DELAYED EFFECTS OF TEMPORARY WEIGHT DECAY IN MUON PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

A temporary optimization intervention can increase validation loss when it ends yet reduce it at the final training budget. We demonstrate this with temporary weight decay in Muon-trained decoder-only Transformers with 162M, 405M, and 1.4B parameters. Within each scale, moving a fixed-duration decay window from early to late training reverses its final effect. At 162M, changing window duration alters final loss even at approximately matched total shrinkage. Most strikingly, a schedule selected at 162M and transferred without retuning to 1.4B is 0.26 nats worse than no decay when removed, overtakes it only after 73% of training, and ultimately finishes 0.028 nats better while also outperforming the best sampled constant-decay baseline. Early decay leaves a persistent weight-norm deficit, which keeps relative steps larger after the intervention ends. An approximate norm-dynamics recurrence captures the slow convergence of these relative steps, and fixed-norm replays of the measured trajectories reproduce the performance ordering and a similar loss difference. A separate Gram intervention exhibits related post-removal norm and relative-step dynamics. These results show that temporary optimization schedules have delayed effects long after removal and should therefore be evaluated at the intended final training budget, not only at removal.

1 INTRODUCTION

A temporary optimization intervention can increase validation loss when it ends and reduce it much later, even after the intervention has been removed. Figure 1 shows an extreme example. In a 1.4B-parameter Transformer (Vaswani et al., 2017) trained with Muon (Jordan et al., 2024), a temporary weight-decay schedule leaves validation loss 0.26 nats worse than no decay when the schedule ends. Both runs then continue with the same explicit update rule. Nevertheless, the treated run gradually recovers, overtakes the no-decay baseline only after 73% of training, and finishes 0.0275 nats better. Performance at the endpoint of an intervention can therefore be a poor predictor of its eventual effect.

Temporary changes during early training are known to have lasting effects on the learned solution (Achille et al., 2017; Frankle et al., 2020), and prior work has shown that regularization can matter disproportionately during early learning (Golatkar et al., 2019). We study a different aspect of this phenomenon: the effect of a temporary intervention can continue to evolve after the intervention has ended, to the point that its sign or ranking changes at later checkpoints.

This creates a basic problem for evaluating temporary optimization schedules. Such schedules are usually specified by an update rule and the interval over which it is applied. But when the intervention ends, the treated and reference runs generally enter the subsequent phase with different parameters and optimizer states. Their loss difference can therefore continue to evolve even though their explicit update rules are again identical. Evaluating an intervention when it is removed can consequently select a schedule that is inferior at the intended final training budget—or reject one that will eventually be beneficial.

We study this phenomenon using temporary weight decay in Muon-trained decoder-only Transformers. Weight decay provides a particularly controlled setting: turning it off cleanly separates its direct shrinkage from the training state it leaves behind. Muon also provides a useful connection between

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

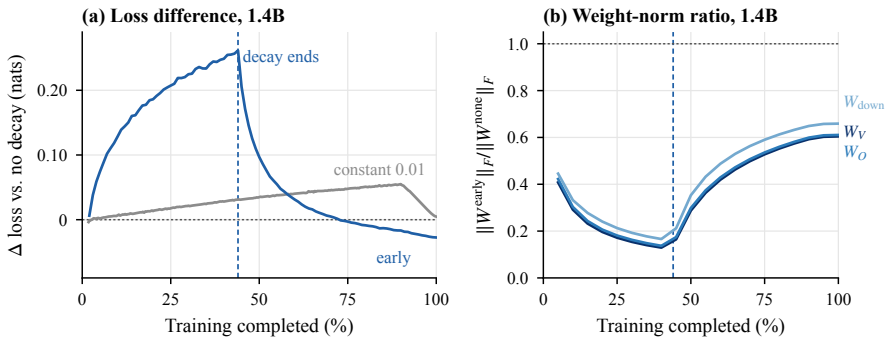


Figure 1: **A temporary intervention can be harmful at removal and beneficial much later.** 1.4B, FineWeb100B, 28,000 updates. Decay ($\lambda = 0.15$) is applied for the first 44% of training; the schedule was selected at 162M and transferred without retuning. It is 0.26 nats worse at removal, crosses no decay at 73% of training, and finishes 0.0275 nats better. (b) The norm deficit persists after removal.

weight norms and subsequent optimization. Its matrix updates have approximately controlled magnitude, so for a given projected update norm, reducing the weight norm increases the update size relative to the current weights (Jordan et al., 2024; Kosson et al., 2023; Wen et al., 2026; Liu et al., 2026). Temporary decay can therefore alter subsequent optimization even after its coefficient has returned to zero.

We ask two related questions:

> **Central question.** How should a temporary optimization intervention be evaluated, and what aspects of the training state carry its effect after the intervention is removed?

We address the first question by varying when decay starts, how long it lasts, and when its effect is evaluated. Across models from 162M to 1.4B parameters, moving a fixed-duration decay window from early to late training reverses its final effect: early windows improve final loss, whereas a window beginning at 60% of training hurts. At 405M, even among beneficial early schedules, the ranking changes depending on whether they are compared at a common final training budget or after the same amount of post-removal training. At 162M, changing window duration also changes final loss substantially even when total multiplicative shrinkage is approximately matched. These results separate three notions that are often conflated: when an intervention acts, how long the model trains after it is removed, and the final budget at which it is evaluated.

We address the second question by examining what remains different after decay is removed. Early decay leaves a persistent weight-norm deficit, which keeps relative steps larger than in the no-decay run long after both models resume the same update rule. An approximate recurrence accounts for the slow convergence of these relative-step trajectories. An opposite perturbation—boosting the learning rate early—raises weight norms and leaves relative steps smaller after the boost ends, as predicted by the same account. More directly, fixed-norm replays using the measured relative-step trajectories reproduce the performance ordering and a similar loss difference; swapping trajectories at the end of the decay window shows that most of the replay advantage arises from the altered steps after decay has ended. These experiments explain how an early intervention can continue to modify optimization after removal, although they do not establish that norms are the only persistent state or explain why a particular relative-step trajectory is beneficial.

Together, these experiments characterize both the evaluation consequences of delayed optimization effects and one mechanism by which an early intervention can remain active through the state it leaves behind.

2 Related Work

Early-training interventions. Temporary changes during early optimization can have lasting effects on the learned solution. Achille et al. (2017) and Frankle et al. (2020) study critical early phases in neural-network training, showing that perturbations applied near initialization can influence later optimization and the final solution. Most directly related, Golatkar et al. (2019) show in image classification that weight decay and data augmentation matter disproportionately during early learning and comparatively little near convergence. Our focus is different: we study what happens after a temporary intervention is removed. In our setting, the loss difference can continue to evolve for a large fraction of training, reverse sign, and change the ranking of schedules even though the treated and reference runs again use the same explicit update rule.

Weight norms and effective step size. Several recent works characterize the relation between weight decay, parameter norms, and effective learning rate. Kosson et al. (2023) analyze rotational equilibrium under weight decay; Wen et al. (2026) derive related equilibrium behavior for Muon and introduce Hyperball, which explicitly controls weight and update norms; Liu et al. (2026) connect effective learning rate to language-model loss dynamics; and Kosson et al. (2024) analyze the large relative steps produced by weights below equilibrium. These works characterize persistent coefficients, equilibrium behavior, or explicit norm control. We instead study the transient after temporary decay is removed: the weights can remain far below the no-decay norm trajectory, causing relative step sizes to remain different long after both runs resume the same explicit update rule.

Muon-specific analyses. Recent work has characterized Muon through spectral-norm constraints and row-norm control (Chen et al., 2025; Lion et al., 2026). Our experiments are complementary: rather than modifying Muon's norm control directly, we ask how an earlier temporary intervention changes the optimization trajectory that Muon follows afterward. The resulting norm deficit provides a concrete state variable through which the effect of a removed intervention can persist.

3 Experimental Setting and Evaluation

We train decoder-only Transformers with 162M, 405M, and 1.4B parameters. The main configuration uses Muon, a matrix optimizer that orthogonalizes momentum updates, with squared-ReLU (ReLU^2) MLPs, a context length of 1,024, and 1,024 sequences per update. The learning rate is 0.01 for the first 90% of training and decreases linearly to zero over the final 10%. The default configuration has no initial warmup; Section 6.3 tests 10% linear warmup and shows that the delayed crossover persists. The compute-optimal budget is about 20 tokens per parameter (Hoffmann et al., 2022). Matched intervention and reference runs share initialization, data order, and learning-rate schedule.

