T^3: TRAINING-TIME TESTING FOR DATA-CONSTRAINED PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

High-quality data for pretraining is limited and grows more slowly than compute, which makes it important to get the most out of a fixed corpus. Prior work improves generalization under fixed data with heavy weight decay, ensembling, distillation, and layer looping. We ask whether the training data can also serve as validation data for every update. We propose **T**raining-**t**ime **t**esting (T^3): it splits each batch into two halves, takes a temporary descent step on the first half, evaluates the gradient of the second half at the adapted weights, discards the temporary step, and updates with the mean of the two gradients. Added to strongly tuned data-constrained recipes, T^3 lowers validation loss by 0.012 to 0.015 nats per token at 300M, 600M, and 1.4B parameters. On SlowRun, adding the step alone improves the top one-hour script by 0.010 to 0.012 within its time limit, and scripts built on it set the leaderboard records on both the 15-minute and one-hour tracks. On both benchmarks the gain persists after ensembling. Probes show that the gain requires a descent step tested on other tokens; the cosine between the gradients of different examples does not increase, and flatness alone is insufficient. In a nonlinear model of repeated training, we prove that at a common training-loss target, cross-example testing learns more shared signal and less sample-specific noise than ordinary training, yielding lower population risk.

1 INTRODUCTION

High-quality text for language-model pretraining is limited and grows more slowly than the compute available to train on it (Villalobos et al., 2024). Once the corpus is fixed, a model can only see the same data again, and repetition helps for a few passes before its returns vanish (Muennighoff et al., 2023). Data-constrained pretraining has therefore become an optimization problem of its own: how low can held-out loss go when data is the limit? Two settings pose it concretely. Kim et al. (2025) study pretraining under infinite compute: for a fixed corpus and each model size, they tune the learning rate, weight decay, and number of passes jointly with coordinate descent. The SlowRun benchmark (qlabs-eng, 2026) fixes the data and a wall-clock budget and ranks training scripts by validation loss.

Progress in both settings has come from techniques that improve generalization rather than fitting. The strongest recipes regularize far more heavily than standard practice, with large weight decay and dropout; they ensemble independently trained models and distill ensembles back into single models; and they loop middle layers to spend more computation per token or average weights over training (Kim et al., 2025; qlabs-eng, 2026). These techniques act on the weights, the architecture, or the final prediction, but none of them tests the update itself. An ordinary gradient step fits the batch it is computed from, and nothing checks whether that fit carries over to other data. Held-out validation data serve this purpose in practice, but only once per run, to select hyperparameters or to stop training. We ask:

Q1: *Can the training data itself serve as validation data for every update?*

We introduce training-time testing (T^3, Figure 1): the held-out half of each batch tests a temporary step taken on the other half, and its gradient at the adapted weights enters the update. T^3 uses the same tokens, optimizer, and schedule as ordinary training, and the deployed model is static. We add it without retuning to the tuned recipes of Kim et al. (2025) at three model sizes (Section 3.1) and to the

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.

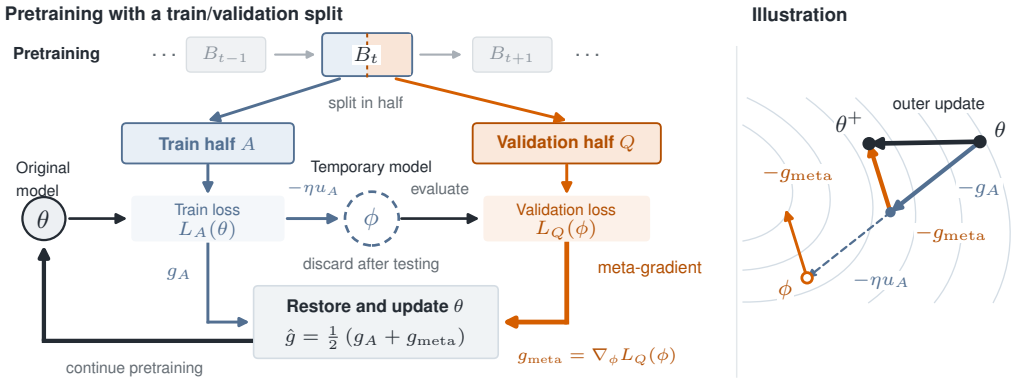


Figure 1: **Each pretraining batch supplies its own train/validation task.** T^3 splits batch B_t into a train half A and a validation half Q, takes a temporary step $\phi = \theta - \eta u_A$ on the plastic MLP matrices, evaluates the first-order meta-gradient $g_{\text{meta}} = \nabla_\phi L_Q(\phi)$ without differentiating through the step, restores θ, and gives the outer optimizer $\hat{g} = \frac{1}{2}[g_A + g_{\text{meta}}]$ with $g_A = \nabla L_A(\theta)$. The validation half comes from the current batch, the deployed model is static, and the right-hand geometry is schematic.

record scripts of both SlowRun tracks (Section 3.2). T^3 lowers validation loss on both benchmarks, and probability ensembles of independently trained models keep the gain, so T^3 adds to the strongest known use of extra compute under fixed data. It also beats a tuned sharpness-aware minimization (Foret et al., 2021) at about half its cost, and it helps in one pass over the data, although this gain diminishes as the data horizon grows.

Why does this help? Two explanations come to mind. T^3 might favor updates on which different examples agree, so that their gradients point in more similar directions (Proposition 1). Or it might make the model flatter, as sharpness-aware minimization does. This leads to:

Q2: *Why does testing each update on other training data help?*

On the tiny SlowRun track (Section 4), the gain requires a descent step tested on other tokens. The cosine between the gradients of the two batch halves does not increase. Flatness alone is also insufficient: an ascent step produces similarly flat models without improving validation loss.

To study what the update learns, we analyze a nonlinear model with shared signal and sample-specific noise (Section 5). We prove that, at a common training-loss target, cross-example testing learns more signal and less noise than ordinary training or self-testing, yielding lower population risk (Theorem 1).

2 Training-time testing

2.1 The update

Each T^3 step divides the current batch into a support half and a query half and temporarily adapts a selected set of weights. Let $\theta \in \mathbb{R}^d$ be the parameters and $L_S(\theta)$ the mean next-token loss on a set of sequences S. The current batch is split into two equal halves, the support half A and the query half Q, with support gradient $a = \nabla L_A(\theta)$. A fixed coordinate projection P selects the plastic weights, the matrices that the temporary step may move. The temporary direction is the projected support gradient normalized to unit length,

$$u_A = \frac{Pa}{\max\big(\|Pa\|_2,\, \epsilon\big)}, \qquad \phi_A = \theta - \eta\, u_A, \tag{1}$$

where $\epsilon = 10^{-12}$ keeps the division defined and gives $u_A = 0$ when $Pa = 0$. Whenever $\|Pa\|_2 \geq \epsilon$, the temporary weights ϕ_A lie at Euclidean distance η from θ, and we call η the radius.

The outer optimizer receives the mean of the support gradient and the query gradient evaluated at the temporary weights,

$$\hat{g} = \tfrac{1}{2}\big[\nabla L_A(\theta) + \nabla L_Q(\phi_A)\big], \tag{2}$$

which we call the T^3 gradient. The temporary step is undone before the optimizer runs. The optimizer therefore applies $\hat{g}$ at the original weights θ with its usual state, learning rate, and schedule, and $\hat{g}$ has the scale of an ordinary full-batch gradient. The projection P restricts only the temporary displacement. Every parameter receives both gradients in Equation (2), so adapting only MLP matrices also changes the gradient that the attention weights receive. We call the computation of the two half-batch gradients around a temporary step an episode; Algorithm 1 in Appendix A lists the complete step.

2.2 Objective and cost

An objective that adds the loss after a temporary step on independent data motivates this update. Let $L(\theta)$ be the expected loss under the data distribution, and treat A and Q as independent samples with $\mathbb{E}_Q \nabla L_Q(\theta) = \nabla L(\theta)$; the implementation draws them as disjoint halves of one shuffled batch. The objective

$$J(\theta) = \tfrac{1}{2}\, L(\theta) + \tfrac{1}{2}\, \mathbb{E}_A\big[L(\phi_A(\theta))\big] \tag{3}$$

adds to the ordinary loss the expected loss after one temporary step taken on another sample. Its exact gradient differentiates through the temporary step, whose Jacobian $I - \eta\, \partial u_A / \partial \theta$ involves the curvature of L_A. The T^3 gradient drops this Jacobian, as first-order model-agnostic meta-learning does (Finn et al., 2017), and needs only the two half-batch gradients. In general its expectation differs from ∇J already at first order in η (Appendix A).

Proposition 1 (Expansion of the T^3 objective)**.** *Let L be three times differentiable with bounded third derivative, let A and Q be independent with $\mathbb{E}_Q \nabla L_Q(\theta) = \nabla L(\theta)$, and let $\|P \nabla L_A(\theta)\|_2 \geq \epsilon$ almost surely. Then*

$$J(\theta) = L(\theta) - \frac{\eta}{2}\,\Lambda(\theta) + \frac{\eta^2}{4}\,\Gamma(\theta) + O(\eta^3), \tag{4}$$

where $\Lambda(\theta) = \mathbb{E}_{A,Q}\langle \nabla L_Q(\theta),\, u_A \rangle$ and $\Gamma(\theta) = \mathbb{E}_A\big[u_A^\top \nabla^2 L(\theta)\, u_A\big]$.

The first-order coefficient Λ is the expected projection of a query gradient onto the normalized support direction. The proof is a Taylor expansion of $L(\theta - \eta u_A)$ with $\|u_A\|_2 = 1$ (Appendix B). Where $\Lambda > 0$, the first-order term favors the descent step. An ascent step, like the perturbation of sharpness-aware minimization (Foret et al., 2021), reverses the sign of this term. A random direction that is independent of the data and symmetric about zero has zero mean and removes the term. Section 4.2 tests these alternatives. The expansion describes J only, and the T^3 gradient need not follow it, so the proposition motivates these controls without predicting their outcome. Appendix A derives the expected T^3 gradient, gives a quadratic example with a raw inner step, and states the relation to Reptile (Nichol et al., 2018).

Evaluating the second half-batch gradient at the temporary weights leaves the number of training tokens and gradient evaluations unchanged, because an ordinary step that accumulates its batch in two halves also computes two half-batch gradients. The added work is the norm of the projected support gradient, the temporary step, and the restore. In the fixed-data experiments the support gradient is normalized after averaging it over the support half, and the median step time is 1.06× the baseline's at 300M and 1.4B and 1.09× at 600M (Appendix D). Throughout the main text T^3 runs an episode at every optimizer step; Appendix A defines the other variants we tested.

3 Experiments

We evaluate T^3 on two testbeds for data-constrained pretraining, both with repeated passes over a fixed corpus.

Pre-training under infinite compute. We use the certified AdamW recipes of Kim et al. (2025). The corpus is a fixed pool of 196.6M DCLM tokens (Li et al., 2024), repeated in shuffled order for 16 passes at 300M parameters and 8 passes at 600M and 1.4B; validation uses 1,000 held-out documents (Appendix H). Each T^3 run is paired with a baseline run that shares its model seed, data

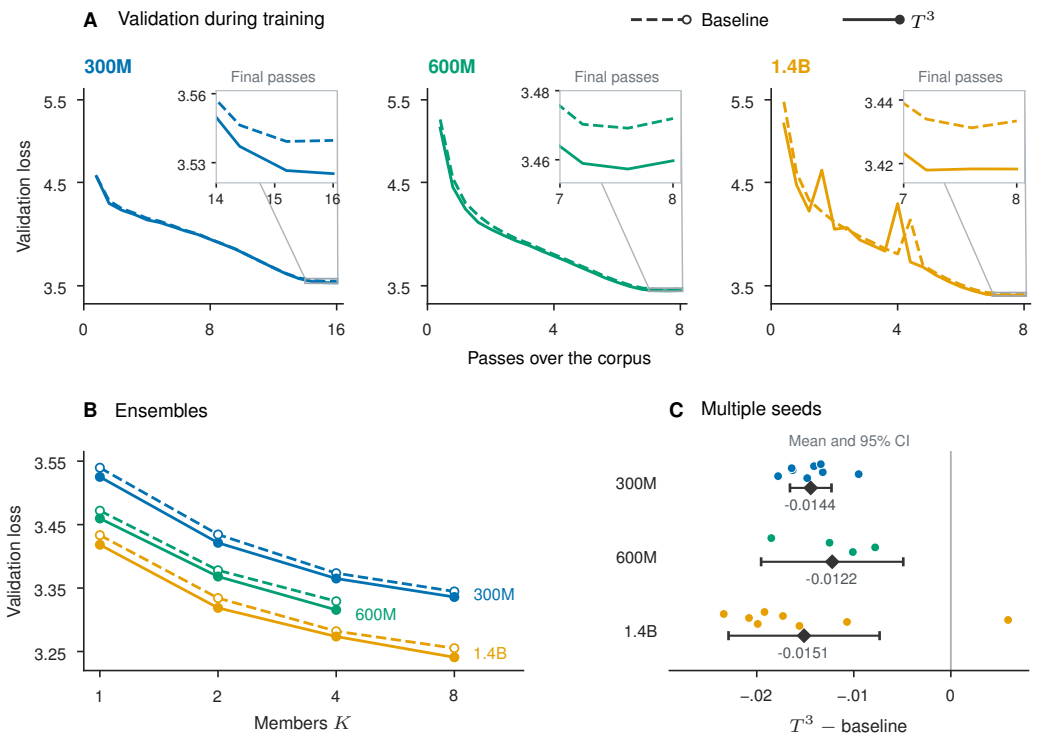


Figure 2: **T^3 improves single-model and ensemble validation loss in data-constrained pretraining.** Results on the fixed DCLM corpus with certified AdamW recipes, in nats per token. Colors denote model size; dashed and solid lines denote the baseline and T^3. **(A)** Validation loss averaged over paired seeds, with a common logarithmic loss scale and initialization omitted. Connected insets magnify the boxed final passes on linear loss axes with separate vertical ranges. **(B)** Loss after averaging the predictions of K independently trained models. Open and filled markers denote the baseline and T^3. At $K = 1$, we show the mean member loss of the largest ensemble; ensembles at different K share members. **(C)** Final loss differences for individual seed pairs, $L_{\mathrm{T}^3} - L_{\mathrm{baseline}}$; negative favors T^3. Black diamonds and bars show the paired mean and its pointwise 95% Student-t confidence interval ($n = 8, 4, 8$ for 300M, 600M, and 1.4B), assuming independent pairs and approximately normal paired differences. The tuning seed is excluded.

order, schedule, and evaluation, so a paired difference isolates the change to the update. T^3 adapts the MLP matrices of the second half of the layers with radius 0.3. We chose the radius on one seed at 300M and report other seeds only; Appendix E collects the tuning sweeps and further replications, including a 150M model.

SlowRun. Both SlowRun tracks train on the same 100M FineWeb tokens and score the validation loss of the final prediction. The one-hour track trains a 1.4B-parameter model for eleven passes with Muon, and its record script predicts with the probability average of its last three checkpoints; the tiny track allows fifteen minutes and a model of about 300M parameters. We start from the record scripts, the best published scripts of both tracks at the time, and pair every run with a baseline on the same seed. The *episode-only* runs add a T^3 episode at every step of the unchanged one-hour script and isolate T^3; the *submitted* scripts also adjust the radius schedule and two regularizers (Appendix M).

