix β_2^{M} coefficient, so no separate preconditioner choice is shown. Mean BSZ regret averages the gap to the best tested rule for the same optimizer and batch size over the three target batches.

Rank	β_1^{M}	β_2^{M}	β_{sh}	β_1	β_2	512K	1M	2M	Mean BSZ regret
KL-SOAP-W									
1	**F**	**F**	**F**	**F**	**E**	**1.0812**	**1.1035**	**1.1614**	**0.00123**
2	F	F	F	F	F	1.0787	1.1033	1.1650	0.00153
3	F	E	F	F	F	1.0821	1.1042	1.1659	0.00327
4	F	E	F	F	E	1.0829	1.1045	1.1660	0.00367
5	F	F	F	E	E	1.0793	1.1069	1.2031	0.01563
6	F	E	F	E	E	1.0809	1.1067	1.2054	0.01687
7	F	E	F	E	F	1.0801	1.1087	1.2173	0.02123
8	F	F	F	E	F	1.0803	1.1119	1.2171	0.02230
9	E	E	F	F	E	1.0786	1.1097	1.2297	0.02520
10	E	E	E	F	E	1.0791	1.1126	1.2302	0.02650
11	E	E	E	F	F	1.0799	1.1136	1.2304	0.02717
12	E	E	F	F	F	1.0780	1.1115	1.2345	0.02720
13	E	E	F	E	E	1.0777	1.1175	1.2868	0.04653
14	E	E	E	E	E	1.0794	1.1209	1.2907	0.04953
15	E	E	E	E	F	1.0799	1.1204	1.2935	0.05047
16	E	E	F	E	F	1.0798	1.1191	1.2994	0.05197
17	E	F	F	F	E	1.0794	1.1372	1.3338	0.06933
18	F	F	E	F	E	1.0831	1.1271	1.3433	0.07037
19	E	F	F	F	F	1.0788	1.1380	1.3478	0.07407
20	E	F	F	E	E	1.0808	1.1429	1.3607	0.08067
21	F	E	E	F	F	1.0871	1.1519	1.3797	0.09210
22	F	F	E	F	F	1.0820	1.1449	1.3918	0.09210
23	F	E	E	F	E	1.0867	1.1594	1.3735	0.09240
24	E	F	F	E	F	1.0818	1.1460	1.3972	0.09420
25	E	F	E	F	E	1.0795	1.1449	1.4224	0.10147
26	E	F	E	E	E	1.0807	1.1505	1.4273	0.10537
27	E	F	E	F	F	1.0806	1.1521	1.4477	0.11267
28	F	F	E	E	E	1.0807	1.1799	1.4467	0.12163
29	E	F	E	E	F	1.0807	1.1594	1.5018	0.13317
30	F	E	E	E	E	1.0844	1.1929	1.5375	0.15747
31	F	F	E	E	F	1.0812	1.2512	1.5237	0.17123
32	F	E	E	E	F	1.0854	1.2774	1.6103	0.21023
ShampooW									
1	**F**	**F**	**–**	**F**	**E**	**1.0847**	**1.1039**	**1.1525**	**0.00040**
2	F	F	–	F	F	1.0845	1.1039	1.1568	0.00177
3	F	F	–	E	E	1.0848	1.1083	1.1885	0.01390
4	F	F	–	E	F	1.0846	1.1091	1.2043	0.01937
5	E	E	–	F	E	1.0842	1.1150	1.1991	0.01947
6	E	E	–	F	F	1.0835	1.1161	1.2009	0.02020
7	E	E	–	E	E	1.0851	1.1208	1.2316	0.03253
8	E	E	–	E	F	1.0849	1.1227	1.2399	0.03587
9	E	F	–	F	E	1.0877	1.1461	1.2772	0.05703
10	E	F	–	E	E	1.0877	1.1453	1.3130	0.06870
11	E	F	–	F	F	1.0869	1.1445	1.3184	0.06997
12	E	F	–	E	F	1.0879	1.1536	1.3394	0.08033
13	F	E	–	F	F	1.0842	1.1046	1.4766	0.10850
14	F	E	–	E	E	1.0841	1.1060	1.6090	0.15307
15	F	E	–	F	E	1.0844	1.1044	1.6558	0.16823
16	F	E	–	E	F	1.0847	1.1073	2.0020	0.28470