Main timing and scale experiments use FineWeb100B, a 100B-token sample of FineWeb, while the replicated mechanism experiments use FineWeb-Edu (Penedo et al., 2024). Table 2 in Appendix A gives the dataset and reference for each experiment group. Losses are compared only within matched groups.

Temporary decay. For each hidden-layer weight matrix, decoupled weight decay (Loshchilov & Hutter, 2017) gives

$$W_{t+1} = (1 - \eta_t \lambda_t) W_t - \eta_t U_t, \qquad \lambda_t = \lambda \mathbf{1}\{s \le t < e\}, \tag{1}$$

where U_t is the Muon update before multiplication by the learning rate η_t, and $[s, e)$ is the intervention window. Decay is zero outside this window unless a constant background coefficient is stated. Embeddings and the output head receive no decay.

Evaluation and references. All losses are next-token cross-entropy on held-out validation tokens, in nats per token (Appendix A). We report $\Delta_t = \mathcal{L}_{\mathrm{intervention}}(t) - \mathcal{L}_{\mathrm{reference}}(t)$ at update t; the final difference is Δ_T at the last update T, and negative values indicate an improvement. We call the end of the window *removal*. In the timing experiments, *early* windows start within the first 20% of training and the *late* window starts at 60%. The *best sampled constant* is the best of the constant coefficients we tried for that scale and training setting, and the *best reference* is the better of this

baseline and no decay. Claims that a schedule helps are made against both. We use no decay alone when asking what the window itself changes (Section 5), because constant decay adds norm dynamics of its own.

4 Temporary Decay Has Delayed Effects

Having established the delayed crossover in Figure 1, we now separate three properties of a temporary schedule: when the window starts, how long it lasts, and when its effect is evaluated. These factors need not agree in their ranking of schedules because they determine both the state at removal and the subsequent optimization trajectory.

4.1 A Short Early Window Improves Final Loss

We first verify the effect in a replicated comparison. Across five paired 162M runs, a 3200-update early-decay window improves final loss by 0.0172 nats over no decay and by 0.0074 nats over the best sampled constant (Appendix C, Table 5).

4.2 Start Time Changes the Sign; Evaluation Time Changes the Ranking

At all three scales, a 10% window starting at initialization beats both references, whereas the same-duration, same-coefficient window starting at 60% is worse than both (Figure 2(a); Appendix C). Within each scale these windows lie in the constant-learning-rate phase and apply the same multiplicative shrinkage, so the sign reversal is a timing effect rather than a difference in total shrinkage.

Comparing at the final checkpoint fixes total training, whereas comparing a fixed number of updates after removal fixes post-removal training. At 405M these two evaluations reverse the ranking of the early schedules (Figure 2(b)): the 20%-start window is better at the common final checkpoint, while the initialization-start window is better after equal post-removal training. For the 0% and 20% starts, the equal-continuation checkpoints remain in the constant-learning-rate phase, so the reversal is not caused by the final learning-rate decay; the 60% start is less directly comparable because its continuation enters that phase. Thus, the preferred early start depends on the evaluation horizon, although early starts help and the late start hurts under both comparisons.

4.3 Duration Changes Outcomes beyond Total Shrinkage

Across our grid, final loss is more sensitive to window duration than to the decay coefficient. At fixed coefficient, varying duration changes the final improvement by 0.026–0.035 nats; at fixed duration, varying the coefficient changes it by only 0.007–0.010 nats. We train 162M models for 16,000 updates, five times the compute-optimal budget, in a 4×3 grid of durations and coefficients, with every window starting at initialization. At each coefficient, longer windows give lower loss

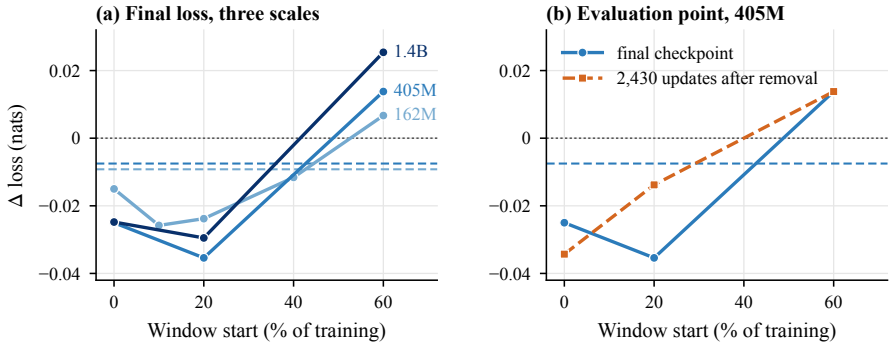


Figure 2: **Start time changes the sign; evaluation horizon changes the ranking.** (a) Final loss difference for 10% windows by start time; dashed lines show the best sampled constant. (b) At 405M, the ranking of early starts reverses when evaluated after equal post-removal training.

(Figure 3(a)). The best sampled constant, 0.05, improves on no decay by 0.0122. Ten of the twelve cells beat it; the two that do not are the shortest windows with the two smallest coefficients.

Longer windows also apply more decay in total. To separate the two effects, we compare schedules with approximately equal total log-shrinkage,

$$D = -\sum_t \log(1 - \eta_t \lambda_t). \tag{2}$$

For three schedules with $D \approx 2.1$, the magnitude of the final improvement more than doubles as the window lengthens from 700 to 3,564 updates (Figure 3(b)). Matching D fixes how much the initial weights are scaled down, but not how later updates are scaled (Appendix E).

The best schedule in this grid, $\lambda = 0.15$ over the first 44% of training, improves on the best sampled constant by 0.0335. We select this schedule here, at 162M, and transfer it without retuning to the 1.4B experiment in Figure 1 and Section 6.1. It is also the longest window in this grid, so these runs do not locate the optimal duration. Separate fixed-coefficient sweeps at the compute-optimal budget find an interior optimum at both 162M and 405M, beyond which longer windows hurt (Appendix C). Figure 7(a) in Appendix E shows how the loss difference evolves for three windows at $\lambda = 0.15$: each trails no decay while decay is on and overtakes it afterwards.

Together, these experiments show that a temporary schedule cannot be ranked from its removal-time loss or total shrinkage alone. Start time, duration, and evaluation horizon can all change the comparison, because they alter either the state at removal or the amount of subsequent optimization. We next ask which component of that state can preserve the effect of an earlier decay window.

5 A Persistent Norm Deficit Changes Later Relative Steps

Once decay is removed, the explicit update rule is again the same as in the no-decay run. Any continuing effect must therefore be carried by the parameters or optimizer state created during the intervention. Temporary decay can change weight directions, weight norms, and momentum. We focus on norms because they directly affect the size of a Muon update relative to the current weights. We first show that early decay leaves a long-lived norm deficit and correspondingly larger relative steps; we then test the relative-step trajectory directly with fixed-norm replays, before analyzing why the difference converges only slowly. This analysis explains how an early intervention can continue to modify optimization after removal, but does not establish that norms are the only persistent state or explain why a particular relative-step trajectory lowers loss.

5.1 Early decay creates a persistent norm deficit

In the transferred 1.4B run of Figure 1(b), norms fall to 12–16% of their no-decay values at removal and remain only 60-66% as large at the final checkpoint. The same qualitative deficit persists in the 162M mechanism runs.

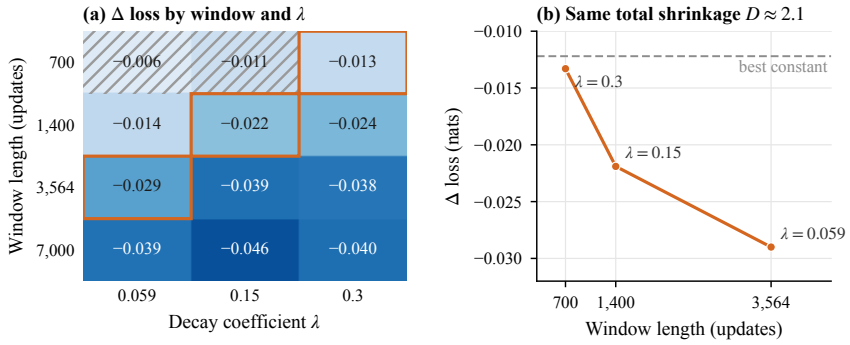


Figure 3: **Duration affects loss beyond total shrinkage.** 162M, FineWeb100B, 16,000 updates. (a) Final loss difference across duration and decay coefficient; hatching marks schedules that do not beat the best sampled constant. (b) At matched $D \approx 2.1$, longer, weaker decay performs better.