3.1 T^3 Improves Tuned Recipes and Their Ensembles

Single models. T^3 lowers the final loss at every size (Figure 2C): by 0.0144 ± 0.0026 nats per token over eight seeds at 300M, 0.0122 ± 0.0046 over four at 600M, and 0.0151 ± 0.0093 over eight at 1.4B, improving every pair but one at 1.4B. Figure 2A shows the corresponding validation-loss trajectories during training. Since the recipe's learning rate, weight decay, and number of passes are

already tuned, the gain is not a substitute for tuning; rerunning the neighbouring learning rates and weight decays with T^3 leaves the certified point best (Appendix E).

Ensembles. Ensembling independently trained models is the strongest known way to spend more compute under fixed data (Kim et al., 2025), so we ask whether it absorbs the gain. An ensemble of K members averages their predicted next-token probabilities, and its loss decomposes exactly as

$$L_{\text{ens}} = L_{\text{member}} - D, \qquad D = \left\langle \log \bar{p}(y) - \frac{1}{K} \sum_{j=1}^{K} \log p_j(y) \right\rangle \geq 0, \tag{5}$$

where L_{member} is the mean member loss, $\bar{p}$ the averaged prediction, and D the disagreement between members. T^3 ensembles stay below baseline ensembles of the same seeds at every size and member count (Figure 2B), by 0.0087 at 300M and 0.0144 at 1.4B with eight members. Their disagreement is close to the baseline's (Table 13), so T^3 does not buy its single-model gain with less diverse members.

Sharpness-aware minimization and a single pass. Sharpness-aware minimization (Foret et al., 2021) also computes the gradient after perturbing the weights, but along the ascent direction and on the same tokens. Tuned over seven radii on the 300M recipe, it improves on the baseline by less than a third as much as T^3 and nearly doubles the step time, while T^3 adds 6% (Appendix E). T^3 also helps when every token is seen once: in a single pass over the corpus with a Muon recipe it lowers the loss by 0.025 at equal updates and by 0.011 at equal training time (Appendix N). This one-pass gain diminishes as the data horizon grows, and it is within 0.003 of zero over a 1.7B-token pass of the modded-nanogpt optimization track (Appendix O). The gain therefore belongs most clearly to recipes that revisit limited data, consistent with the early lead in Figure 2A.

3.2 T^3 Sets Records Under a Fixed Training Time

Results. Adding only the episode improves the one-hour score on each of three seeds, by 0.0111 ± 0.0011 nats per token, within the time limit (mean $\pm$ sample standard deviation of paired differences; Figure 3A,D; scores in Table 24). The retuned T^3 recipes improve every seed of both tracks, by about 0.020 on the one-hour track and 0.012 on the tiny track, and they became the records of both tracks when submitted (Figure 3D). As in the first testbed, the lead is largest after the first pass and stays positive at the end of every pass (Figure 3B; per-seed differences in Figure 13A). Ensembles of eight tiny-track models keep the full member gain (Figure 3C), with unchanged disagreement (Table 25). Adding only T^3 increases mean benchmark training time by just 0.4 minutes (0.7%), from 59.03 to 59.43 minutes, over the three paired one-hour runs (Figure 3A).

4 What the Temporary Step Does

Proposition 1 suggests that T^3 may reward agreement between the gradients of different examples. We test this and other accounts with probes on the tiny SlowRun track, where a run takes fifteen minutes, using two seeds per probe and the EMA endpoint that the recipe evaluates (Appendix L lists every endpoint); Section 5 then analyzes why a test on other examples helps.

4.1 Agreement Between Batch Halves Does Not Change

If T^3 worked by making gradients agree, the cosine between the gradients of the two halves of a batch would be higher along its trajectory. We recorded this prediction before measuring, and it fails (Figure 4A). Measured every ten steps over all Muon matrices, the cosine under T^3 is within 0.005 of the record script's on both seeds, and in both methods it falls from about 0.6 to 0.1 over training. This cosine is a sample estimate and does not exclude an agreement effect in the population, but the simple form of the account is not supported.

4.2 The Step Must Descend and Be Tested on Other Tokens

We changed how the temporary step is taken and where it is tested, keeping the rest of T^3 fixed: a step of the same length in a random or an ascent direction; a step of zero length, where the update reduces to the full-batch gradient; a split of each sequence into its first and second half, so that the

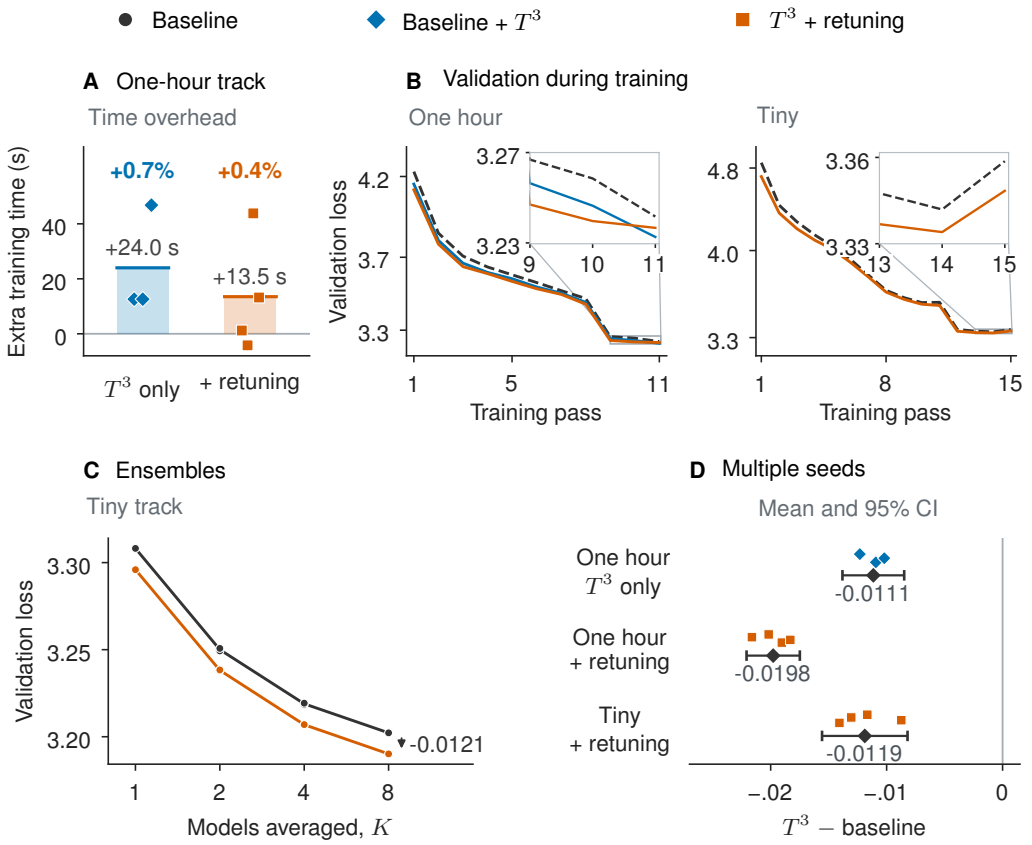


Figure 3: **T^3 improves SlowRun scores with small training-time overhead.** **(A)** Additional one-hour training time relative to the same-seed baseline. Bars show paired means, markers show individual runs, and percentages are relative to the mean baseline time; all times follow the benchmark training timer ($n = 3$ for T^3 only and $n = 4$ with retuning). **(B)** Validation loss at the current weights after each training pass, averaged over seeds 42–44 (one hour) or 42–45 (tiny). Overview axes are logarithmic; connected insets show the final three passes on linear axes. **(C)** Tiny-track probability ensembles over seeds 42–49. At $K = 1$, losses are mean member losses; two disjoint member sets are shown at $K = 2, 4$, with lines connecting their means. **(D)** Final score differences from the same-seed baseline; negative favors T^3. Black diamonds and bars show paired means and pointwise 95% Student-t confidence intervals ($n = 3, 4, 4$ from top to bottom), assuming independent pairs and approximately normal paired differences. Final scores include the scripts' checkpoint averaging, unlike the current-weight losses in B. All losses and scores are in nats per token; blue adds only T^3, while orange also retunes the recipe.

step is tested on later positions of the same documents; and steps tested on their own tokens, with the same or a fresh dropout mask, or taken and tested on the full batch.

Results. On the tiny track (Figure 4B), T^3 gains 0.0115 and 0.0080 on the two seeds and the within-sequence split 0.0121 and 0.0105. Testing on the step's own tokens loses 0.015 to 0.017 with the same dropout mask and 0.012 to 0.016 with a fresh one; the full-batch step gains 0.0013 and loses 0.0022. Random and ascent directions stay within 0.0044 of the record script, and keeping only the part of the T^3 correction parallel or perpendicular to the ordinary gradient does not reproduce the gain on both seeds. In the 300M fixed-data recipe the descent step gains 0.025 while random and ascent steps gain less than 0.003 (Table 21), and doubling the computation with a same-batch step recovers about half of the gain on one seed and loses to the baseline on another. With the radius set to zero, the one-hour SlowRun recipe loses its improvement. The gain therefore requires a descent step tested on held-out tokens, which may come from the same documents.

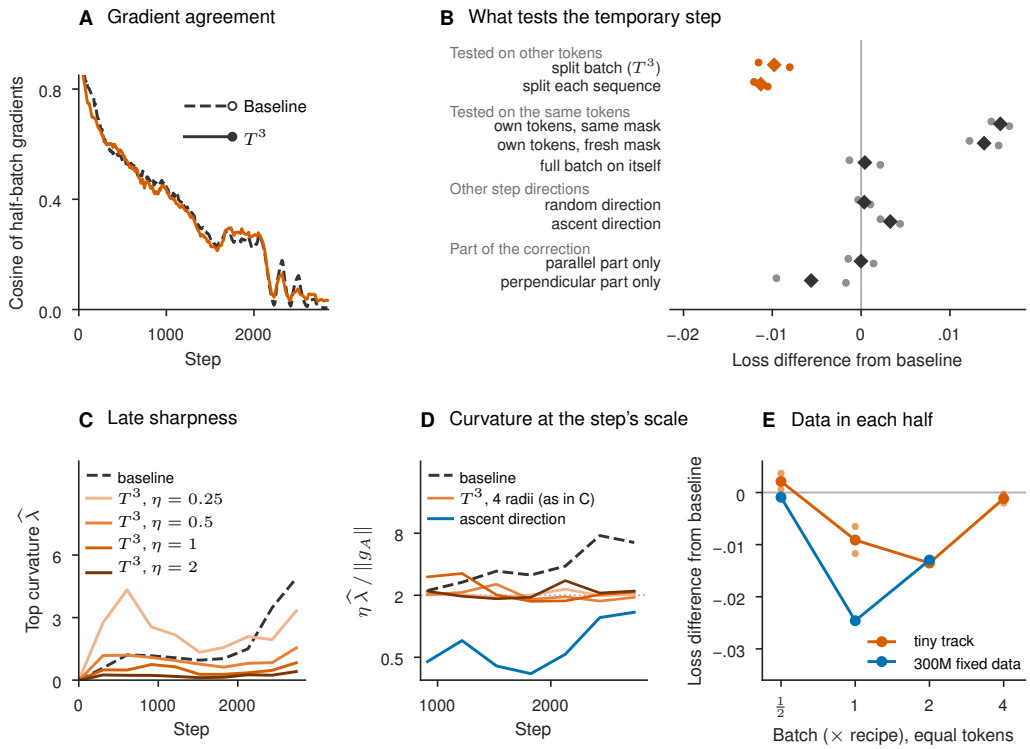


Figure 4: **What the temporary step does.** Tiny SlowRun track, two seeds; lines are seed means. Loss differences are the run minus the same-seed baseline (the record script) at the EMA endpoint, in nats per token, so negative favors the run. (A) Cosine between the gradients of the two batch halves on the Muon matrices, smoothed. (B) Effect of each variant of the temporary step; small points are seeds and diamonds their mean. (C) Finite-difference estimate of the top curvature of the training gradient on the MLP matrices, for the baseline and T^3 at four radii η. (D) This curvature times η over the mean support-gradient norm; the baseline and the ascent control are evaluated at $\eta = 0.5$, and the dotted line marks 2. (E) Effect of T^3 against the batch at equal tokens, on the tiny track (points are seeds) and in the 300M fixed-data setting.

4.3 Late Sharpness Falls to the Scale of the Temporary Step

Gradient descent with step size s settles where the top curvature of the loss is close to $2/s$ (Cohen et al., 2021). The normalized temporary step is a gradient step with step size $\eta/\|g_A\|$, where g_A is the support gradient on the plastic matrices, so this scale corresponds to $\eta\lambda/\|g_A\| = 2$. We estimate the top curvature $\widehat{\lambda}$ of the plastic matrices by power iteration on finite-difference products of the training gradient (Appendix L).

Results. As the learning rate decays the record script sharpens, while T^3 ends two thirds flatter, and more so at larger radii (Figure 4C). The ratio $\eta\widehat{\lambda}/\|g_A\|$ settles near 2 in every T^3 run, four radii on two seeds, with medians between 1.9 and 2.1 from step 900 on, while the record script's rises to 6 to 8 (Figure 4D). T^3 thus holds the plastic matrices at the curvature its temporary step can descend. Flatness alone does not produce the gain: the ascent direction ends as flat and gains nothing, and the gain saturates beyond radius 1 while the curvature keeps falling.

4.4 The Gain Needs Data in Each Half of the Batch

A step that adapts on one half of a batch and tests on the other might act like training with half the batch. We varied the batch at a fixed number of tokens, on the tiny track and in the 300M fixed-data recipe.

Results. Halving the batch removes the gain in both settings: T^3 ends 0.0006 and 0.0037 above the record script on the tiny track and 0.0009 below the baseline at 300M, even though the record script itself improves at half the batch (Figure 4E). Twice the batch keeps it (0.0138 and 0.0132 on the tiny track, 0.0130 at 300M), and four times the batch, with a quarter of the steps, loses it again (0.0003 and 0.0020). T^3 is therefore not training with half the batch: it needs enough data in each half to produce a useful test, and enough steps to use it.

4.5 T^3 Models Under Test-Time Adaptation

Test-time training takes temporary steps at inference (Sun et al., 2020; Tandon et al., 2025). We took one or two normalized steps along the loss of the first half of each validation sequence and scored the second half, with the prefix in the context and, separately, with it removed so that it reaches the suffix only through the step (Appendix O.2). Models trained with T^3 gain more from these steps and tolerate larger ones.

Results. With the prefix in context, one step of radius 0.25 lowers the suffix loss by 0.0048 and 0.0056 for T^3 models and by 0.0047 and 0.0046 for the record script's. At radius 0.5 the record script's models lose 0.0006 to 0.0008, while T^3 models still gain up to 0.0013. With the prefix removed, one step of radius 0.5 recovers 0.0153 and 0.0163 for T^3 against 0.0148 for the record script, while a step along another sequence's prefix recovers at most 0.001, so the recovered information is specific to the document. Two steps of radius 2 raise the suffix loss by 0.28 to 0.30 for T^3 models and by 0.32 to 0.35 for the record script's, consistent with their lower late sharpness.

5 How Cross-Example Testing Can Favor Shared Features

We study whether cross-example testing favors shared signal over sample-specific memorization in a nonlinear model. Train–validation splitting can shape representations in meta-learning (Saunshi et al., 2021); here we study repeated training on one fixed dataset.

Shared signal and sample-specific noise. Fix an unknown orthonormal basis $u_0, \dots, u_M$ of $\mathbb{R}^{M+1}$. Each example has a label $y \in \{-1, +1\}$ and two input patches: a shared signal yu_0 and a nuisance patch vu_J, presented in random order. The signs y, v and index $J \in \{1, \dots, M\}$ are independent and uniform. We train one shared nonlinear neuron,

$$f_w(x_1, x_2) = h(w^\top x_1) + h(w^\top x_2), \qquad h(z) = z + z^3. \tag{6}$$

Draw $S = \{(x_i, y_i)\}_{i=1}^n$ iid once and reuse it, with empirical loss $L_S(w) = n^{-1} \sum_i \ell_i(w)$ and $\ell_i(w) = \frac{1}{2}(y_i - f_w(x_i))^2$. When their nuisance indices are distinct, fitting u_0 helps every example, whereas fitting a nuisance direction memorizes one example's random label association. Write $\kappa(w) = h(w^\top u_0)$ for the signal output and $N(w) = M^{-1} \sum_{j=1}^M h(w^\top u_j)^2$ for the nuisance output energy. Independence of the signs on a fresh example gives the exact population risk

$$R(w) = \tfrac{1}{2}\big[(1 - \kappa(w))^2 + N(w)\big]. \tag{7}$$

Thus nuisance fitting can reduce training loss while adding to test loss.

Training rules. Write $g_i = \nabla \ell_i$ and $\phi = w - \eta g_i(w) / \max\{\|g_i(w)\|_2, \epsilon\}$. Ordinary training, cross-example testing (*split*), and same-example testing (*self*) apply SGD to the averages of $g_i(w)$ with $g_j(w)$, $g_j(\phi)$, and $g_i(\phi)$, respectively. The temporary parameter is discarded. All coordinates are trainable. Each rule repeats the same fixed cycle of all ordered pairs $i \neq j$, using a constant outer rate α_D for rule D.

Theorem 1 (Feature allocation and generalization at matched fit)**.** *Fix $n \geq 2$, $M \geq n$, $\rho \in (0, 1)$, $r_* \in (0, 1/2]$, and $0 < \eta \leq r_*/32$. Set $\epsilon = r_*/8$. There are bounds $\alpha_0(n, r_*, \eta) > 0$ and $\sigma_0(n, r_*, \eta, \rho) > 0$ such that the following holds for every choice of $0 < \alpha_D \leq \alpha_0$ and $0 < \sigma \leq \sigma_0$ made before sampling. Run the three rules from a common $w_0 \sim \mathcal{N}(0, \sigma^2 I_{M+1})$, independent of S, and define*

$$\tau_D = \inf\{k \geq 0 : L_S(w_{D,k}) \leq r_*^2/2\}, \qquad \widehat{w}_D = w_{D,\tau_D}.$$

With probability at least $1-\rho-n(n-1)/(2M)$ *over* S, w_0*, every* τ_D *is finite and, writing* $\kappa_D = \kappa(\widehat{w}_D)$*,* $N_D = N(\widehat{w}_D)$*, and* $R_D = R(\widehat{w}_D)$*,*

$$\kappa_{\text{split}} > \kappa_{\text{self}} > \kappa_{\text{ordinary}}, \qquad N_{\text{split}} < N_{\text{self}} < N_{\text{ordinary}},$$
$$\min\{R_{\text{self}} - R_{\text{split}},\, R_{\text{ordinary}} - R_{\text{self}}\} \geq c\,\frac{r_*^5\eta}{n^2}, \tag{8}$$

where $c > 0$ *is universal. At each checkpoint,* $r_* - C\alpha_D \leq \sqrt{2L_S(\widehat{w}_D)} \leq r_*$ *for a universal* $C > 0$*.*

Appendix C gives explicit constants and the proof.

Why this improves generalization. Under the cycle-averaged dynamics from zero, every training margin equals $h(a) + h(b)$, where $a = w^\top u_0$ is the signal coordinate and $b = y_i v_i w^\top u_{J_i}$ the common nuisance coordinate oriented toward the label. Before reaching the target, the averaged dynamics at a common state $a > b > 0$ satisfy $(da/db)_{\text{split}} > (da/db)_{\text{self}} > (da/db)_{\text{ordinary}}$. Thus split learns more signal per unit of nuisance. At the same training residual, it has larger a and smaller b, lowering both terms in (7). The proof controls the error from finite updates and random initialization to preserve this ordering.

6 Related work

Pretraining with limited data. Muennighoff et al. (2023) characterize the diminishing value of repeated tokens and develop scaling laws that account for data reuse; Kim et al. (2025) and SlowRun (qlabs-eng, 2026) define the two testbeds we use. Optimizer studies emphasize that comparisons depend on hyperparameter tuning and on the training budget at which models are evaluated (Wen et al., 2025).

Meta-learning and adaptation. Gradient-based meta-learning trains parameters using performance after adaptation. MAML learns an initialization that can adapt to a new task from a small support set (Finn et al., 2017). First-order MAML omits differentiation through the adaptation steps, while Reptile moves the initialization toward the weights reached by an inner optimization loop (Nichol et al., 2018). These ideas also support training models for generalization without test-time adaptation: MLDG tests updates on held-out source domains (Li et al., 2018), and Fish uses a Reptile-style update to encourage agreement between domain gradients (Shi et al., 2022). In continual learning, MER and La-MAML combine meta-learning with replay to reduce interference (Riemer et al., 2018; Gupta et al., 2020); Abbes et al. (2025) study replay with Reptile in continual language-model pretraining. A separate line adapts models on evaluation inputs: test-time training uses self-supervision to address distribution shift (Sun et al., 2020), and Tandon et al. (2025) meta-train language models to learn from long contexts through next-token updates at test time.

Temporary parameter updates in optimization. Optimization methods also use gradients evaluated away from the current parameters. Extragradient evaluates a gradient after a temporary step before updating the original parameters (Korpelevich, 1976), and extrapolated SGD applies this principle to large-batch neural-network training (Lin et al., 2020). Lookahead interpolates between the original weights and those reached after several inner optimizer steps (Zhang et al., 2019). Sharpness-aware minimization (SAM) instead evaluates gradients after an ascent perturbation to minimize loss in a neighborhood of the parameters (Foret et al., 2021); its generalization effects depend in part on the stochastic implementation (Andriushchenko & Flammarion, 2022).

7 Conclusion

Testing a temporary descent step on the other half of each batch improves the static model in tuned data-constrained recipes, in their ensembles, and within fixed training times. The gain requires a descent step tested on other tokens with enough data in each half. In a nonlinear model of repeated training, such cross-example testing provably shifts learning from sample-specific noise toward the shared signal at a common training loss.

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

The supplementary material contains an anonymized, self-contained archive with the code, run definitions, and figure data for every result in the paper. It includes a minimal PyTorch implementation of the T^3 step with a test; the JAX/Levanter trainer with pinned dependencies and the resolved configuration of every fixed-data run; the SlowRun scripts as submitted, together with the research trainer, the probe and test-time-step scripts, and unified diffs against the pinned benchmark commit; and, for the one-pass experiments, the trainer with every case definition and a patch to the pinned nanochat commit. One script rebuilds every data figure and generated table from the included per-run records, and the rebuilt figures are pixel-identical to those in the paper. Data preparation is scripted against public corpora pinned by revision and SHA-256, and each directory states the run time of its experiments.

REFERENCES

Istabrak Abbes, Gopeshh Subbaraj, Matthew Riemer, Nizar Islah, Benjamin Therien, Tsuguchika Tabaru, Hiroaki Kingetsu, Sarath Chandar, and Irina Rish. Revisiting replay and gradient alignment for continual pre-training of large language models. *arXiv preprint arXiv:2508.01908*, 2025.

Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning (ICML)*, 2022. arXiv:2206.06232.

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

Abhimanyu Dubey et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, 2017.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations (ICLR)*, 2021. arXiv:2010.01412.

Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-MAML: Look-ahead meta learning for continual learning. *arXiv preprint arXiv:2007.13904*, 2020. arXiv comments: "Accepted for Oral Presentation at NeurIPS 2020".

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

Keller Jordan et al. modded-nanogpt: Speedrunning the NanoGPT baseline. https://github.com/KellerJordan/modded-nanogpt, 2024b.

Konwoo Kim, Suhas Kotha, Percy Liang, and Tatsunori Hashimoto. Pre-training under infinite compute. *arXiv preprint arXiv:2509.14786*, 2025.

G. M. Korpelevich. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody*, 12:747–756, 1976.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11596. URL https://ojs.aaai.org/index.php/AAAI/article/view/11596.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. DataComp-LM: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*, 2024.

Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Extrapolation for large-batch training in deep learning. In *International Conference on Machine Learning (ICML)*, 2020. arXiv:2006.05720.

Jinlong Liu, Yunzhi Bai, Guoqing Jiang, Ting Chen, and Huayan Wang. Understanding why neural networks generalize well through GSNR of parameters. In *International Conference on Learning Representations (ICLR)*, 2020. arXiv:2001.07384.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. In *Advances in Neural Information Processing Systems*, 2023.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

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, 2024.

qlabs-eng. Slowrun: a fixed-data, fixed-time language-model pretraining benchmark. https://github.com/qlabs-eng/slowrun, 2026.

Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018. arXiv comments: "ICLR 2019".

Nikunj Saunshi, Arushi Gupta, and Wei Hu. A representation learning perspective on the importance of train-validation splitting in meta-learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9333–9343, 2021. URL https://proceedings.mlr.press/v139/saunshi21a.html.

Yuge Shi, Jeffrey Seely, Philip H. S. Torr, N. Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vDwBW49HmO.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248, 2020.

Arnuv Tandon, Karan Dalal, Xinhao Li, Daniel Koceja, Marcel Rød, Sam Buchanan, Xiaolong Wang, Jure Leskovec, Sanmi Koyejo, Tatsunori Hashimoto, Carlos Guestrin, Jed McCaleb, Yejin Choi, and Yu Sun. End-to-end test-time training for long context, 2025.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? limits of LLM scaling based on human-generated data. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Kaiyue Wen et al. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025.

Michael R. Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

OVERVIEW OF THE APPENDICES

The appendices give the complete accounts behind the main text. Appendix A defines every variant of the update and analyzes the objective. Appendix B gives the proofs, and Appendix D reports costs and token counts. Appendix C gives the model, theorem, and complete proof for the feature-learning comparison in Section 5. Appendix E collects the tuning sweeps and replications, Appendix F the search grid of every recipe, and Appendix G the settings without a gain. Appendices H to K report the fixed-data experiments in full, including every comparison on the tuning seed and across sizes, the sharpness-aware minimization sweep, the ensemble members and evaluation, and the Muon comparisons with matched controls and external likelihood. Appendix L collects the mechanism probes. Appendices M and M.5 give the complete SlowRun results and the zero-displacement control. Appendix N reports one-pass compute experiments and test-time training of T^3 models.

A DETAILED UPDATE DEFINITIONS AND OBJECTIVE ANALYSIS

Algorithm 1 One optimizer step of T^3.

Require: weights θ; the current batch split into halves A and Q; plastic projection P; radius η; outer optimizer

1: $a \leftarrow \nabla L_A(\theta)$ (support gradient)
2: $u_A \leftarrow Pa/\max(\|Pa\|_2, \epsilon)$ (normalized over the plastic weights)
3: $\theta \leftarrow \theta - \eta\, u_A$ (temporary step)
4: $q \leftarrow \nabla L_Q(\theta)$ (query gradient at the temporary weights)
5: $\theta \leftarrow \theta + \eta\, u_A$ (restore the original weights)
6: apply the outer optimizer to θ with gradient $\frac{1}{2}(a+q)$ (state and schedules as usual)

T^3 evaluates a query half-batch after a temporary update on a support half-batch and averages their gradients for an outer update at the original weights.

A.1 THE OBJECTIVE AND ITS TWO GRADIENT ESTIMATORS

Write $L(\theta) = \mathbb{E}_x\, \ell(\theta; x)$ for the population loss of a model with parameters $\theta \in \mathbb{R}^d$ and $L_S(\theta) = \frac{1}{|S|}\sum_{x\in S} \ell(\theta; x)$ for its loss on a sample S. For the objective analysis, let the adaptation batch A and test batch Q be independent samples. The implementation uses disjoint halves of the current minibatch from the baseline's data sampler; independence is an idealization of the shuffled, repeated-data runs. The adaptation direction is the normalized gradient of the adaptation half,

$$u_A(\theta) = \frac{\nabla L_A(\theta)}{\|\nabla L_A(\theta)\|}, \qquad \phi_A(\theta) = \theta - \eta\, u_A(\theta), \tag{9}$$

with $u_A = 0$ when the gradient vanishes. For a nonzero gradient, $\phi_A(\theta)$ is the adapted parameter after one temporary step of norm η. This equation uses the full parameter gradient for exposition. In the record runs, a fixed coordinate projection P restricts the direction to the plastic matrices, giving $u_A = P\nabla L_A/\|P\nabla L_A\|$ with the same zero convention; raw and preconditioned directions are also tested. The JAX implementation floors this denominator at 10^{-12}; the unit-direction identities below describe gradients above that floor. The objective J of Equation (3) measures the loss that the adapted parameter incurs on the held-out half, while retaining the ordinary training loss. The second term is the expected validation loss after one step of adaptation. At training time the model is tested, on data it did not adapt on, in the state it would be in after a test-time update, and that test loss enters J. The single-sample estimator of J is $\hat{J} = \frac{1}{2}L_A(\theta) + \frac{1}{2}L_Q(\phi_A(\theta))$. Since Q is independent of A, $\mathbb{E}_Q \nabla L_Q(\phi_A(\theta)) = \nabla L(\phi_A(\theta))$. With regularity permitting differentiation through the expectations, $\hat{J}$ and its exact gradient are unbiased for J and ∇J.