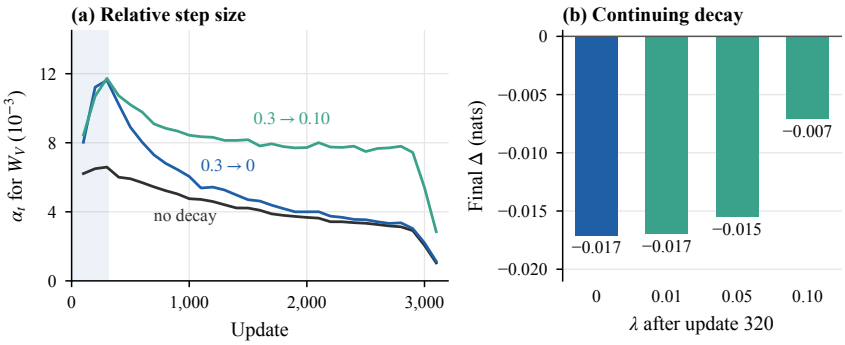


Figure 4: **Early decay increases relative step size, but larger relative steps do not always improve loss.** (a) Layer-averaged relative step size α_t for W_V; the shaded band is the decay window. (b) Final loss differences from no decay after using $\lambda = 0.3$ for the first 320 updates and then the indicated coefficient. 162M, FineWeb-Edu, 3,200 updates.

5.2 The norm deficit produces persistently larger relative steps

For the weight change $\Delta W_t = W_{t+1} - W_t$, we define the relative step size as the part of the change perpendicular to W_t, divided by the norm of W_t:

$$\alpha_t = \frac{\|P^{\perp}_{W_t} \Delta W_t\|_F}{\|W_t\|_F} = \eta_t \frac{\|P^{\perp}_{W_t} U_t\|_F}{\|W_t\|_F}, \tag{3}$$

where $P^{\perp}_W X = X - \langle W, X \rangle_F W / \|W\|^2_F$. This projection removes the decay term, which is parallel to W. For a fixed projected update norm, α_t is inversely proportional to $\|W_t\|_F$, as in norm-based analyses of the effective learning rate (Kosson et al., 2023; Wen et al., 2026; Liu et al., 2026).

At 162M, early decay lowers norms and increases α_t. After decay is removed, the relative steps approach the no-decay trajectory but remain larger until the end of training (Figure 4(a)). The norm deficit is not the whole story: at update 2,800, the inverse norm ratio is 1.17 whereas the relative-step ratio is 1.07, showing that changes in the projected update norm partly offset the effect of the smaller weights.

An early learning-rate boost provides the opposite perturbation. It raises weight norms and, as predicted by the same inverse relationship, leaves subsequent relative steps smaller than in the no-decay run. All three tested boosts finish with higher loss (Appendix G).

5.3 Replaying the relative-step trajectory reproduces the gain

The correlation between smaller norms and larger relative steps does not establish that the altered relative-step trajectory matters for loss. To test this directly, we remove norm variation altogether. Using a variant of Hyperball (Wen et al., 2026), we hold every weight norm at its initial value and prescribe the relative-step trajectory α_t measured from either the no-decay or early-decay Muon run, averaged over the six matrix families (Figure 5(a)).

Across three replay seeds, the trajectory measured from the early-decay run improves final loss by 0.022 to 0.023 nats relative to the no-decay trajectory, compared with a 0.0171-nat difference between the Muon runs from which the trajectories were taken (Figure 5(b)). Thus, with norms fixed, the measured relative-step trajectories reproduce both the ordering and a similar magnitude of the original loss difference.

A trajectory-swap experiment localizes where this difference arises. Using no-decay steps during the first 320 updates and early-decay steps afterward accounts for 0.0181 of the 0.0221-nat replay difference. Conversely, replaying only the early-decay trajectory during the intervention and then returning to ordinary Muon improves loss by only 0.0003 nats. Most of the replay advantage therefore comes from the altered relative steps after weight decay has already been removed (Figure 5(b)).

The replay uses a modified optimizer and averages α_t over matrix families, so it does not show that relative steps are the only relevant state. It does show that the observed post-removal relative-

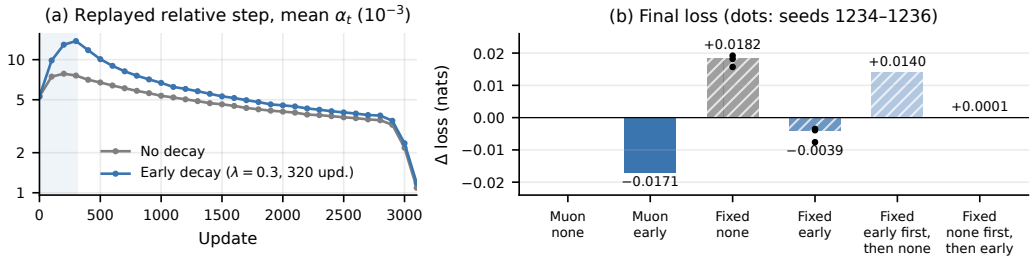


Figure 5: **Fixed-norm replay reproduces the post-removal gain.** 162M, FineWeb-Edu, 3,200 updates. (a) Relative-step trajectories measured from the source Muon runs and replayed at fixed weight norms. (b) Final loss relative to Muon without decay; dots show three replay seeds and the last two bars switch trajectory at update 320.

step trajectory is sufficient, in this controlled setting, to reproduce the performance ordering and approximate loss difference.

5.4 Why do the trajectories converge so slowly?

We next ask why the relative-step difference created by a short intervention decays so slowly.

Let $a_t = \|\Delta W_t\|_F / \|W_t\|_F$ and let c_t be the cosine between W_t and ΔW_t, so $\alpha_t = a_t\sqrt{1-c_t^2}$. From update 400 on, $|c_t| < 0.20$, so $\alpha_t/a_t > 0.98$. Expanding $\|W_t + \Delta W_t\|_F^2$ and approximating consecutive Muon updates as equal in norm gives

$$\frac{1}{a_{t+1}^2} - \frac{1}{a_t^2} \approx 1 + 2\kappa_t, \qquad \kappa_t = \frac{\langle W_t, \Delta W_t\rangle_F}{\|\Delta W_t\|_F^2}. \tag{4}$$

If the recurrence increments are similar in the early-decay and no-decay runs, their offset in $1/a_t^2$ remains approximately constant. Meanwhile $1/a_t^2$ itself grows. The fixed offset therefore becomes a progressively smaller fraction of the total, causing the ratio of relative steps to converge slowly rather than disappear immediately. Appendix G derives the recurrence from the exact norm identity and reports the corresponding checks across matrix families.

At 162M, κ_t is similar between the two runs, and the measured growth of $1/a_t^2$ agrees closely with the recurrence for four matrix families. From 400 to 2,800, $1/\alpha_t^2$ grows roughly fourfold whereas the gap between the runs changes much less, and the relative-step ratio falls only from about 1.6–1.8 to about 1.1. Appendix G quantifies these deviations and shows the per-family diagnostics.

The early learning-rate boosts introduced in Section 5.2 provide an opposite-sign check of this account: they raise the norms and produce persistent reductions in later relative steps, with the measured offsets behaving as predicted by the same recurrence (Appendix G).

At 1.4B, we do not have the corresponding relative-step logs for the no-decay run, so we cannot test the recurrence as directly. We can nevertheless track the norm deficit itself. From update 14,000 to 25,200, the layer-averaged squared-norm gap between the no-decay and early-decay runs grows by 39% for W_V and W_O and 17% for W_{down} while the squared norms in the no-decay run roughly double. Thus, the absolute deficit does not remain fixed, but it becomes progressively smaller relative to the growing weight norms, consistent with the slow recovery seen at 162M.

What norms do not explain. The norm account explains why an early intervention can continue to alter optimization after removal, but not which relative-step trajectory is beneficial. Larger relative steps are not universally better. Continuing with decay coefficient 0.10 after the window keeps relative steps larger than stopping decay, yet yields a smaller final improvement (0.0071 versus 0.0171 nats (Figure 4(b)). Likewise, a window starting at 60% of training shrinks the weights by the same factor as one starting at initialization but hurts final loss (Figure 2(a)). Thus, persistent norm and relative-step differences explain the delayed influence of the intervention, not why early and late trajectories have opposite effects.