By the chain rule the gradient of the estimator is

$$\nabla \hat{J} = \tfrac{1}{2}\nabla L_A(\theta) + \tfrac{1}{2}\Big(\frac{\partial \phi_A}{\partial \theta}\Big)^{\top} \nabla L_Q(\phi_A(\theta)), \qquad \frac{\partial \phi_A}{\partial \theta} = I - \eta\, \nabla u_A(\theta). \tag{10}$$

This is the meta-gradient of model-agnostic meta-learning (Finn et al., 2017) with A as the support set and Q as the query set, plus the retained support gradient. The exact estimator differentiates

through the adaptation map, which requires a second-order product with the curvature of L_A. The first-order estimator drops ∇u_A,

$$\hat{g}_{\mathrm{FO}} = \tfrac{1}{2}\nabla L_A(\theta) + \tfrac{1}{2}\nabla L_Q\big(\theta - \eta\, u_A(\theta)\big), \tag{11}$$

and needs the gradient of Q at the adapted point in place of the gradient of Q at θ, so with split pairing (the two halves of the current batch, Section A.2) it computes the same two half-batch gradients as a baseline step that accumulates its batch in two halves. For the full-gradient normalized direction of Equation (9), $\nabla u_A = \|\nabla L_A\|^{-1}(I - u_A u_A^\top)H_A$ with $H_A = \nabla^2 L_A(\theta)$, so the two estimators differ by

$$\nabla \hat{J} - \hat{g}_{\mathrm{FO}} = -\frac{\eta}{2\|\nabla L_A\|}\, H_A\, (I - u_A u_A^\top)\, \nabla L_Q(\phi_A(\theta)), \tag{12}$$

the curvature on A applied to the part of the test gradient orthogonal to the adaptation direction. For a projected direction, the corresponding Jacobian is $Du_A = (P - u_A u_A^\top)H_A/\|P\nabla L_A\|$ whenever the denominator is nonzero. The difference vanishes when the test gradient at the adapted point is parallel to the adaptation step and is suppressed while gradients are large relative to curvature. Thus the first-order estimator is generally biased for ∇J, already at order η. With raw full-parameter gradients and equal inner learning rates, its sum of support and query gradients is proportional to the displacement of a two-step Reptile loop (Nichol et al., 2018). This equivalence does not generally extend to our normalized inner step, because the outer estimator retains raw gradients.

All experiments run Algorithm 1. On the SlowRun tracks, each batch is partitioned into groups, and each group is split into a support half and a query half. Each support gradient is normalized separately over the plastic matrices; the resulting episode gradients are then averaged. Averaging support gradients over the batch before normalization was also tested (Appendix M.3). The data, number of passes, schedules, optimizer state initialization and update rules, and evaluation are identical to the baseline's, and the gradient handed to the optimizer has the scale of the ordinary full-batch mean. The added work is the norm, the temporary step, and the restore, about 12 ms of a 1.5 to 1.9 s step in the SlowRun scripts. The JAX data-constrained implementation constructs the direction from the support gradient averaged over the full batch; its historical AdamW runs cost about 6 percent more per step at 300M and 1.4B. A plastic mask restricts only the temporary parameter change. All parameters still receive their support and adapted-query outer gradients. Thus adapting only MLP weights can also change the outer gradient on attention weights.

A.2 The design choices the experiments vary

The variants studied below differ in the pairing of the two halves, in the inner step, and in the episode period. Split pairing uses the two halves of the current batch, so an episode (lines 1 to 5 of one execution of Algorithm 1) consumes exactly the batch the baseline would have consumed and costs 1.06× the baseline step time at 300M and 1.4B in our implementation (1.09× at 600M and 1.16× at 150M; Appendix D gives the step times). Consecutive pairing adapts on the whole current batch and tests on the whole next batch, which the next optimizer step then reuses; an episode then costs a full extra gradient, twice the baseline step time when it is run at every step.

The inner step is the temporary move. The normalized step of Equation (9) moves a fixed Euclidean distance η along the gradient restricted to a plastic subset of the parameters, the matrices the step is allowed to change (all matrix parameters, the multilayer perceptron (MLP) matrices, or the MLP matrices of the second half of the network in the different experiments); on the one-hour SlowRun track η follows the learning-rate schedule with a peak of 0.5, and elsewhere it is a constant. A raw step uses the unnormalized gradient, and a preconditioned step divides the gradient by the outer optimizer's second-moment estimate's square root plus epsilon, using its own inner learning rate. Inner weight decay contributes a parameter-proportional term to the temporary direction; its placement depends on the inner rule, and it is discarded when the temporary move is restored. The every-step design named throughout the paper is split pairing with a normalized step at every optimizer step, applied to each baseline recipe without retuning (a constant step norm of 0.3 on the second-half MLP matrices in the data-constrained runs, the scheduled norm above in the record scripts); it consumes exactly the baseline's tokens and gradient evaluations.

The episode period k sets how often the episode runs. An episode every k steps replaces every kth baseline step by Algorithm 1; the other steps are ordinary. The temporary step is always undone before the optimizer step, so the persistent parameters see only the combined gradient, and the optimizer state is never exposed to the adapted parameters.

A.3 OBJECTIVE EXPANSION AND FIRST-ORDER UPDATE

Proposition 1 expands J to second order in the radius. For the full-parameter direction of Equation (9) and $\nabla L(\theta) \neq 0$, its first-order coefficient factors into a norm and an expected cosine,

$$\Lambda(\theta) = \langle \nabla L(\theta), \mathbb{E}_A\, u_A(\theta) \rangle = \|\nabla L(\theta)\|\, \rho(\theta), \qquad \rho(\theta) = \mathbb{E}_A \cos \angle\big(\nabla L_A(\theta), \nabla L(\theta)\big) \in [-1, 1], \tag{13}$$

where the first equality uses the independence of Q and A. The proof is a Taylor expansion of $L(\theta - \eta u_A)$ with $\|u_A\| = 1$, averaged over A (Appendix B). The first-order term is a validation projection, the expected projection of a held-out gradient onto the direction adapted on the other half. For a fixed plastic set the same expansion holds provided $\|P \nabla L_A\|_2 \geq \epsilon$ almost surely, and $\Lambda = \|P\nabla L\| \rho_P$ when $P\nabla L \neq 0$, where ρ_P uses the projected gradients. The directional curvature term need not be positive for a nonconvex loss. This is an expansion of J; it does not establish that the first-order estimator minimizes the expansion.

Writing $g = \nabla L$, $H = \nabla^2 L$, and $m = \mathbb{E}_A u_A$, a Taylor expansion of the expected first-order update instead gives

$$\mathbb{E}\hat{g}_{\mathrm{FO}} = g - \tfrac{\eta}{2} H m + O(\eta^2). \tag{14}$$

Under regularity allowing differentiation through the expectation, the exact gradient additionally contains $-\frac{\eta}{2}\mathbb{E}_A[(Du_A)^\top] g$ at this order. The two fields therefore need not produce the same changes in alignment along training.

The projection $\|\nabla L\| \rho$ depends on both norm and direction. Under a high-dimensional isotropic noise model with concentrated norm, $\rho \approx (1 + \sigma^2/\|\nabla L\|^2)^{-1/2}$ is related to the gradient signal-to-noise ratio studied by Liu et al. (2020); the projection itself is not that ratio. For the full-gradient directions of Equation (9), independent identically distributed unit gradients have expected pairwise cosine $\|m\|^2$. It equals ρ^2 when m is collinear with g, as under rotationally symmetric noise, but is not a general monotone proxy for ρ. Neither identity requires a higher cosine along the first-order T^3 trajectory than along the baseline's.

The sign is the opposite of sharpness-aware minimization, which adapts up the gradient and penalizes $\|\nabla L\|$ (Foret et al., 2021; Andriushchenko & Flammarion, 2022); reversing the sign of η turns the reward into a penalty. An independent, symmetric random direction of the same norm has $\mathbb{E}u_A = 0$, so $\Lambda = 0$ and only the curvature term survives. And with an episode every k steps the average objective is $L - \frac{\eta}{2k}\Lambda + \ldots$, so the local first-order coefficient scales with $1/k$. This coefficient alone does not predict linear scaling of the final validation gain with episode frequency.

In a raw-step quadratic model, the lookahead reduces the expected step more in directions of higher curvature.

Lemma 2 (Quadratic model). *Let $L(\theta) = \frac{1}{2}\theta^\top H \theta$ with $H \succeq 0$, $\eta > 0$, $\alpha > 0$, and $\eta\lambda_{\max}(H) < 2$, let the half-batch gradients be $\nabla L_A(\theta) = H\theta + \xi_A$ and $\nabla L_Q(\theta) = H\theta + \xi_Q$ with zero-mean noises that do not depend on θ, let the adaptation direction be the unnormalized gradient $u_A = \nabla L_A(\theta)$, and let the outer step be $\theta' = \theta - \alpha\, \hat{g}_{\mathrm{FO}}$. Then $\mathbb{E}\,\hat{g}_{\mathrm{FO}} = (I - \frac{\eta}{2}H)H\theta$, so in the eigenbasis of H the expected update multiplies coordinate i by $1 - \alpha\lambda_i(1 - \frac{\eta}{2}\lambda_i)$. The effective learning rate of direction i is $\alpha(1 - \frac{\eta}{2}\lambda_i)$, positive and smallest in the sharpest directions, and the expected iteration is stable whenever $0 < \alpha\lambda_i(1 - \frac{\eta}{2}\lambda_i) < 2$ for every positive eigenvalue; zero-eigenvalue directions are neutral. If H has a positive eigenvalue, the stable range of α is strictly larger than for gradient descent.*

The proof is in Appendix B. A normalized inner step uses the batch-dependent scalar $\eta/\|\nabla L_A\|$, which is correlated with the support gradient. It cannot generally be pulled outside the expectation in the lemma, so the raw-step calculation does not establish damping for the normalized update. A larger stability range does not imply faster convergence at a fixed outer learning rate, and the lemma does not cover the adaptive outer optimizers used in our main experiments. Appendix L measures training loss, alignment, and controlled variants without identifying a unique mechanism.

Placement. The query term is first-order MAML with the minibatch halves as support and query (Finn et al., 2017); T^3 averages it with the retained support gradient. For raw full-parameter inner steps with equal learning rates, this sum is proportional to the two-step Reptile displacement (Nichol et al., 2018). On the optimizer side it is an extragradient step with a descent inner step, closest to extrapolated SGD (Lin et al., 2020), which does not average the two gradients, and to the lookahead

optimizer (Zhang et al., 2019), which averages weights after several fast steps instead of gradients within one step; we use "lookahead" below for the temporary step of Algorithm 1. Sharpness-aware minimization is the ascent counterpart (Foret et al., 2021). Section 6 places these and the data-constrained literature more fully.

B PROOFS

Proof of Proposition 1. Under the assumption $\|P\nabla L_A(\theta)\|_2 \geq \epsilon$, Equation (1) gives $\|u_A\|_2 = 1$. Taylor-expand $L(\theta - \eta u_A) = L(\theta) - \eta\langle \nabla L(\theta),\, u_A\rangle + \frac{\eta^2}{2} u_A^\top \nabla^2 L(\theta) u_A + O(\eta^3)$; the remainder is uniform in A because $\|u_A\| = 1$ and the third derivative is bounded. Take the expectation over A and average with $L(\theta)$ as in Equation (3). This gives Equation (4) with $\Lambda = \langle \nabla L(\theta),\, \mathbb{E}_A u_A\rangle$, and the independence of Q from A together with $\mathbb{E}_Q \nabla L_Q(\theta) = \nabla L(\theta)$ gives $\Lambda = \mathbb{E}_{A,Q}\langle \nabla L_Q(\theta),\, u_A\rangle$. For Equation (13), which uses the full-parameter direction ($P = I$), u_A is parallel to ∇L_A, so $\langle \nabla L,\, u_A\rangle = \|\nabla L\| \cos \angle(\nabla L_A, \nabla L)$. □

Proof of Lemma 2. With $L(\theta) = \frac{1}{2}\theta^\top H\theta$, $u_A = H\theta + \xi_A$, and the test-half noise ξ_Q independent of the evaluation point, $\nabla L_Q(\theta - \eta u_A) = H\theta - \eta H(H\theta + \xi_A) + \xi_Q$. Averaging with $\nabla L_A = H\theta + \xi_A$ and taking expectations over the two noises gives $\mathbb{E}\,\hat{g}_{\mathrm{FO}} = H\theta - \frac{\eta}{2}H^2\theta = (I - \frac{\eta}{2}H)H\theta$. In the eigenbasis of H the expected update of coordinate i is $\theta_i' = \theta_i - \alpha\lambda_i(1 - \frac{\eta}{2}\lambda_i)\theta_i$, which is the stated multiplier; it lies in $(-1, 1)$ exactly when $0 < \alpha\lambda_i(1 - \frac{\eta}{2}\lambda_i) < 2$, whose left inequality follows from $\alpha > 0$, $\lambda_i > 0$, and $\eta\lambda_i < 2$. Its right inequality admits a larger α than gradient descent's $\alpha\lambda_i < 2$. If $\lambda_i = 0$, the multiplier equals one and the direction is neutral. This stability comparison does not imply faster convergence at the same α or bound the stationary variance of the noisy iterates. □

A one-step relation to gradient signal-to-noise ratio. The following statement links gradient noise to the difference between population loss and loss on the sampled training batch. The global cosine probe of Section L.2 does not identify this ratio, and this statement does not determine the T^3 training trajectory.

Proposition 3 (Gap growth per unit of training progress)**.** *Let S be a training sample of size b with $\mathbb{E}\,\nabla L_S(\theta) = \nabla L(\theta)$ and $\operatorname{tr}\operatorname{Cov}(\nabla L_S(\theta)) = \sigma_b^2(\theta)$, and let one step $\theta' = \theta - \alpha\nabla L_S(\theta)$ be taken. To first order in α, the loss on S decreases in expectation by $\alpha(\|\nabla L\|^2 + \sigma_b^2)$ and the population loss by $\alpha\|\nabla L\|^2$, so the gap $L - L_S$ grows by $\alpha\sigma_b^2$, and the fraction of the training progress that is not population progress is $\sigma_b^2/(\|\nabla L\|^2 + \sigma_b^2) = 1/(1 + \mathrm{GSNR}_b(\theta))$ with $\mathrm{GSNR}_b = \|\nabla L\|^2/\sigma_b^2$.*

Proof. $L_S(\theta') - L_S(\theta) = -\alpha\|\nabla L_S\|^2 + O(\alpha^2)$ and $L(\theta') - L(\theta) = -\alpha\langle \nabla L,\, \nabla L_S\rangle + O(\alpha^2)$; take expectations over S using $\mathbb{E}\|\nabla L_S\|^2 = \|\nabla L\|^2 + \sigma_b^2$ and $\mathbb{E}\langle \nabla L,\, \nabla L_S\rangle = \|\nabla L\|^2$. □

For a half-batch gradient with mean ∇L and isotropic noise of total variance σ^2 in high dimension, where the noise norm concentrates at σ and is nearly orthogonal to ∇L, the expected cosine of Equation (13) is $\rho \approx (1 + \sigma^2/\|\nabla L\|^2)^{-1/2}$. Under rotational symmetry around the nonzero population gradient, $m = \mathbb{E}_A u_A$ is collinear with that gradient and the expected pairwise cosine $\|m\|^2$ equals ρ^2. Monotonicity in ρ additionally requires $\rho \geq 0$. Without collinearity, pairwise cosine does not generally determine ρ. These conditional identities do not guarantee a higher cosine along a T^3 trajectory. The ratio here is global, whereas Liu et al. (2020) study a per-parameter version.

C FEATURE ALLOCATION AND GENERALIZATION

This appendix gives the constants and proof for Theorem 1. We first specify the finite-data model and three update rules, then prove the risk and feature orderings for their actual first-hitting checkpoints.

C.1 MODEL AND UPDATE RULES

Fix an unknown orthonormal basis $u_0, \ldots, u_M$ of $\mathbb{R}^{M+1}$. Independently draw y, v uniformly from $\{-1, +1\}$ and J uniformly from $\{1, \ldots, M\}$. The input consists of the patches (yu_0, vu_J) in random order, and the target is y. The prediction and sample loss are

$$f_w(x_1, x_2) = h(w^\top x_1) + h(w^\top x_2), \qquad h(z) = z + z^3, \qquad \ell(w; x, y) = \tfrac{1}{2}(y - f_w(x))^2.$$

Every coordinate of w is trained. The learner is not given the basis or the signal direction u_0. Draw $n \geq 2$ examples once and reuse this finite dataset S. Write $L_S(w) = n^{-1}\sum_i \ell_i(w)$ and $R(w) = \mathbb{E}\ell(w; x, y)$, where the expectation in R is over an independent population example.

Let $a = w^\top u_0$ and define the shared-signal output and nuisance energy by

$$\kappa(w) = h(a), \qquad N(w) = \frac{1}{M}\sum_{j=1}^{M} h(w^\top u_j)^2.$$

Oddness of h gives $f_w = yh(a) + vh(w^\top u_J)$. Independence of the fresh signs therefore yields the exact identity

$$R(w) = \tfrac{1}{2}\big[(1-\kappa(w))^2 + N(w)\big]. \tag{15}$$

This identity holds for every learned w, including a parameter that depends on all of S. Noise output always adds to fresh risk.

Three training rules. Repeat a fixed cycle containing all $K = n(n-1)$ ordered pairs (i, j) with $i \neq j$, for example in lexicographic order. For $g_i = \nabla\ell_i$, define

$$u_i(w) = \frac{g_i(w)}{\max\{\|g_i(w)\|_2, \epsilon\}}, \qquad \phi = w - \eta u_i(w).$$

Each method updates its permanent parameter by $w^+ = w - \alpha_D G_D(w; i, j)$, with its own constant outer rate $\alpha_D > 0$, where

$$\begin{aligned} G_{\text{ordinary}}(w; i, j) &= \tfrac{1}{2}\big[g_i(w) + g_j(w)\big], \\ G_{\text{split}}(w; i, j) &= \tfrac{1}{2}\big[g_i(w) + g_j(\phi)\big], \\ G_{\text{self}}(w; i, j) &= \tfrac{1}{2}\big[g_i(w) + g_i(\phi)\big]. \end{aligned}$$

The temporary parameter is discarded. There is no differentiation through the inner update, and the normalization acts on all trainable coordinates. Every step computes two singleton gradients. Over a complete cycle, each example receives $2(n-1)$ gradient evaluations under each rule. Self-testing repeats one example within a step.

C.2 Detailed statement

The statement below gives the choices $c = 1/60\,000\,000$ and $C = 2^{16}$ in Theorem 1, together with a bound on the stopping times.

Theorem 1 (detailed statement). *Fix $n \geq 2$, $M \geq n$, $0 < \rho < 1$, a target residual $0 < r_* \leq 1/2$, and $0 < \eta \leq r_*/32$. Set*

$$\epsilon = r_*/8, \qquad \ell_* = r_*^2/2, \qquad \gamma = \frac{r_*^5\eta}{30\,000\,000\,n^2}, \qquad T = 2\log(2/r_*) + 1.$$

There are positive constants $\alpha_0(n, r_*, \eta)$ and $\sigma_0(n, r_*, \eta, \rho)$, specified in the proof, with the following property. Choose the rates $0 < \alpha_D \leq \alpha_0$ and $0 < \sigma \leq \sigma_0$ before drawing S and a common initialization $w_0 \sim N(0, \sigma^2 I_{M+1})$. For each of the three rules, let

$$\tau_D = \inf\{k \geq 0 : L_S(w_{D,k}) \leq \ell_*\}, \qquad \widehat{w}_D = w_{D,\tau_D}.$$

With probability at least $1 - \rho - n(n-1)/(2M)$, all stopping times are finite, $\tau_D \leq \lceil T/\alpha_D \rceil$, and

$$R(\widehat{w}_{\text{split}}) + \gamma/2 \leq R(\widehat{w}_{\text{self}}), \qquad R(\widehat{w}_{\text{self}}) + \gamma/2 \leq R(\widehat{w}_{\text{ordinary}}). \tag{16}$$

At these checkpoints,

$$\kappa(\widehat{w}_{\text{split}}) > \kappa(\widehat{w}_{\text{self}}) > \kappa(\widehat{w}_{\text{ordinary}}), \qquad N(\widehat{w}_{\text{split}}) < N(\widehat{w}_{\text{self}}) < N(\widehat{w}_{\text{ordinary}}). \tag{17}$$

Each observed nuisance coordinate has strictly smaller absolute value under split than self, and under self than ordinary. The squared alignment $(w^\top u_0)^2/\|w\|_2^2$ has the reverse strict ordering. The empirical-fit overshoot is bounded by

$$r_* - 2^{16}\alpha_D \leq \sqrt{2L_S(\widehat{w}_D)} \leq r_*. \tag{18}$$

Scope of the comparison. The comparison concerns the permanent predictor at a common training-loss target. It separates learning allocation from training speed. It holds uniformly over independent choices of constant outer rate in the stated interval. The sufficient interval is extremely conservative and is an existence guarantee, not a practical prescription for a learning rate. The cost at a checkpoint is exactly $2\tau_D$ example-gradient evaluations. These costs can differ between methods. Choosing $M \geq n(n-1)/(2\rho)$ makes the success probability at least $1-2\rho$. The target r_* can be arbitrarily small and positive, but the allowed radius must decrease with it. The theorem does not assert convergence to zero training loss with a fixed positive radius.

C.3 Proof of Theorem 1

C.3.1 Exact finite-data coordinates

The nuisance indices in S are distinct with probability at least $1-n(n-1)/(2M)$. Condition on this event. Orient the observed nuisance axes for analysis by $\widetilde{u}_i = y_i v_i u_{J_i}$ and put $b_i = w^\top \widetilde{u}_i$. The coordinates $(a, b_1, \dots, b_n)$ are orthonormal parameter coordinates. The sample residual and gradient become

$$r_i = 1 - h(a) - h(b_i), \qquad g_i = -r_i\big(h'(a)e_0 + h'(b_i)e_i\big). \tag{19}$$

All unobserved coordinates remain unchanged under all three rules. Their contribution to (15) cancels in every pairwise risk difference.

When $r_i > 0$ and the floor is inactive, write

$$A = h'(a), \quad B_i = h'(b_i), \quad q_i = (A^2 + B_i^2)^{1/2}, \quad c_i = a + \eta A/q_i, \quad d_i = b_i + \eta B_i/q_i.$$

The exact split update is

$$\begin{aligned} a^+ &= a + \frac{\alpha_{\text{split}}}{2}\big[r_i A + (1 - h(c_i) - h(b_j))h'(c_i)\big], \\ b_i^+ &= b_i + \frac{\alpha_{\text{split}}}{2} r_i B_i, \\ b_j^+ &= b_j + \frac{\alpha_{\text{split}}}{2}(1 - h(c_i) - h(b_j))B_j. \end{aligned} \tag{20}$$

Ordinary training replaces c_i by a in this expression and uses its own rate. Self-testing instead updates only a, b_i by

$$\begin{aligned} a^+ &= a + \frac{\alpha_{\text{self}}}{2}\big[r_i A + (1 - h(c_i) - h(d_i))h'(c_i)\big], \\ b_i^+ &= b_i + \frac{\alpha_{\text{self}}}{2}\big[r_i B_i + (1 - h(c_i) - h(d_i))h'(d_i)\big]. \end{aligned} \tag{21}$$

These are finite-radius identities. Below we verify the positive residuals and inactive floor for the actual stopped trajectories.

C.3.2 Reference trajectories and their learning allocation

Average the finitely many actual update fields over a complete ordered-pair cycle. Start this reference ODE at zero. This zero start is only a proof device; the implemented initialization remains Gaussian. Permutation symmetry keeps its observed coordinates equal to a common b. At such a state define

$$A = h'(a), \quad B = h'(b), \quad q = (A^2 + B^2)^{1/2}, \quad c = a + \eta A/q, \quad d = b + \eta B/q,$$
$$r = 1 - h(a) - h(b), \qquad r_s = 1 - h(c) - h(b), \qquad r_h = 1 - h(c) - h(d).$$

The averaged equations are

D	$\dot{a}$	$\dot{b}$
ordinary	rA	rB/n
split	$(rA + r_s h'(c))/2$	$B(r + r_s)/(2n)$
self	$(rA + r_h h'(c))/2$	$(rB + r_h h'(d))/(2n)$.

(22)

This is a finite-sample arithmetic average. No conditional training gradient has been replaced by a fresh population gradient.

On $0 \le b \le a$ and $r \ge r_*/2$, we have $a, b < 0.7$, $c, d < 3/4$, $1 \le A, B \le 5/2$, and $h'(c), h'(d) \le 3$. Indeed $h(0.7) > 1$ and $\eta \le 1/64$. The reference support norm is $rq \ge r_*/\sqrt{2} > \epsilon = r_*/8$, so the averaged equations use the inactive-floor branch throughout this region. Consequently

$$r_h \ge r - 8\eta \ge r_*/4 > 0, \qquad r_s > r_h. \tag{23}$$

All coordinate velocities are positive. The slope calculations below imply that the region $a \ge b$ is invariant, including at the initial boundary. Also

$$-\dot{r} = A\dot{a} + B\dot{b} \ge r/2. \tag{24}$$

Thus each reference trajectory reaches $r = r_*/2$ by time $2\log(2/r_*)$, without leaving this region.

Because $\dot{b} > 0$, parameterize a reference curve by b. Let $k = h'(c)/A$ and $l = h'(d)/B$. Its slope $F_D = da/db$ is

$$F_{\text{ordinary}} = \frac{nA}{B}, \qquad F_{\text{split}} = \frac{nA}{B}\frac{r + r_s k}{r + r_s}, \qquad F_{\text{self}} = \frac{nA}{B}\frac{r + r_h k}{r + r_h l}. \tag{25}$$

For $\zeta = \eta/q$, polynomial expansion gives

$$k = 1 + 6\zeta a + 3\zeta^2 A, \qquad l = 1 + 6\zeta b + 3\zeta^2 B, \qquad k - l = (a - b)\big[6\zeta + 9\zeta^2(a + b)\big].$$

Hence $k \ge l > 1$, with $k > l$ if $a > b$. Using (23),

$$\frac{r + r_s k}{r + r_s} > \frac{r + r_h k}{r + r_h} > \frac{r + r_h k}{r + r_h l} \ge 1. \tag{26}$$

The last inequality is strict when $a > b$. Every slope lies between n and $3n$: for split, its ratio in (25) is at most k, so $F_{\text{split}} \le nh'(c)/B \le 3n$; the other slopes are smaller. In particular,

$$nb \le a_D(b) \le 3nb \le 4nb. \tag{27}$$

This also proves the asserted invariance from the origin, since $n \ge 2$.

Scalar ODE comparison, applied at common values of (a, b), now gives

$$a_{\text{split}}(b) > a_{\text{self}}(b) > a_{\text{ordinary}}(b) \quad (b > 0)$$

until the faster curve reaches its target. For self versus ordinary, the slopes agree at the origin but are strictly ordered as soon as $b > 0$, because $a \ge nb > b$. Strictness follows by integrating the one-sided Lipschitz comparison for their difference. Comparing slopes at different states would not by itself prove this ordering. Let (a_D^*, b_D^*) denote the intersection with $h(a) + h(b) = 1 - r_*$. Monotonicity of this contour in both coordinates implies

$$a^*_{\text{split}} > a^*_{\text{self}} > a^*_{\text{ordinary}}, \qquad b^*_{\text{split}} < b^*_{\text{self}} < b^*_{\text{ordinary}}. \tag{28}$$

At this contour, writing $z = h(b)$, the ideal population risk is

$$R^*(z) = \tfrac{1}{2}\big[(r_* + z)^2 + (n/M)z^2\big], \qquad \frac{dR^*}{dz} = r_* + (1 + n/M)z > 0. \tag{29}$$

Thus improved feature allocation directly lowers population risk.

C.3.3 A QUANTITATIVE GAP

We track a positive margin to transfer the ordering to finite updates. On the region $r \ge r_*$, $q < 4$ and the polynomial identities imply

$$l - 1 \ge \eta b/2, \qquad k - l \ge \eta(a - b).$$

Also $r_h \ge 3r_*/4$, $r, r_s, r_h \le 1$, and $k, l \le 4$. From (26), retain the difference

$$\frac{r + r_h k}{r + r_h} - \frac{r + r_h k}{r + r_h l} = \frac{(r + r_h k) r_h (l - 1)}{(r + r_h)(r + r_h l)}.$$

Its denominator is at most 10. For self versus ordinary, use $r_h(k - l)/(r + r_h l)$, whose denominator is at most 5. Since $A/B \ge 1$ and $a - b \ge b$, these bounds give

$$F_{\text{split}} - F_{\text{self}} \ge \frac{nr_*^2 \eta b}{40}, \qquad F_{\text{self}} - F_{\text{ordinary}} \ge \frac{nr_*^2 \eta b}{40}. \tag{30}$$

A uniform derivative bound on the same region is

$$|\partial_a F_D| \le L_c := 1000n/r_*^2. \tag{31}$$

For completeness, $|\partial_a q| \le 5$, $|\partial_a(\eta A/q)| \le 4\eta$, and $|\partial_a(\eta B/q)| \le 2\eta$. The last two inequalities follow by differentiating the normalized vector: $\partial_a(A/q) = A'B^2/q^3$ and $\partial_a(B/q) = -BAA'/q^3$. Consequently $|\partial_a r_s| \le 6$, $|\partial_a r_h| \le 7$, $|\partial_a h'(c)| \le 10$, and $|\partial_a h'(d)| \le 1$. Writing $F_D/n = P_D/Q_D$, the split and self numerators are at most 6, their derivative magnitudes are at most 46 and 62, the denominators are at least r_*, and their derivative magnitudes are at most 30 and 56. The quotient rule bounds the respective derivatives by $226/r_*^2$ and $398/r_*^2$. Ordinary has a smaller bound.