6 SCALE TRANSFER AND ROBUSTNESS

We next test whether the delayed effect persists across model scale and training conditions. We first transfer a schedule selected at 162M directly to 1.4B without retuning, then compare shorter windows across scale and test robustness to longer training, warmup, architecture, and optimizer choice.

6.1 A SCHEDULE SELECTED AT 162M TRANSFERS TO 1.4B

We transfer the best schedule from the 162M duration sweep—$\lambda = 0.15$ for the first 44% of training—to a 1.4B model without retuning. It finishes 0.0275 nats ahead of no decay and 0.0264 ahead of the best sampled constant-decay baseline; as Figure 1 shows, this advantage emerges entirely after removal. The best schedule sampled directly at 1.4B improves on the transferred schedule by only 0.0020 nats.

6.2 THE EFFECT PERSISTS ACROSS SCALE

A 10% decay window starting at initialization beats both no decay and the best sampled constant at every scale (Table 1). At 405M, the first-10% window improves over no decay by 0.0247–0.0262 nats across three matched seeds, and over constant 0.02 by 0.0172–0.0194 nats. More broadly, every tested constant coefficient improves on no decay at 162M and 405M, whereas at 1.4B only the two smallest tested coefficients do so, and only slightly. These comparisons show that the benefit of an early window persists across the tested scales, but not that its magnitude follows a scaling law: the fixed 10% schedule is also farther from the best sampled duration at 162M than at the larger scales. Appendix C reports the complete constant and schedule sweep.

6.3 ROBUSTNESS TO TRAINING CONDITIONS

The effect is not limited to the compute-optimal, no-warmup configuration. At 162M trained for 64,000 updates—$20\times$ the compute-optimal budget—a first-10% window still beats the best sampled constant by 0.0180 nats (Appendix C). With 10% linear warmup, a decay window confined to warmup ties the best sampled constant at 162M and beats it at 405M and 1.4B; extending decay beyond warmup restores a clear advantage at 162M and 405M. Importantly, warmup does not eliminate the delayed crossover: in a 16,000-update 162M run, temporary decay is 0.0237 nats worse than no decay at update 8,000 but 0.0373 nats better by update 14,400 (Appendix F). Early decay also improves loss with SwiGLU MLPs and with AdamW at 162M, but not with AdamW at 405M (Appendix C). Thus, the phenomenon is robust to several training variations but is not optimizer-independent.

Across four data domains, temporary decay improves on constant 0.01 both at removal and at the final checkpoint, but the domain ranking changes between the two evaluations; Appendix D reports these comparisons and exploratory corpus-level analyses.

Table 1: **Early decay across scale.** FineWeb100B, compute-optimal budgets, no warmup. "Best const." is the best sampled constant coefficient; full sweeps are in Appendix C.

Scale	Updates	Early schedule (λ, window)	Early loss	No decay	Best const.	vs. best ref.
162M	3,200	0.3, 10%	3.3710	3.3860	3.3768 (0.05)	-0.0058
405M	8,100	0.3, 10%	3.0287	3.0537	3.0462 (0.02)	-0.0175
1.4B	28,000	0.2, 10%	2.6506	2.6754	2.6743 (0.003)	-0.0237
1.4B	28,000	0.15, 44% (transferred)	2.6479	2.6754	2.6743 (0.003)	-0.0264

7 A Second Norm-Changing Intervention Shows Related Dynamics

The preceding mechanism could be specific to the multiplicative form of weight decay. We therefore test a qualitatively different intervention that also changes weight norms. During an early window, we add the off-diagonal Gram penalty

$$\mathcal{R}_t = \mu \mathbf{1}0 \le t < e \sum_{\ell=1}^{L} \sum_{W \in W_V, W_O, W_{\text{down}}} E(W_\ell), \qquad E(W) = |\operatorname{OffDiag}(W^\top W)|_F^2. \tag{5}$$

Muon orthogonalizes the combined task and penalty gradient, so the applied update is not the raw penalty gradient (Appendix A).

The Gram penalty exhibits the same qualitative timing dependence as temporary decay. At 162M, 10% windows starting at 0%, 20%, and 60% of training change final loss by -0.0147, -0.0080, and +0.0209 nats, respectively. Across five paired 162M runs, an early Gram penalty improves final loss by 0.0142 nats relative to no regularization.

The raw penalty gradient contains a radial component that changes the weight norm and a tangential component that changes only direction to first order. Radial–tangential and state-copy controls indicate that norm changes account for much of the improvement: when applied after the Muon update, the radial component nearly matches the full penalty while the tangential component has little effect, and copying column norms recovers more of the advantage than copying normalized column correlations (Appendix E). The penalty also leaves larger post-removal relative steps primarily in the matrices it targets (Appendix G).

Early Gram regularization also improves loss at 162M and 405M across both tested datasets, while 1.4B requires retuning of its strength (Appendix B).

Together, these results provide supporting evidence that the post-removal norm and relative-step dynamics of Section 5 are not unique to multiplicative weight decay, without implying that norm reduction is generally beneficial.

8 Discussion and Conclusion

The endpoint of a temporary optimization intervention need not be the endpoint of its effect. In Muon pretraining, temporary decay can be worse at removal yet better at the final budget; both the sign and ranking of schedules can change after removal. This behavior persists across the tested scales, and a schedule selected at 162M transfers without retuning to 1.4B.

Our mechanism experiments identify one way this delayed effect can be carried after removal. Early decay leaves a persistent weight-norm deficit and correspondingly larger relative steps; an approximate recurrence explains their slow convergence, and fixed-norm replays show that the measured relative-step trajectory is sufficient to reproduce the performance ordering and a similar loss difference. A separate Gram intervention exhibits related timing, norm, and post-removal relative-step behavior. These results explain how an early intervention can continue to alter optimization after it is removed, but not why a particular relative-step trajectory is beneficial: larger relative steps are not universally better, and we do not establish that norms are the only persistent state.

The practical implication is methodological: temporary schedules should be evaluated at the training budget at which the model will be used, not only when the intervention ends. Our conclusions are specific to the tested regimes. Most configurations outside the replicated comparisons use a single seed (Appendix A). The numerical decay coefficients are not universal; detailed mechanism measurements are concentrated at 162M; some robustness comparisons use constant 0.01 rather than a full constant-decay sweep; and early decay does not improve every tested optimizer setting. Finally, the downstream evaluations do not show a clear advantage over constant decay, so the evidence here concerns pretraining validation loss rather than downstream performance. Whether similar delayed dynamics arise broadly across optimizers and temporary interventions remains an open question.

AI USE STATEMENT

Generative AI tools were used to help edit the manuscript, and write and debug code. The authors reviewed all AI-assisted content and take full responsibility for the manuscript, including its claims, results, and supporting code.

REPRODUCIBILITY STATEMENT

Appendix A describes the models, datasets, optimizer settings, learning-rate schedules. Training code is based on publicly available code: https://github.com/microsoft/dion. Appendices B and C give the Gram settings and the decay schedules; Appendix E derives the Gram decomposition; Appendix G derives the relative-step recursion and describes the relative-step measurements and the fixed-norm replay procedure.

REFERENCES

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical Learning Periods in Deep Neural Networks. arXiv preprint arXiv:1711.08856, 2017. URL https://arxiv.org/abs/1711.08856.

Lizhang Chen, Jonathan Li, and Qiang Liu. Muon Optimizes Under Spectral Norm Constraints. arXiv preprint arXiv:2506.15054, 2025. URL https://arxiv.org/abs/2506.15054.

CodeParrot. GitHub Code Clean. Hugging Face dataset, n.d. URL https://huggingface.co/datasets/codeparrot/github-code-clean.

Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The Early Phase of Neural Network Training. arXiv preprint arXiv:2002.10365, 2020. URL https://arxiv.org/abs/2002.10365.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Time Matters in Regularizing Deep Networks: Weight Decay and Data Augmentation Affect Early Learning Dynamics, Matter Little Near Convergence. arXiv preprint arXiv:1905.13277, 2019. URL https://arxiv.org/abs/1905.13277.

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

Shawn Im, Changdae Oh, Zhen Fang, and Sharon Li. How do transformers learn to associate tokens: Gradient leading terms bring mechanistic interpretability. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=A4Us8jxVGq.