At every target, $h(t) \le 2t$ for $0 \le t \le 0.7$ and (27) imply

$$b_D^* \ge \frac{1 - r_*}{9n} \ge \frac{1}{18n}. \tag{32}$$

Consider either adjacent fast/slow pair in (28). For $0 \le b \le b_{\text{fast}}^*$, put $D(b) = a_{\text{fast}}(b) - a_{\text{slow}}(b) \ge 0$. Equations (30)–(31) imply

$$D'(b) \ge -L_c D(b) + nr_*^2 \eta b/40.$$

The interval $[b_{\text{fast}}^* - 1/L_c, b_{\text{fast}}^*]$ lies in $[b_{\text{fast}}^*/2, b_{\text{fast}}^*]$ by (32). Integrating over only this interval, and using $1 - e^{-1} \ge 1/2$, gives

$$D(b_{\text{fast}}^*) \ge \frac{r_*^4 \eta}{2\,880\,000\, n}. \tag{33}$$

Let $\Delta_a = a_{\text{fast}}^* - a_{\text{slow}}^* > 0$ and $\Delta_b = b_{\text{slow}}^* - b_{\text{fast}}^* > 0$. Equal fit and $1 \le h' \le 5/2$ give $\Delta_b \le (5/2)\Delta_a$ and $\Delta_b \ge \Delta_a/3$. Since the slow slope is at most $3n$,

$$D(b_{\text{fast}}^*) \le \Delta_a + 3n\Delta_b \le 9n\Delta_a.$$

Both signal deficits $1 - h(a_D^*)$ are at least r_*. The signal term alone in (15) therefore improves by at least $r_*\Delta_a$. The nuisance term also improves. We obtain

$$R_{\text{slow}}^* - R_{\text{fast}}^* \ge \frac{r_*^5 \eta}{25\,920\,000\, n^2} > \gamma, \qquad \Delta_a \ge \frac{\gamma}{r_*}, \quad \Delta_b \ge \frac{\gamma}{3r_*}. \tag{34}$$

C.3.4 Finite updates and random initialization

All norms in this subsection concern the $n+1$ active coordinates. We give explicit sufficient constants:

$$\begin{aligned} V &= 2048, & L_0 &= 2^{18}, & E &= e^{L_0 T}, \\ \delta &= \min\{r_*/512, \gamma r_*/2^{33}, 1/8\}, & \alpha_0 &= \frac{\delta}{4VKE}, \\ \sigma_0 &= \frac{\delta}{2E\sqrt{n+1}\sqrt{2\log(2(n+1)/\rho)}}. \end{aligned} \tag{35}$$

Every displayed bound is positive and is fixed before sampling. These deliberately loose constants establish a nonempty regime.

On the active coordinate box of radius 2 in the maximum norm, $|h| \le 10$, $h' \le 13$, $|h''| \le 12$, and $|r_i| \le 21$. Thus $\|g_i\|_2 < 387$ and $\|\nabla^2 \ell_i\|_{\text{op}} \le 590$. Temporary coordinates lie in the box of radius 3, where $\|g_i\|_2 < 2416$ and $\|\nabla^2 \ell_i\|_{\text{op}} \le 2666$. The map $g \mapsto g/\max\{\|g\|_2, \epsilon\}$ is $1/\epsilon$-Lipschitz, since it is projection of g/ϵ onto the unit ball. Because $\eta/\epsilon \le 1/4$, every actual update field has norm at most V and Lipschitz constant at most

$$\tfrac{1}{2}\big[590 + 2666(1 + 590/4)\big] < L_0.$$

The floor is included in these estimates.

Periodic-update error. Let $F_1, \dots, F_K$ be the update fields $-G_D$ for the ordered pairs in one cycle, and $\overline{F} = K^{-1}\sum_j F_j$ their average. Let $x(t)$ solve $\dot{x} = \overline{F}(x)$ from zero. For step size α and $H = K\alpha$, one cycle differs from $x + H\overline{F}(x)$ by at most $L_0 V H^2/2$. This follows by summing the within-cycle movements $j\alpha V$. The averaged ODE differs from the same Euler step by at most $L_0 V H^2/2$. The cycle map is $e^{L_0 H}$-Lipschitz, so iteration bounds the cycle-end error by

$VH(e^{L_0 t} - 1)$, before adding the initialization error. Moving to any intermediate microstep costs at most $2VH$. Consequently

$$\|w_k - x(k\alpha)\|_2 \le E\big(\|w_{0,\text{active}}\|_2 + 2VK\alpha\big) \tag{36}$$

for grid times up to the reference hitting time $t_{1/2}$ of $r_*/2$. Only this part of the reference path is needed. A first-exit argument makes the coordinate bounds noncircular: the reference coordinates are less than 1, the error bound below is at most $\delta \le 1/8$, and a step has length at most $V\alpha$. Hence an actual path cannot be the first to leave the radius-2 box before this reference stopping time.

Conditional on any distinct dataset, the oriented active Gaussian coordinates remain independent $N(0, \sigma^2)$ variables. With conditional probability at least $1 - \rho$,

$$\max_{0 \le i \le n} |w_{0,i}| \le \sigma\sqrt{2\log(2(n+1)/\rho)} \le \frac{\delta}{2E\sqrt{n+1}}.$$

On this event, (35) makes (36) at most δ, for each method and any of its allowed rates. No restriction on the signs of the initialization is imposed.

Actual first-hitting checkpoints. The empirical RMS residual

$$Q(w) = \sqrt{2L_S(w)} = \left(\frac{1}{n}\sum_i [1 - h(a) - h(b_i)]^2\right)^{1/2}$$

is 32-Lipschitz on the radius-2 box, and $Q(x(t)) = r(t)$ on a reference trajectory. A single microstep changes Q by at most $32V\alpha$. This proves the overshoot bound once the first hit exists. Initially $Q > r_*$ on the Gaussian event. At $k = \lfloor t_{1/2}/\alpha \rfloor$, tracking gives

$$Q(w_k) \le r_*/2 + 32V\alpha + 32\delta \le 5r_*/8 < r_*.$$

Here $\alpha \le \delta/V$, $\delta \le r_*/512$, and $t_{1/2} \le T - 1$. Thus the first hit exists and $\tau_D \le \lceil T/\alpha_D \rceil$. At that hit,

$$|r(\tau_D \alpha_D) - r_*| \le 32\delta + 32V\alpha_D.$$

Along the reference segment between this time and its target crossing t_D^*, (24) gives $|\dot{r}| \ge r_*/4$. Its speed is at most V. We conclude

$$\|\widehat{w}_{D,\text{active}} - x_D(t_D^*)\|_2 \le \left(1 + \frac{256V}{r_*}\right)\delta \le \frac{2^{20}\delta}{r_*}. \tag{37}$$

Every permanent sample residual before and at this checkpoint is at least $r_*/2 - 32\delta > r_*/8$. This follows by applying the same coordinate Lipschitz bound to each sample residual. Since $h' \ge 1$, its gradient norm exceeds $\epsilon = r_*/8$. Thus the exact normalized recurrences used above apply through every stop.

Risk and feature transfer. Remove the common, unchanged contribution of unobserved coordinates from (15). The remaining function is 512-Lipschitz on the active radius-2 box. Its signal derivative has magnitude at most 143, and its nuisance-gradient norm is at most $130\sqrt{n}/M \le 130$. By (37), the combined error in either adjacent risk difference is at most

$$\frac{2^{30}\delta}{r_*} \le \gamma/8.$$

Together with (34), this proves (16). Each coordinate error is at most $2^{20}\delta/r_* \le \gamma/8192$. The ideal coordinate gaps in (34) therefore survive strictly. The ideal observed nuisance coordinates are at least $1/(18n)$, so their perturbed values are positive at the checkpoints. Their strict coordinate ordering proves the strict nuisance-energy ordering. Monotonicity of h proves the signal-output ordering. The shared coordinate is positive and larger, while every observed nuisance magnitude is smaller and every unobserved magnitude is identical. This proves the alignment claim. Finally, the Gaussian event and the distinct-index event have joint probability at least $1 - \rho - n(n-1)/(2M)$. Since $32V = 2^{16}$, the overshoot estimate is (18). This completes the proof. □

C.4 Interpretation and scope

Learning allocation versus speed. Multiplying an averaged vector field by a positive scalar changes its clock but leaves da/db unchanged. Ordinary training's reference curve satisfies

$$\arctan(\sqrt{3}\,a) = n\arctan(\sqrt{3}\,b).$$

Split testing changes this curve and hence its intersection with a fixed-fit contour. The finite-update theorem allows different sufficiently small constant rates, but does not cover arbitrary stable rates, momentum, or adaptive optimizers.

Roles of nonlinearity and nuisance. If $h(z) = z$, the derivative ratios k, l in the proof both equal one, so all three reference slopes equal n. Their allocation and risk at matched fit then agree. If the nuisance patch is removed, matched fit fixes the only prediction coordinate, and the risk separation again disappears. Both the nonlinear response and the alternative of memorizing nuisance are needed for this argument. The signal changes by a nonvanishing amount: $h(a_D^*) \geq n(1-r_*)/(n+1) \geq 1/3$, by (27) and convexity of h. The model nevertheless describes only one nonlinear feature on a restricted input distribution.

Measurements suggested by the result. Within this model, signal error $(1-\kappa)^2$, nuisance energy N, and population risk R can be measured directly at matched training-loss thresholds. The training-label correlation of the nuisance output, $n^{-1}\sum_i y_i v_i h(w^\top u_{J_i})$, is also smaller under split than self or ordinary at the theorem's checkpoints. These comparisons follow from the coordinate ordering; they are distinct from comparing training loss at a fixed iteration.

Limits of the model. Self-testing also improves over ordinary training in this construction. The result therefore establishes an additional benefit from cross-example testing, while leaving the empirical failure of self-testing unexplained. The orthogonal nuisance support, cyclic schedule, residual-cubic activation, and small-step regime are assumptions of the theorem. The proof does not establish the same result for random minibatches, general nuisance geometry, or Transformer training.

D Cost, tokens, and reproducibility

Step cost. Table 1 gives the median optimizer-step time of each design at 300M in the setting of Section 3.1. Split pairing uses the same two half-batch gradients as ordinary accumulation, adding normalization, the temporary step, and restoration; consecutive pairing at every step adds a full gradient. On the SlowRun one-hour track the added work is about 12 ms of a 1.5 to 1.9 s step (norm and temporary step 10.5 ms, restoration 0.8 ms).

design (300M, batch 64×4096)	plain step (s)	episode step (s)	relative cost
baseline	0.609	–	1.00×
every step, split (normalized or raw)	–	0.648	1.06×
every step, consecutive	–	1.216	2.00×

Table 1: Median step times at 300M, relative to the baseline's step time. At 1.4B the every-step design's step is 1.401 s against the baseline's 1.320 s (1.06×), at 600M 1.443 against 1.323 s (1.09×), and at 150M 0.324 against 0.280 s (1.16×).

Tokens. Split pairing consumes exactly the baseline's tokens and gradient evaluations, so every comparison of the every-step design with its baseline is at equal tokens and equal steps.

Receipts. The newly audited Muon and one-pass studies retain resolved configurations, acceptance records, and source/output identities. The primary paired-loss figure uses AdamW seed pairs from archived four-decimal member tables. Its scripts retain this precision rather than treating rounded table entries as full-precision measurements. The SlowRun submissions are pull requests 98 (one-hour track) and 99 (tiny track) to the benchmark repository (qlabs-eng, 2026), with the training logs of the

submitted runs attached as public gists. Archived experiment artifacts retain evaluation code, launch configurations, and the campaign's prediction and outcome records.

E SWEEPS AND REPLICATIONS

This appendix collects the tuning sweeps and replications behind the main text, one table per block. Unless noted, values are validation losses or differences in nats per token, a difference is the variant minus its stated comparator (negative favors the variant), and a sweep is a single run per setting. Detailed tables with every run are in the appendices cited in each caption.

E.1 T^3 SETTINGS AT 300M

The settings of the temporary step at 300M (length, direction and preconditioning, plastic set, pairing, and episode period) and the outer neighbours with T^3 in place are listed in Appendix F (Table 7); the sharpness-aware minimization sweep is in Appendix I.

E.2 COMPARISONS OF SECTION 3.1

Table 2: **Settings of the comparisons in Section 3.1.** All rows use the 300M model on the DCLM corpus; differences are T^3 minus the comparator in nats per token. Seed 41 is the tuning seed.

comparison	setting	Δ
best of seven SAM radii	certified AdamW recipe, seed 41; step time 1.94× (SAM), 1.06× (T^3)	−0.0178
one pass, Muon, equal updates	one pass over the pool (750 updates), seed 41	−0.0246
best of an expanded baseline grid	ordinary Muon at learning rates 0.003 and 0.006	−0.0224
equal training time	best ordinary or zero-displacement baseline	−0.0112
SlowRun Newton–Schulz coefficients	best of four baselines	−0.0178

E.3 INTERACTION WITH THE RECIPE

Table 3: **T^3 and the outer recipe, 300M, tuning seed.** Outer neighbours in Table 23; pass count in Appendix L; learning-rate bracket from the matched-control campaign.

comparison	setting	Δ
outer neighbours with T^3	LR $\times\sqrt{2}$, $/\sqrt{2}$; WD 1.2, 2.0; radii 0.1, 0.3, 0.7	certified point best or within 0.003
24 passes	T^3 3.5623 vs baseline 3.5918	−0.0295
32 passes	T^3 3.6085 vs baseline 3.6205	−0.0120
vs own zero, LR 0.0015 / 0.003 / 0.006	same computation with radius 0	−0.0039 / −0.0245 / −0.0838

Extra passes hurt both methods (by 0.04 to 0.09), so T^3 does not make passes beyond the certified sixteen useful. At learning rate 0.006 both arms are worse than at 0.003, so the large difference there is against a poor baseline.

Table 4: **Replications of the fixed-data gain.** Differences are T^3 minus the same-seed baseline.

replication	detail	Δ
tuning seed at every size	150M / 300M / 600M / 1.4B, seed 41	-0.002 / -0.025 / -0.006 / -0.021
150M, four seeds	three of four pairs improve	-0.0045 ± 0.0057
150M, four-member ensemble		-0.0016
300M, independent data orders	data seed $43 + s$, four seeds, all improve	-0.0115 ± 0.0038
four-member ensemble	3.3646 vs 3.3703	-0.0057
fitted asymptote L_∞	$L = L_\infty + CN^{-\beta}$, four sizes, seed 41	3.404 vs 3.414

Table 5: **Muon at 300M, tuning seed unless noted.** In-domain DCLM loss. Full account in Appendices H and K.

variant	comparator	Δ
primary recipe	zero displacement	-0.0022 (ensemble -0.0028)
SlowRun Newton–Schulz coefficients	zero displacement	-0.0048 (ensemble -0.0051)
radii 0.03 / 0.1	zero displacement	$+0.0007$ / $+0.0001$
one pass, 750 updates, SlowRun coefficients	best of four baselines	-0.0178

E.4 Replications

E.5 Muon variants

E.6 SlowRun

E.7 Tiny-track probes

Table 22 lists every probe arm on both seeds. At the recipe's batch, T^3 gains 0.0090 and 0.0110 with half the learning rate and 0.0136 and 0.0124 with twice the learning rate (EMA endpoint, seeds 42 and 43). Without dropout the record script is 0.012 worse and T^3 gains 0.0162 and 0.0114. Without weight decay the record script overfits, ending near 4.1 instead of 3.3, and the gain is not reliable ($+0.0149$ and -0.0060). Without the diagnostic probes the pair gains 0.0117 and 0.0065.