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/.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Rotational Equilibrium: How Weight Decay Balances Learning Across Neural Networks. arXiv preprint arXiv:2305.17212, 2023. URL https://arxiv.org/abs/2305.17212.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Analyzing & Reducing the Need for Learning Rate Warmup in GPT Training. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2410.23922.

Kai Lion, Florian Hübler, Bingcong Li, Antonio Orvieto, and Niao He. Muown: Row-Norm Control for Muon Optimization. arXiv preprint arXiv:2605.10797, 2026. URL https://arxiv.org/abs/2605.10797.

Zihan Liu, Ruiheng Zheng, Shaobo Zhang, Changxin Tian, Kunlong Chen, Zhiqiang Zhang, and Lei Wu. Effective Learning Rate Governs Loss Dynamics in Language Model Pretraining. arXiv preprint arXiv:2608.24814, 2026. URL https://arxiv.org/abs/2608.24814.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. arXiv preprint arXiv:1711.05101, 2017. URL https://arxiv.org/abs/1711.05101.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. OpenWebMath: An Open Dataset of High-Quality Mathematical Web Text. arXiv preprint arXiv:2310.06786, 2023. URL https://arxiv.org/abs/2310.06786.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv preprint arXiv:1910.10683, 2019. URL https://arxiv.org/abs/1910.10683.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic Pretraining Optimizers and Where to Find Them II: Hyperball Optimization. arXiv preprint arXiv:2606.16899, 2026. URL https://arxiv.org/abs/2606.16899.

Appendix A gives the training and evaluation details. Appendices B and C report additional Gram and weight-decay comparisons. Appendices D and E contain supporting domain and component analyses. Appendix F covers warmup and downstream evaluation, and Appendix G gives the norm-dynamics checks.

A Training and Evaluation Details

Experiment map Main timing and scale experiments use FineWeb100B, a 100B-token sample of FineWeb, while the replicated mechanism experiments use FineWeb-Edu (Penedo et al., 2024). Table 2 gives the dataset and reference for each experiment group. Losses are compared only within matched groups. Domain experiments additionally use OpenWebMath (Paster et al., 2023), GitHub-Code-Clean (CodeParrot, n.d.), and multilingual C4 (Raffel et al., 2019).

Table 2: **Experiment map.** Each result is compared only with the reference in the same row.

Purpose	Data	Scale	Reference
Replicated comparison	FineWeb-Edu	162M	No decay; best constant
Timing and scale	FineWeb100B	162M–1.4B	No decay; best constant
Norm and replay controls	FineWeb-Edu	162M	No decay
Warmup	FineWeb-Edu; FineWeb100B	162M–1.4B	No decay; best constant
Domain comparison	Four corpora	162M	Constant 0.01
Gram penalty across scale	FineWeb-Edu; FineWeb100B	162M–1.4B	Constant 0.01

Model and optimizer. The models use pre-norm Transformer blocks, parameter-free RMS normalization, QK normalization, rotary position embeddings with base 10,000, a vocabulary of 50,304, and untied input and output embeddings. The main MLP uses a squared-ReLU activation and hidden width $4d$, where d is the model width. Table 3 lists the dimensions and compute-optimal budgets.

Table 3: Model dimensions and compute-optimal budgets. Every update processes 1,024 sequences of 1,024 tokens. The duration grid uses 16,000 updates at 162M.

Parameters	Width	Layers	Heads	Updates
162M	768	12	6	3,200
405M	1,024	24	32	8,100
1.4B	2,048	24	32	28,000

Muon uses learning rate 0.01, momentum 0.95, spectral-norm learning-rate adjustment, and no gradient clipping. Embeddings and the output head use AdamW with $\beta = (0.95, 0.98)$ and no decay. The learning rate decreases linearly to zero over the final 10% of updates, and there is no warmup except in Appendix F. Matrices are initialized with standard deviation $\min(1, \sqrt{d_{\text{out}}/d_{\text{in}}})/\sqrt{d_{\text{in}}}$, and embeddings with unit standard deviation. Each run uses eight H100 GPUs, with gradient accumulation where needed to reach the global batch.

Data and evaluation. All corpora use the GPT-2 tokenizer, with documents separated by its end-of-text token. The mathematical, code, and multilingual corpora each contain 2B training tokens and a 50M-token validation shard. Validation loss is computed on 10,485,760 tokens, usually every 100 updates. The domain analyses instead use a fixed set of 33,554,432 validation targets for each paired comparison, so their losses differ slightly from the training-log values.

Interventions. Weight-decay windows use the half-open interval $[s, e)$ of Equation 1. The Gram term in Equation 5 uses the input-side Gram matrix $W^\top W$ of each target matrix and is summed over targets and layers without normalization. In the five-seed 162M experiment, $\mu = 10^{-3}$, $e = 320$, and simultaneous decay uses $\lambda = 0.3$ over the same window. Relative steps are computed from the weight change between consecutive updates and averaged over the layers of each matrix family.

Seeds. The replicated comparisons are the five-seed 162M experiment (Table 5), the three-seed 405M comparison of early decay and no decay, the four-seed warmup experiment, the two-seed matrix-family comparison, and the three-seed fixed-norm replay. Other configurations use a single seed. Losses are compared only within matched groups.

B Gram-Penalty Scale and Target Ablations

Spectral motivation. In initial AdamW runs, the stable rank $\|W\|_F^2/\|W\|_2^2$, which is large when many singular values are of similar size and small when a few dominate, decreases during early training, particularly in W_V, W_O, and W_{down}. This separate observation motivated the Gram targets but does not explain the gains in the Muon experiments.

The Gram comparisons in Section 7 keep constant decay of 0.01 throughout training and add the penalty on $\{W_V, W_O, W_{\text{down}}\}$ during the first 10% of updates. Table 4 shows that the penalty lowers loss by 0.013 to 0.015 nats at 162M and 405M on both datasets. At 1.4B, $\mu = 10^{-3}$ is too strong and $\mu = 10^{-4}$ helps.

Table 4: Gram regularization across scale. Differences are from the constant-0.01 reference, in nats. The 1.4B reference (2.6796) is a separate run of the same constant-0.01 configuration; the run in Table 7 reaches 2.6802.

Data	Scale	μ	Reference	Gram	Difference
FineWeb-Edu	162M	10^{-3}	3.1829	3.1699	-0.0130
FineWeb-Edu	405M	10^{-3}	2.8551	2.8421	-0.0130
FineWeb100B	162M	10^{-3}	3.3820	3.3673	-0.0147
FineWeb100B	405M	10^{-3}	3.0480	3.0332	-0.0148
FineWeb100B	1.4B	10^{-3}	2.6796	2.6813	$+0.0017$
FineWeb100B	1.4B	10^{-4}	2.6796	2.6742	-0.0054

Strength and targets. At 405M on FineWeb-Edu, every μ from 10^{-4} to 10^{-2} improves on the reference, with gains between 0.009 and 0.013 nats. Target ablations identify W_{down} as the main contributor: W_{down} alone recovers 0.0117 of the 0.0130-nat three-target gain, whereas target sets that exclude it improve loss by at most 0.0005 nats. Penalizing all six matrix families changes loss by $+0.0068$ nats at 405M but -0.0071 at 162M. SwiGLU experiments also favor W_{down} over all three targets, so the broad six-family penalty is not a reliable replacement.

Timing. The Gram penalty has the same timing pattern as weight decay. At 162M on FineWeb-Edu (3,200 updates), a 320-update penalty starting at 0%, 20%, and 60% of training changes final loss by -0.0147, -0.0080, and $+0.0209$ nats; decay with $\lambda = 0.3$ in the same windows gives -0.0177, -0.0241, and $+0.0063$.

C Replication, Schedule Sweeps, and Optimizer Variants

Replicated intervention comparison and baseline comparison. Table 5 summarizes the matched 162M comparison. Each individual intervention improves on no regularization. Combining the Gram penalty with weight decay gives a smaller additional change than either intervention gives relative to no regularization. For "Both", the seed-1234 run computes the penalty gradient directly, and the other seeds use the radial–tangential split with both parts at full scale; the two implementations have the same gradient.

Start time. Figure 2 uses 10% windows starting at 0%, 10%, 20%, 40%, and 60% of training at 162M, and at 0%, 20%, and 60% at 405M and 1.4B. The coefficient is $\lambda = 0.3$ at 162M and 405M and $\lambda = 0.2$ at 1.4B. The 405M comparison at equal continuation evaluates the three runs at updates 3,240, 4,860, and 8,100, each 2,430 updates after removal.

Table 5: Mean final loss in five matched 162M runs (FineWeb-Edu, 3,200 updates). Temporary interventions end at update 320. Values are mean ± sample standard deviation. The constant coefficient 0.05 is the best sampled from $\{0.01, 0.05, 0.1, 0.2, 0.3\}$.

Intervention	Final loss	Difference from none
No regularization	3.1873 ± 0.0028	0
Constant decay ($\lambda = 0.05$)	3.1775 ± 0.0028	-0.0098
Gram penalty	3.1732 ± 0.0027	-0.0142
Early weight decay ($\lambda = 0.3$)	3.1701 ± 0.0027	-0.0172
Both	3.1682 ± 0.0030	-0.0191

Window length. In the duration grid, constant 0.05 is the best constant baseline, improving on no decay by 0.0122 nats. For a window in the constant-learning-rate phase, $D = -e \log(1 - 0.01\lambda)$; the three outlined cells in Figure 3 all have $D \approx 2.10$. Longer windows help only up to a point (Table 6): the gain is largest for a window over 50% of training at 162M and 20% at 405M. The 75% window hurts at 405M, and decay throughout training hurts at both scales.

Table 6: Final loss difference from no decay for windows starting at initialization with $\lambda = 0.3$, by window length as a fraction of training (FineWeb-Edu; 3,200 updates at 162M and 8,100 at 405M). The 162M row uses seed 1235. Differences in nats; "–" marks lengths we did not run.

Scale	5%	10%	20%	50%	75%	100%
162M	–	-0.0177	-0.0302	$\mathbf{-0.0374}$	-0.0140	$+0.0279$
405M	-0.0145	-0.0249	$\mathbf{-0.0352}$	-0.0283	$+0.0112$	$+0.0880$

Scale. Table 7 lists the final losses behind Table 1, including every constant coefficient we ran. The transferred schedule keeps $\lambda = 0.15$ and rounds the window to 44% of training (12,320 of 28,000 updates). It is 0.2616 nats worse than no decay at removal and first overtakes it at update 20,440. The best sampled schedules are $\lambda = 0.3$ for the first 1,400 updates at 162M, $\lambda = 0.1$ for the first 2,100 updates at 405M, and $\lambda = 0.2$ from 20% to 30% of training at 1.4B. At 405M we repeated no decay and the first-10% window with seeds 1234, 1235, and 1236; early decay improves on no decay by 0.0250, 0.0262, and 0.0247 nats. Against constant 0.02, the paired improvements are 0.0175, 0.0194, and 0.0172 nats. The seed-1234 values use the run in Table 1.

Table 7: Final losses behind Table 1 (FineWeb100B, compute-optimal budget, no warmup, seed 1234). The best sampled reference at each scale is in bold. "–" marks coefficients we did not run.

Schedule	162M	405M	1.4B
No decay	3.3860	3.0537	2.6754
Constant 0.001	3.3852	3.0529	2.6744
Constant 0.003	–	–	**2.6743**
Constant 0.005	3.3836	3.0502	–
Constant 0.01	3.3820	3.0480	2.6802
Constant 0.02	3.3796	**3.0462**	–
Constant 0.05	**3.3768**	3.0497	2.7151
Constant 0.10	3.3805	–	2.7558
First 10% (λ)	3.3710 (0.3)	3.0287 (0.3)	2.6506 (0.2)
Transferred, $\lambda = 0.15$ for 44%	–	–	2.6479
Best sampled schedule	3.3510	3.0180	2.6459

Longer training. Table 8 lists 162M runs on FineWeb100B with 64,000 updates, 20 times the compute-optimal budget. "Stop at reduction" applies decay until the final learning-rate reduction begins at update 57,600. We report the best coefficient for each schedule class.

Table 8: 162M runs with 64,000 updates. Differences from no decay, in nats.

Schedule	Final loss	Difference
No decay	3.0826	0
Best constant (0.01)	3.0665	−0.0161
Best stop at reduction (0.05)	3.0617	−0.0209
First 10%, $\lambda = 0.2$	3.0485	−0.0341

Other optimizers and MLPs. Table 9 compares early decay with no decay in other settings. Early decay helps in every listed setting except AdamW with ReLU^2 at 405M. The Muon and AdamW rows also differ in learning rate, warmup, and coefficient grid, so this exception cannot be attributed to the optimizer alone. The sampled 162M AdamW and SwiGLU runs also improve on constant 0.01.

Table 9: Best sampled first-10% window versus no decay. FineWeb-Edu at the compute-optimal budget, except the Muon–ReLU^2 1.4B row (FineWeb100B, 6,000 updates). Muon rows use no warmup and AdamW rows use 10% warmup. For AdamW at 405M, λ is the smallest coefficient tried. Differences in nats.

Optimizer	MLP	Scale	No decay	Early λ	Difference
Muon	ReLU^2	162M	3.1864	0.3	−0.0171
Muon	ReLU^2	405M	2.8601	0.3	−0.0249
Muon	ReLU^2	1.4B	2.8811	0.2	−0.0181
Muon	SwiGLU	162M	3.2019	0.3	−0.0189
Muon	SwiGLU	405M	2.8790	0.3	−0.0235
AdamW	SwiGLU	162M	3.2401	3.0	−0.0168
AdamW	ReLU^2	162M	3.2190	3.0	−0.0157
AdamW	ReLU^2	405M	2.9134	1.5	+0.0109

D Domain Comparison

Data effects across domains. We compare temporary decay with constant 0.01 in matched 162M runs on web, mathematics, multilingual text, and code. Temporary decay improves loss in all four domains both when decay is removed and at the final checkpoint, but the ranking changes substantially between the two evaluations (Figure 6; Table 10). In particular, the domains with the largest immediate gains do not have the largest final gains.

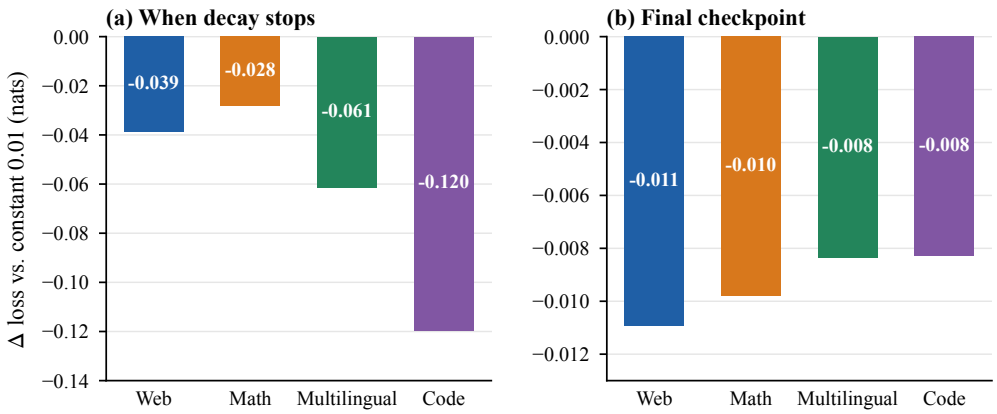


Figure 6: **The domain ranking changes between removal and the final checkpoint.** Validation-loss differences from constant 0.01 when decay is removed and at the final checkpoint; negative values favor temporary decay. Each comparison uses a 162M model; temporary decay uses $\lambda = 0.3$ for the first 10% of training. The vertical scales differ.

Table 10: Domain cross-entropy and paired loss differences. Temporary decay uses $\lambda = 0.3$ for 320 of 3,200 updates; the reference uses constant 0.01. All values in nats.

Domain	H_{bi}	Δ_{320}	Δ_{3200}
Web	5.982	−0.0388	−0.0109
Mathematics	5.513	−0.0284	−0.0098
Multilingual	4.245	−0.0614	−0.0083
Code	3.815	−0.1196	−0.0083

Bigram cross-entropy. Motivated by recent work studying the emergence of token associations during early Transformer training (Im et al., 2026), we use bigram cross-entropy as a simple measure of how strongly next-token prediction can be supported by local token associations.

For each corpus, we fit unigram and bigram counts on three 100M-token training shards and apply additive smoothing toward the training unigram distribution with strength 5. We evaluate the resulting bigram cross-entropy H_{bi} on the same validation targets as the paired model losses. This quantity measures how difficult a domain is to predict from the previous token alone.