Table 6: **SlowRun sweeps.** One-hour track, seed 42, record script 3.2005, unless noted; score is the checkpoint average. "Synced" averages the support direction over the batch before normalizing. Full record in Appendix M.

run	result
synced, every step, seeds 42–44	-0.0089 / -0.0042 / -0.0070; seed 42 took 61.4 min
synced, episode every 16 / 8 / 4 / 2 steps	-0.002 to -0.003; -0.005 at 2
synced, every 16 steps, seeds 42–44	-0.0029 / $+0.0002$ / -0.0020
synced, norm 0.1 / 0.3 / 1.0; MLP matrices only	-0.0074 / -0.0090 / $+0.0083$; -0.0093
episodes in the first half only; in the averaging phase only	keeps 0.007 of 0.011; keeps 0.003
tiny script with diagnostics, seeds 42 / 43	-0.0155 / -0.0113
query gradient alone, consecutive batches	$+0.021$
unequal weights on the two half-gradients	$+0.03$ to $+0.05$

F HYPERPARAMETER SEARCH GRIDS

This appendix lists the values we searched for every recipe in the paper and the value we selected. All searches used one seed, model seed 41 with data seed 42 (the tuning seed, which the multi-seed results of Section 3.1 exclude), one run per value.

Baselines. We did not retune the AdamW baselines. They are the certified recipes of Kim et al. (2025), whose learning rate, weight decay, and number of passes were tuned jointly for each size (Table 9). The baseline checks reported in this paper are the pass count at 300M (16, 24, and 32 passes; 16 is best) and a lower weight decay at 1.4B (1.8 instead of 3.2), in Appendices E and G. The Muon baseline is our own and was tuned before any T^3 run on it (Table 8). The one-pass baselines searched only the learning rate, and the modded-nanogpt one-pass recipes are used as published. The SlowRun baselines are the published record scripts, used unchanged; our search on that benchmark changed only the T^3 script.

T^3. We searched the temporary step on the certified 300M AdamW recipe one setting at a time and then checked the outer learning rate and weight decay with the step in place (Table 7). The step lengths 0.1 and 0.7 were also crossed with the four outer neighbours (Table 23). The selected step was transferred to 150M, 600M, and 1.4B without retuning.

Where the records come from. The run records behind these grids are collected in `data/tuning_records.json` (built by `scripts/build_tuning_data.py`); the SAM and modded-nanogpt records are in Appendices I and O.

Table 7: **T^3 search on the fixed-data AdamW recipes.** 300M: certified recipe (learning rate 3×10^{-3}, weight decay 1.6, 16 passes), one setting varied at a time from the selected step. For inner steps other than the normalized one, the values are the inner learning rate. Other sizes: the selected step on each size's certified recipe, with the listed outer checks at 1.4B.

setting	values searched	selected
300M, temporary step		
step length η	0.1, 0.2, 0.3, 0.5, 0.7	0.3
inner step	normalized; normalized per tensor ($\eta = 0.3$); raw (0.1, 0.2); sign (10^{-5}, 3×10^{-5}); Adam-preconditioned (10^{-5} to 3×10^{-4}, 4 values); along the outer update (3×10^{-5} to 10^{-3}, 4 values)	normalized
direction	support gradient; persistent average of support gradients ($\beta = 0.9$, 0.99); random; ascent	support gradient
plastic set	MLP matrices of the second half of the layers; all parameters ($\eta = 0.3$, 1)	second-half MLP
pairing	other half of the batch; consecutive batches ($\eta = 0.2$); same batch	other half
episode period k	1, 2, 4, 8 steps	1
300M, outer recipe with T^3 in place		
learning rate	2.12×10^{-3}, 3×10^{-3}, 4.24×10^{-3}	3×10^{-3} (certified)
weight decay	1.2, 1.4, 1.6, 2.0	1.6 (certified)
other sizes (certified learning rate, weight decay, passes)		
150M (3×10^{-3}, 0.8, 16)	step transferred, not searched	300M step
600M (10^{-3}, 3.2, 8)	step transferred, not searched	300M step
1.4B (10^{-3}, 3.2, 8)	learning rate 1.5×10^{-3} with T^3; weight decay 1.8 with and without T^3	certified

Table 8: **Search grids of the other recipes.** Muon: 300M, 16 passes; the auxiliary AdamW group keeps learning rate 3×10^{-3} and weight decay 1.6. The T^3 radius was compared with a zero-radius control on the same split halves. SAM: canonical sharpness-aware minimization on the certified 300M AdamW recipe, whose learning rate and weight decay were not retuned (Appendix I). One pass: 300M, one pass over the fixed pool (750 updates); the ordinary baseline's best learning rate is the smallest tested. modded-nanogpt: the published MuonW and AdamW recipes of the one-pass track (Appendix O); no T^3 setting gave a reliable gain, so none was selected. *Fixed in advance as the primary radius; radius 1 was added later and is reported as secondary.

setting	values searched	selected
Muon, fixed data		
matrix learning rate	7.5×10^{-4}, 1.5×10^{-3}, 3×10^{-3}, 6×10^{-3}, 1.2×10^{-2} (weight decay 1.6)	6×10^{-3}
weight decay	0.8, 1.6, 2.4 (learning rate 6×10^{-3})	1.6
T^3 radius	0 (split halves), 0.03, 0.1, 0.3, 1	0.3
sharpness-aware minimization		
radius ρ	0.003, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2	0.01
one pass, Muon		
matrix learning rate	3×10^{-3}, 6×10^{-3}, 1.2×10^{-2} (standard Newton–Schulz coefficients); 3×10^{-3}, 6×10^{-3} (SlowRun coefficients)	all reported
arm	ordinary; zero-radius control; T^3 radius 0.3; radius 1 (SlowRun coefficients only)	radius 0.3*
one pass, modded-nanogpt		
MuonW T^3	radius 0.2, 0.4; support direction per support partition or averaged over the batch (131K batch); averaged (524K batch)	none
AdamW T^3	radius 0.2, 0.4, 0.8, 1.6, averaged (131K and 524K batches)	none

G SETTINGS WITHOUT A GAIN

The gain belongs to recipes that train for many passes over limited data. The settings below mark its boundary; the query-only update and the same-batch lookahead are discussed in Section 4.2.

One long pass over a large corpus. On the modded-nanogpt optimization track (FineWeb, a 12-layer model, one pass) with its tuned MuonW and AdamW recipes, T^3 stays within 0.003 of MuonW at two batch sizes, while 10% more baseline steps gain 0.012. With AdamW it gains 0.015 on one seed and loses 0.004 on two further seeds, whose baselines are 0.012 better. Repeating the data 4.3 or 17 times inside these recipes does not create a gain at the validation minimum (Appendix O). A short pass behaves differently: with the 300M DCLM model and Muon trained for 750 updates, T^3 is 0.025 lower at equal updates and 0.011 at equal time on one seed, although it did not reach our 5% speedup criterion (Appendix N). This matches the early lead of Sections 3.1 and 4.3: a run that ends early keeps it, and a long single pass can close it.

Too little data per half, or a weaker regularizer. Halving the batch removes the gain in both settings we tested (Section 4.4). At 1.4B, lowering the certified weight decay from 3.2 to 1.8 costs the baseline 0.065 to 0.086, and T^3 is then level with the baseline on two seeds (-0.001), where it gains 0.016 and 0.020 at the certified weight decay (Appendix H.4).

H COMPLETE FIXED-DATA COMPARISONS

This appendix gives the complete account of the fixed-data experiments summarized in Section 3.1. It fixes the protocol, then reports the Muon comparisons on the tuning seed, the every-step design applied to the certified recipes without retuning, the same-seed comparison across four sizes, and the ensembles at every size.

H.1 PROTOCOL

Every comparison in this section trains on the same fixed corpus and is evaluated on the same held-out tokens. The corpus is the released DCLM training data of Kim et al. (2025) (Li et al., 2024), 196,608,000 tokens per pass in 750 batches of 64 sequences of 4,096 tokens under the Llama 3 tokenizer (Dubey et al., 2024); later passes repeat the corpus in a deterministic shuffled order. The validation set is 316 held-out sequences of 4,096 tokens from 1,000 documents with no overlap with training, and the reported loss is the mean next-token cross-entropy over their 1,294,020 predicted tokens. Models are Llama-architecture transformers of 150M, 300M, 600M, and 1.4B parameters (154M, 300M, 602M, and 1,431M exactly). The baseline at each size is the recipe certified by the protocol, tuned jointly in learning rate, weight decay, and pass count for that size (Table 9); The certified-recipe comparisons use AdamW (Loshchilov & Hutter, 2019) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, one percent warmup and cosine decay to zero, gradient-norm clipping at 1.0, and bfloat16 compute with float32 parameters. The separately tuned Muon comparison below retains the model, data, batch, horizon, and precision, with its own matrix and auxiliary optimizer groups.

Every number compared in this section uses the same model seed and data order as its comparator. All results reported here use model seed 41, the seed the recipes were tuned on and called the tuning seed below, and data seed 42 unless a seed range is stated, so that treatment and control differ only in the treatment. Comparisons across seeds use the same model seeds in every arm. An arm is one design run on a set of seeds, a member is one such run, a member's twin is the member of another arm trained with the same seed, and a paired gap is the loss of a T^3 member minus its baseline twin, reported as mean $\pm$ standard deviation over the seeds (negative is a gain).

H.2 INNER DIRECTION AND UPDATE SCOPE

We fix one setting for the new optimizer and mechanism comparisons: the 300M model, 16 passes, data seed 42, every-step split pairing, and a normalized temporary step of radius 0.3 on the 18 MLP matrices in layers 6–11. The primary outer optimizer is mixed Muon, selected before reading the corresponding T^3 endpoint; AdamW is an auxiliary comparison. These comparisons use tuning seed 41. Muon uses matrix learning rate 0.006 and weight decay 1.6, with auxiliary Adam learning rate

size	parameters	passes	steps	learning rate	weight decay
150M	154,147,328	16	12,000	3×10^{-3}	0.8
300M	299,649,792	16	12,000	3×10^{-3}	1.6
600M	602,457,088	8	6,000	1×10^{-3}	3.2
1.4B	1,431,373,824	8	6,000	1×10^{-3}	3.2

Table 9: The certified baseline recipes of the data-constrained protocol. Batch 64 sequences of 4,096 tokens at every size; 750 steps per pass.

0.003 and weight decay 1.6. Its matrix directions use momentum 0.95, Nesterov momentum, five Newton–Schulz iterations, and Kimi scaling. Only the auxiliary Adam group uses gradient clipping at 1.0.

Changing the inner direction. At the same 18-matrix scope and total radius 0.3, we replace the normalized support gradient with a stateless Muon-shaped direction: five quintic Newton–Schulz iterations per matrix, followed by one global normalization. The outer estimator is unchanged and there is no inner momentum or weight decay. Under outer Muon, terminal loss is 3.504430 versus 3.503228 for normalized inner, a difference of +0.001202 within the predeclared 0.003 practical margin. Under auxiliary AdamW, it is 3.542567 versus 3.522827, a deterioration of 0.019739. This single-seed, fixed-radius comparison finds no useful primary gain from the alternative inner direction; the larger AdamW effect makes the outer optimizer part of the interpretation. It does not establish an optimizer-independent advantage of either direction.

External likelihood for these direction and scope comparisons is reported in Appendix K.

Scope of the temporary update. At fixed total radius 0.3 and the primary outer Muon recipe, adapting the latter-half MLP, all MLP, or all transformer matrices gives losses 3.502605, 3.502888 and 3.505683, respectively (18/36/84 matrices). All parameters still receive the outer gradient. Neither predeclared adjacent contrast reaches the 0.003 practical margin, so we retain the smaller primary mask and do not release the conditional earlier-half control. The widest-minus-primary difference is +0.003078; this stopping decision does not assert equivalence. Wider masks reduce displacement per selected scalar at fixed total radius, so this comparison does not isolate parameter location from its allocation.

One-pass compute check. A separate nanochat pilot on a nonrepeating token stream finds no resolved compute acceleration: the selected normalized radius improves terminal equal-token BPB by only 0.000250 and is 0.006530 worse at the native baseline's measured time (Appendix N). This separate recipe does not change the fixed-data comparison above.

H.3 Every-step adaptation with the certified AdamW recipe

We apply the every-step design to the certified 300M AdamW recipe, retaining its outer learning rate, weight decay, and horizon (Table 10). Here and in the rest of the section the every-step design is split pairing with a normalized step of 0.3 on the second-half MLP matrices, a constant step norm rather than the SlowRun scripts' scheduled one. The comparison is on one seed and one data order throughout, at 300M and 16 passes, and the step norm 0.3 is the best of five on that seed, so the eight-seed comparison of Section H.5 is the unselected estimate. In this AdamW comparison, the tested normalized directions outperform the tested alternatives. With a raw step the every-step run gains 0.002 to 0.010 (3.5439 at 0.1, 3.5363 at 0.2), a sign step does nothing (3.5446 and 3.5450 at 10^{-5} and 3×10^{-5}), an Adam-preconditioned step gains 0.003 to 0.013 with a best of 3.5327 at 3×10^{-5} and no monotone trend over a thirtyfold range of inner learning rate, the outer optimizer's own update direction (its bias-corrected first moment over the square root of its second, read from the AdamW state) gains up to 0.016 at an inner learning rate of 3×10^{-4} (a tenth of an outer step) with a single peak, a random direction of the same norm (3.5433) and the normalized ascent direction (3.5482) leave the baseline where it is, the same normalized step spread over every parameter is harmful (3.5660 at 0.3, 3.9044 at 1.0), while with a normalized descent step it is better at every step norm tried (3.5237, 3.5253, 3.5213, 3.5220, 3.5249 at 0.1, 0.2, 0.3, 0.5, 0.7; flat within 0.004, best at

design (300M, 16 passes, seed 41)	inner step	loss	step cost
baseline, certified recipe	–	3.5459	1.00×
sharpness-aware minimization, best radius	ascent, $\rho = 0.01$	3.5391	1.94×
every step, split, raw	0.1	3.5439	1.06×
every step, split, raw	0.2	3.5363	1.06×
every step, split, sign	10^{-5}	3.5446	1.06×
every step, split, sign	3×10^{-5}	3.5450	1.06×
every step, split, Adam-preconditioned	10^{-5}	3.5463	1.06×
every step, split, Adam-preconditioned	3×10^{-5}	3.5327	1.06×
every step, split, Adam-preconditioned	10^{-4}	3.5426	1.06×
every step, split, Adam-preconditioned	3×10^{-4}	3.5383	1.06×
every step, split, outer-Adam momentum	3×10^{-5}	3.5427	1.06×
every step, split, outer-Adam momentum	10^{-4}	3.5373	1.06×
every step, split, outer-Adam momentum	3×10^{-4}	3.5297	1.06×
every step, split, outer-Adam momentum	10^{-3}	3.5439	1.06×
every step, split, random direction	0.3	3.5433	1.06×
every step, split, normalized ascent	0.3	3.5482	1.06×
every step, split, normalized, all parameters plastic	0.3	3.5660	1.06×
every step, split, normalized, all parameters plastic	1.0	3.9044	1.06×
every step, split, normalized	0.1	3.5237	1.06×
every step, split, normalized	0.2	3.5253	1.06×
every step, split, normalized	0.3	**3.5213**	1.06×
every step, split, normalized	0.5	3.5220	1.06×
every step, split, normalized	0.7	3.5249	1.06×
every step, split, normalized, weight decay 1.4	0.3	3.5217	1.06×
every step, consecutive, normalized	0.2	3.5238	2.00×

Table 10: Every-step T^3 on the certified 300M recipe, same seed and data order throughout. Step cost is the median optimizer-step time relative to the baseline. The sharpness-aware row is canonical SAM (same batch, perturbed gradient only) at its best radius of five; the outer-Adam momentum rows take the temporary step along the outer optimizer's own prospective update; bold marks the lowest loss.