Data repetition. The mathematical, code, and multilingual runs make 1.68 passes over 2B-token training pools, whereas the web runs never repeat data. We therefore repeated the web comparison on a 2B-token FineWeb-Edu pool with the same 1.68 passes. Early decay improves on constant 0.01 by 0.0349 nats at removal and by 0.0128 at the final checkpoint, so web text again has a small gain at removal and the largest final gain. Repetition therefore does not explain the web result. These losses come from the training log, and this run is not included in Table 10.

E Gram Controls and Schedule Algebra

Table 11 summarizes the three controls used to separate norm and directional effects. The following paragraphs provide the corresponding implementation details and compact summaries of the coefficient sweeps.

Gradient-component sweep. In the 162M experiment, adding $\mu(ag_{\parallel} + bg_{\perp})$ before Muon does not reproduce the full penalty when either component is used alone. Across the tested coefficients, norm-only runs end between 3.1887 and 3.1981, direction-only runs end between 3.1868 and 3.1870, and the full penalty reaches 3.1718; no regularization reaches 3.1864. Table 11 sum-

Table 11: Gram-gradient components and copied Gram state. All runs use 162M models, FineWeb-Edu, 3,200 updates, and seed 1234; regularization ends at update 320. "Before Muon" adds a component to the task gradient before orthogonalization. "After Muon" applies it directly to the weights. "Copied state" continues run A after replacing the indicated Gram statistics with those from penalized run B. Recovery is $(\mathcal{L}_A - \mathcal{L})/(\mathcal{L}_A - \mathcal{L}_B)$. Before-Muon runs use $\mu = 10^{-3}$ and no background decay (no regularization: 3.1864). After-Muon runs use constant background decay 0.01 and $\mu = 0.01$ (without a penalty: 3.1829). Copied-state runs also use constant background decay 0.01 but restart from saved checkpoints; their reference is the $A \to B$ row. Losses are comparable only within a block.

	Contribution	Final loss
Before Muon	Full: $g_{\parallel} + g_{\perp}$	3.1718
	Norm only: $ag_{\parallel}$	3.1887–3.1981
	Direction only: $g_{\perp}$	3.1868
	Full plus early weight decay ($\lambda = 0.3$)	3.1663
	Direction plus early weight decay ($\lambda = 0.3$)	3.1710
After Muon	Full penalty gradient	3.1750
	Norm only	3.1745
	Direction only	3.1830
Copied state	Normalized column correlations (27% recovery)	3.1833
	Column norms (62%)	3.1786
	Full Gram matrix (81%)	3.1761
	A → B endpoints	3.1868 → 3.1736

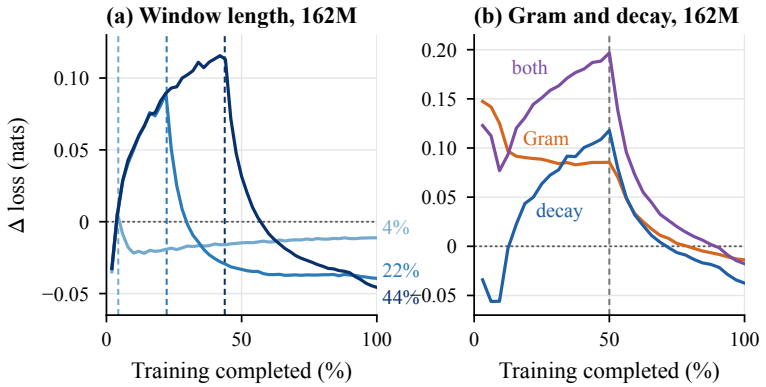


Figure 7: **Loss differences evolve after removal.** (a) Three 162M decay windows at $\lambda = 0.15$ (FineWeb100B). (b) Gram regularization, weight decay, and their combination over the first 50% at 162M (FineWeb-Edu). Dashed lines mark removal. References use no regularization; vertical scales differ.

marizes the more informative controls that apply the components after Muon or copy the resulting state.

Gram penalty combined with weight decay. At 162M, we keep $\mu = 10^{-3}$ and $\lambda = 0.3$ fixed and vary their shared window over the first 10%, 20%, or 50% of training. The combination always beats no regularization, but its gain is smaller than the sum of the individual gains. For the 20% and 50% windows, it also underperforms weight decay alone; for the 10% window, it improves on weight decay alone (-0.0191 versus -0.0177, seed 1235). Figure 7(b) shows the 50% trajectories. At 405M with a 10% window, the combination is worse than weight decay alone with $\lambda = 0.3$ (2.8455 versus 2.8352) and better with $\lambda = 0.12$ (2.8364 versus 2.8413).

Matrix families. With constant 0.01 decay kept on all matrices, adding early decay to either $\{W_V, W_O, W_{\text{down}}\}$ or $\{W_Q, W_K, W_{\text{up}}\}$ gives similar gains of 0.008 nats. Applying it to all six

families approximately adds these gains. At a matched decay budget, distributing decay over all six families also performs better than concentrating it on the three Gram targets.

Gram-gradient decomposition. Write $G = W^\top W$ and $C = \mathrm{OffDiag}(G)$. Since C is symmetric with zero diagonal,

$$dE = 2\langle C, dG\rangle_F = 2\langle C, W^\top dW + (dW)^\top W\rangle_F = 4\langle WC, dW\rangle_F. \tag{6}$$

Hence $\nabla E = 4WC$ and $\langle W, \nabla E\rangle_F = 4\,\mathrm{tr}(GC) = 4\,\mathrm{tr}(C^2) = 4E$. Projecting onto W gives, for $W \neq 0$,

$$\nabla E(W) = 4WC = \underbrace{\frac{4E(W)}{\|W\|_F^2}W}_{g_\parallel} + \underbrace{\left(4WC - \frac{4E(W)}{\|W\|_F^2}W\right)}_{g_\perp}, \qquad \langle W, g_\perp\rangle_F = 0. \tag{7}$$

This identity holds for the raw gradient. It does not imply that the orthogonalized Muon update has the same decomposition.

Total shrinkage. Let $q_t = 1 - \eta_t\lambda_t$ and $A_{a:b} = \prod_{j=a}^{b-1} q_j$, with $A_{b:b} = 1$. Repeated substitution in Equation 1 gives

$$W_T = A_{0:T}W_0 - \sum_{k=0}^{T-1} \eta_k A_{k+1:T}U_k. \tag{8}$$

Matching D fixes the factor $A_{0:T} = e^{-D}$ on the initial weights but not the factors on later updates, and the updates themselves depend on the earlier trajectory. Schedules with the same D can therefore reach different parameters. The identity does not predict which schedule reaches lower loss.

F WARMUP AND DOWNSTREAM RESULTS

Warmup. Table 12 repeats the comparison with a linear warmup over the first 10% of training. When the decay window is the warmup window, early decay beats the best sampled constant at 405M and 1.4B but ties it at 162M. In the 3,200- and 8,100-update runs, all four tested windows that extend past warmup beat the best sampled constant, by 0.011 to 0.025 nats; Table 12 reports two representative settings.

Table 12: **Early decay under 10% learning-rate warmup.** Muon, ReLU^2. "Const." is the best sampled constant, with its coefficient in parentheses. The first three rows use a decay window equal to warmup; the final two extend it. The 162M rows are means over four matched seeds; the 405M and 1.4B rows are single runs. FineWeb-Edu at 162M and 405M, FineWeb100B at 1.4B. Differences are in nats.

Scale / updates	Const.	Early (λ, window)	vs. none	vs. const.
162M / 3,200	3.1944 (0.05)	3.1940 (0.6, 320)	-0.0094	-0.0004
405M / 8,100	2.8724 (0.02)	2.8648 (0.3, 810)	-0.0130	-0.0076
1.4B / 6,000	2.8965 (0.01)	2.8847 (0.4, 600)	-0.0129	-0.0118
162M / 3,200	3.1944 (0.05)	3.1730 (0.3, 960)	-0.0304	-0.0214
405M / 8,100	2.8724 (0.02)	2.8473 (0.3, 1,620)	-0.0305	-0.0251

With warmup and 16,000 updates at 162M (FineWeb100B), $\lambda = 0.15$ over the first 7,000 updates reaches 3.1455, which is 0.0329 nats below the best constant (0.05, 3.1784). It trails no decay by 0.0237 at update 8,000 and leads by 0.0373 at update 14,400, so warmup does not remove the delayed crossover seen without warmup.

Downstream evaluation. Table 13 reports zero-shot accuracy (task-normalized where applicable) for the 10% windows; the 44% checkpoint was not kept. The 405M entries average three seeds and use update 8,000 of 8,100. Mean accuracy differs by at most 0.11 points between early decay and constant 0.01 at either scale, and at 1.4B early decay is 0.65 points above no decay.

Table 13: Zero-shot accuracy (%). The 405M models are trained on FineWeb-Edu and the 1.4B models on FineWeb100B. HS: HellaSwag; ARC-E: ARC-Easy; WG: WinoGrande.

Scale	Schedule	HS	ARC-E	PIQA	LAMBADA	WG	Mean
405M	Constant 0.01	38.21	52.40	65.81	30.19	52.80	47.88
405M	Gram	38.67	52.83	65.83	30.10	52.46	47.98
405M	Early 0.3	38.72	52.30	66.21	30.65	51.85	47.95
1.4B	No decay	54.97	54.59	73.67	52.59	56.20	58.40
1.4B	Constant 0.01	55.78	54.12	72.52	53.31	58.96	58.94
1.4B	Early 0.2	56.92	52.65	73.83	54.55	57.30	59.05

G NORM-DYNAMICS CHECKS

Norm dynamics after the window. Write the update of a weight matrix as $W_{t+1} = W_t + \Delta W_t$, where ΔW_t includes any decay term, and let

$$a_t = \frac{\|\Delta W_t\|_F}{\|W_t\|_F}, \qquad c_t = \frac{\langle W_t, \Delta W_t \rangle_F}{\|W_t\|_F \, \|\Delta W_t\|_F}. \tag{9}$$

The relative step size of Equation 3 is $\alpha_t = a_t\sqrt{1 - c_t^2}$. Thus, a recurrence for a_t transfers to α_t only when c_t is small or changes similarly across the runs being compared. Expanding $\|W_t + \Delta W_t\|_F^2$ gives the exact identity

$$\|W_{t+1}\|_F^2 = \|W_t\|_F^2 \left(1 + 2c_t a_t + a_t^2\right). \tag{10}$$

If, in addition, consecutive Muon updates have equal Frobenius norm, which Muon's normalization makes approximately true at a constant learning rate without decay, dividing Equation 10 by $\|\Delta W_t\|_F^2$ gives

$$\frac{1}{a_{t+1}^2} - \frac{1}{a_t^2} = 1 + 2\kappa_t, \qquad \kappa_t = \frac{c_t}{a_t} = \frac{\langle W_t, \Delta W_t \rangle_F}{\|\Delta W_t\|_F^2}. \tag{11}$$

Without decay, $W_t = W_0 + \sum_{k<t} \Delta W_k$, so

$$\kappa_t = \frac{\langle W_0, \Delta W_t \rangle_F}{\|\Delta W_t\|_F^2} + \sum_{k<t} \frac{\langle \Delta W_k, \Delta W_t \rangle_F}{\|\Delta W_t\|_F^2}. \tag{12}$$

The second term measures how well the current update aligns with all earlier updates. The learning rate enters through the weight changes even though no separate η_t appears in this expression. These equations describe norms and relative steps; they do not predict loss.

Checks at 162M. We test these equations on the two 162M runs of Section 5 (FineWeb-Edu, 3,200 updates, seed 1234), which log α_t and c_t every 100 updates. Figure 8 shows updates up to 2,800, before the final learning-rate reduction. The two runs have similar κ_t and small $|c_t|$. The recurrence closely matches norm growth for four matrix families and overpredicts it for W_V and W_O. The gap $G_t = (\alpha_t^{\text{none}})^{-2} - (\alpha_t^{\text{early}})^{-2}$ changes much less than $1/\alpha_t^2$: from update 400 to 2,800, $1/\alpha_t^2$ grows about fourfold, while G_t varies by up to 54% (for W_V) and at update 2,800 is within 26% of its value at update 400 in every family. Holding G at its early value follows the observed convergence of the relative-step ratio (Figure 8). Over the measured interval, early decay therefore produces an approximately fixed norm deficit when expressed in units of the update norm. Later updates grow the squared norm at similar rates in both runs. As the norm grows, the deficit becomes a smaller fraction of it, and the relative steps converge slowly.

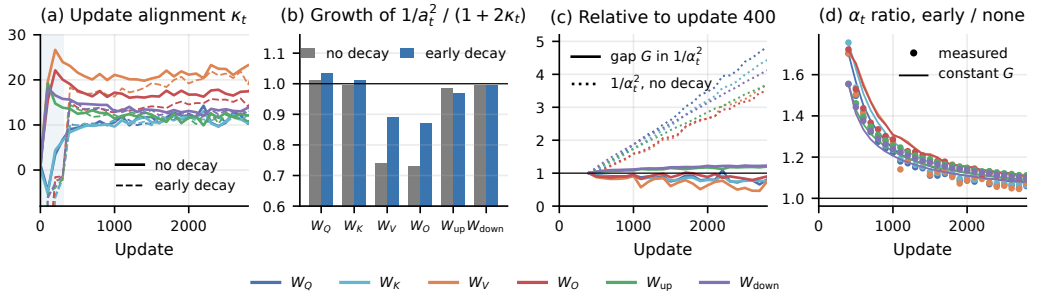


Figure 8: **Norm dynamics at 162M** (FineWeb-Edu, seed 1234): no decay versus $\lambda = 0.3$ for the first 320 updates (shaded). (a) κ_t of Equation 11. (b) Fitted slope of $1/a_t^2$ over updates 500–2,800 divided by the mean of $1 + 2\kappa_t$; 1 is exact agreement. (c) Gap G in $1/\alpha_t^2$ between the runs (solid) and $1/\alpha_t^2$ of the no-decay run (dotted), each divided by its value at update 400. (d) Ratio of relative steps; lines hold G at its update-400 value.

Early learning-rate boost. A learning-rate boost tests the recursion with the opposite perturbation: it raises the norms early, so later relative steps should be smaller. If the matrix learning rate is multiplied by r for the first e updates and κ_t is unchanged, $1/a_t^2$ afterwards exceeds its no-decay value by $(r^2-1)\sum_{k<e}(1+2\kappa_k)$. We ran $r=2$ for 320 updates, $r=4$ for 320 updates, and $r=2$ for 960 updates, without decay. The offset remains approximately constant after each boost. For W_{up} and W_{down}, it agrees with the prediction within 8%; deviations for the attention matrices are consistent with changes in κ_t or unequal update norms. The relative steps remain below the no-decay values, and all three runs end with higher loss.

Replaying relative steps at fixed norm. To isolate the relative-step trajectory, we remove norm changes. After each Muon step, a variant of Hyperball (Wen et al., 2026) rescales the weight change of each matrix to Frobenius norm $\eta_t\|W_0\|_F$ and projects the weights back to norm $\|W_0\|_F$. Weight norms then stay at their initial values, and η_t sets the relative step. We set η_t to the six-family mean of α_t logged in either the no-decay run or the $\lambda=0.3$ run, interpolated linearly between logged updates (Figure 5a). The replays use 162M, FineWeb-Edu, 3,200 updates, and no weight decay. Both trajectories come from seed 1234 and are reused for all three replay seeds. The early-decay trajectory beats the no-decay trajectory by 0.0221–0.0233 nats across the three replays, compared with 0.0171 nats between the Muon source runs (Figure 5b). Splitting the trajectories at update 320 attributes 0.0181 nats of the 0.0221-nat replay difference to the post-window steps. Replaying only the early-window trajectory before continuing with ordinary Muon changes final loss by just 0.0003 nats. With norms held fixed, the early-decay trajectory reproduces the ordering and approximate size of the gain, and most of the replay gain comes from the larger relative steps after the window. Figure 8 relates these larger steps to the reduced norms.

The Gram penalty raises the relative steps of the matrices it targets. The Gram runs also log α_t. We divide α_t in each Gram run by α_t in a reference with the same background decay and no penalty. Immediately after removal, the targeted families take 1.3–2.7 times the reference relative step; near the end of training, the ratio is 1.04–1.11. After removal, untargeted families stay within 5% of the reference when W_V, W_O, and W_{down} are all targeted. When only W_{down} is targeted (SwiGLU), W_V and W_{up} differ from the reference by up to 10% until about update 1,600. The Gram penalty therefore produces the same qualitative pattern as early decay—smaller norms and larger later relative steps—mainly in the matrices it targets.