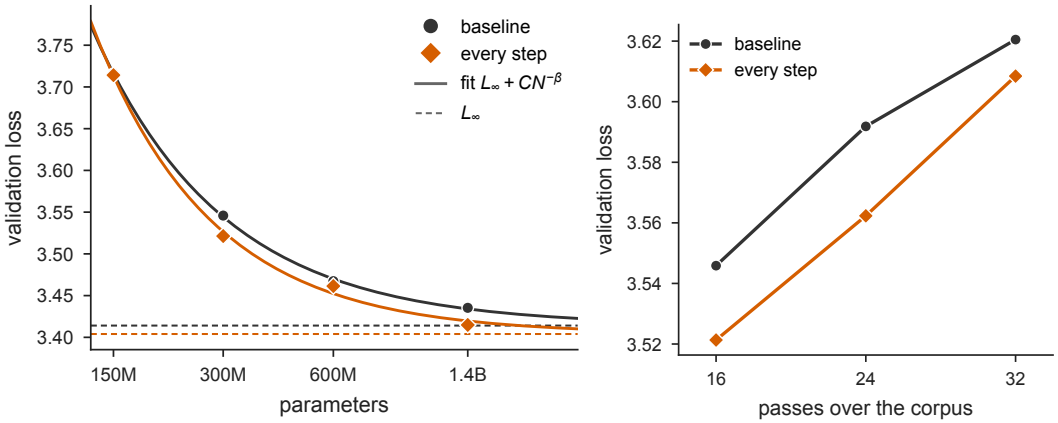
0.3) and 0.025 below the baseline trained on the same seed. Split pairing consumes one batch per step and costs 1.06× a baseline step (0.648 against 0.609 s); consecutive pairing at every step doubles the step time (1.216 s) for the same loss (3.5238), so split pairing is the every-step design of record. Sharpness-aware minimization on the same recipe, which perturbs all parameters up the gradient by a radius ρ on the same batch and trains on the perturbed gradient alone at twice the compute, gains 0.007 at its best radius ($\rho = 0.01$, 3.5391), with both neighbouring radii worse so that the optimum is bracketed by the grid, and loses 0.03 to 0.12 at 0.05 to 0.2, where the ascent step destabilizes training (Appendix I). That best gain is itself the size of the seed standard deviation, while the descent lookahead at 1.06× gains three to four times more. The every-step split design consumes exactly the baseline's gradient tokens. The certified recipe is also the best recipe for the every-step design. One coordinate-descent step around it with the every-step design in place finds no better neighbour (outer learning rate $\times\sqrt{2}$ 3.6687 and $/\sqrt{2}$ 3.5363; weight decay 1.2, 1.4, and 2.0 at 3.5244, 3.5217, and 3.5461 against 3.5213 at 1.6). The same holds at 1.4B, where a 1.5× learning rate or weight decay 1.8 costs the every-step design 0.09 to 0.11 (Section H.4).

H.4 Across sizes

On the tuning seed T^3 lowers the same-seed loss at every certified size, and the every-step design keeps a 0.02 gain at the largest one (Table 11, Figure 5a). The every-step design on the baseline recipes gains 0.025 at 300M, 0.021 at 1.4B, and 0.002 to 0.006 at 150M and 600M. The seed standard deviation of the baseline is 0.006 at 300M (eight members, Section H.5), 0.003 at 600M (four members), and 0.005 at 1.4B (eight members, Section H.6), so the 300M gap and the seed-41 1.4B gap are about four standard deviations (the eight-seed mean at 1.4B is 0.015, Section H.6), while the 150M and 600M single-seed gaps are within one 300M standard deviation. The 600M point understates the effect. Four further seeds give the every-step design a mean paired gap of

		seed 41			multi-seed gap
size	passes	baseline	every step	Δ	(seeds)
150M	16	3.7158	3.7141	-0.002	-0.005 ± 0.006 (4)
300M	16	3.5459	3.5213	-0.025	-0.014 ± 0.003 (8)
600M	8	3.4673	3.4614	-0.006	-0.012 ± 0.005 (4)
1.4B	8	3.4354	3.4148	-0.021	-0.015 ± 0.009 (8)

Table 11: Same-seed validation loss by size. "Every step" runs the SlowRun design (split halves, normalized inner step 0.3, second-half MLP plastic set) on the certified baseline recipe without tuning; Δ is its gap to the baseline on seed 41, and the last column the mean $\pm$ standard deviation of the paired gaps over the seeds in parentheses (Sections H.5 and H.6).



(a) Loss against size with $L_\infty + C\,N^{-\beta}$ fits.

(b) Pass counts.

Figure 5: **Gains at 300M and above on the tuning seed, and no benefit from more passes.** (a) Validation loss against model size for the certified baseline recipes and the every-step design (seed 41); dashed lines mark the fitted asymptotes L_∞. (b) The same-seed pass test of Section L.5 at 300M.

-0.012 ± 0.005 at 600M, twice the seed-41 value, with every pair negative, and at 150M four seeds give -0.005 ± 0.006 (three of four negative), less than two standard errors from zero (Section H.6). The multi-seed means of the every-step design, -0.014 on eight seeds at 300M, -0.012 on four seeds at 600M, -0.015 on eight seeds at 1.4B, and -0.005 on four seeds at 150M, are the paper's headline single-model numbers. The seed-41 runs of Table 11 are the points behind the scaling fit. At 1.4B the baseline recipe is also the right T^3 recipe. Raising its learning rate by $1.5\times$ or lowering its weight decay to 1.8 costs the every-step design 0.11 and 0.09 on the tuning seed. Lowering the baseline's weight decay alone costs 0.076 on that seed and 0.065 to 0.086 on three further seeds (3.5105, 3.5067, and 3.5183 against 3.4246, 3.4417, and 3.4324), and at weight decay 1.8 the T^3 gains do not reappear. On seeds 0 and 1 the every-step design is level with the weight-decay-1.8 baseline (-0.001 on both) where it gains 0.016 and 0.020 at the certified weight decay. The every-step design does not prefer the lower weight decay. In these tested configurations, adding the episode does not recover the benefit of the certified weight decay.

The direct comparison at 1.4B supports the claim; the following fits are descriptive. Following the protocol we fit $L(N) = L_\infty + C\,N^{-\beta}$ to the four points of each arm (Figure 5a). The baseline fit gives $L_\infty = 3.414$ ($\beta = 1.22$) and the every-step fit $L_\infty = 3.404$ ($\beta = 1.34$). These fits use four single-seed points and three parameters; their uncertainty is not quantified. The every-step asymptotic point estimate is lower by 0.010, which does not establish an asymptotic improvement. At the largest certified size the every-step design lowers the loss by 0.021 on the tuning seed and by 0.015 on average over eight further seeds, at $1.06\times$ the step time (1.401 against 1.320 s), on the certified recipe without tuning.

members (300M, 16 passes, seeds 0–7)	K	mean member	ensemble	D	pairwise KL
baseline	2	3.5406	3.4344	0.106	0.42
baseline	4	3.5374	3.3735	0.164	0.42
baseline	8	3.5397	3.3446	0.195	0.42
T^3 every step	2	3.5243	3.4213	0.103	0.41
T^3 every step	4	3.5235	3.3650	0.158	0.41
T^3 every step	8	3.5252	**3.3359**	0.189	0.41
mixed, baseline + every step	8	3.5304	3.3374	0.193	0.41

Table 12: Probability-averaged ensembles of the first K seeds of each arm, same seeds and data order in every arm (single members: baseline 3.5418, every step 3.5270 on seed 0). D is the disagreement term of Equation (5); the pairwise KL between members is in nats per token on the first sequence of each of four probe batches (Appendix J). The mixed ensemble takes half its members from each arm; bold marks the lowest ensemble loss.

H.5 Ensembles at 300M

The single-model gain survives ensembling. We trained eight members of each of two arms with model seeds 0 to 7 and one data order at 300M and 16 passes (the certified baseline and the every-step design on the baseline recipe), kept their final checkpoints, and evaluated probability-averaged ensembles of the first K seeds on the validation set (Table 12, Figure 6; the evaluation is in Appendix J). A member's twin is the member of another arm trained with the same seed, and a paired gap is the difference between twins, reported as mean $\pm$ standard deviation over the eight seeds. The baseline ensembles themselves are worth ten times the single-model gains in this regime. The members average 3.5397 with a seed standard deviation of 0.006 and the ensembles reach 3.4344, 3.3735, and 3.3446 at $K = 2$, 4, and 8, so the disagreement term grows from 0.106 to 0.195. The every-step members are the better single models, a paired gap of -0.0144 ± 0.0026, and keep part of that under ensembling (3.4213, 3.3650, 3.3359; gaps -0.0131, -0.0085, -0.0087) with slightly lower disagreement (0.189 against 0.195) and pairwise KL (0.41 against 0.42); a mixed ensemble of four baseline and four every-step members (3.3374) lands between the pure ones. The T^3 ensemble beats the baseline ensemble at every K.

All members above share one data order and differ only in the model seed. A control with independent data orders (model seed s with data seed $43 + s$, four baseline and four every-step members) gives the same picture: the every-step members beat their twins on every seed, a paired gap of -0.0115 ± 0.0038, their four-member ensemble is 3.3646 against 3.3703 (half the member gap) with disagreement 0.159 against 0.164, and the baseline's disagreement at $K = 4$ rounds to 0.164 with either data-order design (Appendix J). This is a descriptive comparison of two fixed four-member groups; rounded agreement does not establish that data-order variation cannot add diversity.

H.6 Ensembles at 1.4B, 600M, and 150M

At 1.4B the gain survives ensembling on eight seeds with equal disagreement, and one seed goes the other way; that run and three other every-step members, against one baseline member, show an early loss excursion that recovers within a few hundred steps (Appendix J). Eight baseline and eight every-step members (seeds 0 to 7, certified 1.4B recipe, 8 passes) give paired gaps of -0.0156, -0.0199, -0.0107, $+0.0059$, -0.0207, -0.0173, -0.0233, and -0.0192, a mean of -0.0151 ± 0.0093 (the tuning seed 41 gave -0.0205); the members average 3.4334 (sd 0.005) and 3.4183 (sd 0.008). Their ensembles are 3.3187 against 3.3342 at $K = 2$, 3.2736 against 3.2820 at $K = 4$, and 3.2409 against 3.2553 at $K = 8$, so the eight-member ensemble gap (0.0144) is 0.95 of the member gap, with the same disagreement (0.177 against 0.178) and pairwise KL (0.37). The mixed ensemble of two members from each arm is the best four-member model at 3.2708, with the largest disagreement (0.154), while the mixed eight-member ensemble (3.2479) lies between the pure ones. The single-model effect at 1.4B is therefore 0.015, three quarters of its single-seed value and not uniform across seeds, while the ensemble effect is the member effect carried through (Table 15, Figure 7).

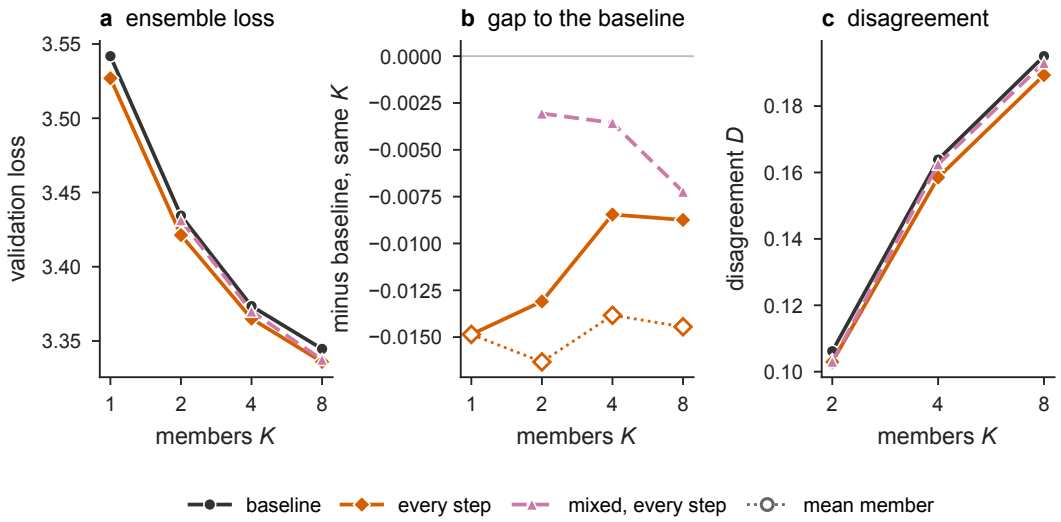


Figure 6: **The T^3 ensemble beats the baseline ensemble at every K.** Data-constrained setting, 300M, 16 passes, seeds 0 to 7. (a) Probability-averaged ensemble loss against the ensemble size K for baseline and every-step members, and the mixed ensemble with half its members from each arm (dashed). (b) Each arm minus the baseline at the same K, for the ensemble (filled markers) and for its mean member (open markers, dotted). (c) The disagreement term D against K.

At 600M the four-seed member gap is twice what the single-seed probe suggested, and the ensembles carry all of it. Four baseline and four every-step members (seeds 0 to 3, certified 600M recipe, 8 passes) give paired gaps of -0.0101, -0.0078, -0.0125, and -0.0185, a mean of -0.0122 ± 0.0046 (the tuning seed 41 gave -0.0059); the members average 3.4719 (sd 0.003) and 3.4597 (sd 0.002), so the baseline seed spread at 600M is half that at 300M and every pair goes the same way. Their ensembles are 3.3682 against 3.3780 at $K = 2$ and 3.3156 against 3.3292 at $K = 4$, an ensemble gap of 0.0137 against a member gap of 0.0122, with the same disagreement (0.144 against 0.143) and pairwise KL (0.36). The mixed ensemble of two members from each arm, 3.3217, lies between the pure ones. At 150M, where the four-seed member gap is -0.0045 ± 0.0057 (baseline members 3.7182, sd 0.004; every-step members 3.7137, sd 0.002), the ensembles are 3.6146 against 3.6156 at $K = 2$ and 3.5575 against 3.5591 at $K = 4$, a third of the member gap, with disagreement 0.156 against 0.159. Across the four sizes the four-member every-step ensemble keeps 0.35 (150M), 0.6 (300M), 1.1 (600M), and 0.8 (1.4B) of the member gap of the seeds ensembled, the eight-member ensemble 0.6 (300M) and 0.95 (1.4B), and the disagreement of the two arms never differs by more than 0.006 in these fixed member groups. The paired effects vary across seeds, including one adverse pair at 1.4B. The every-step design preserves approximately the baseline's member disagreement while improving member loss at 600M and 1.4B.

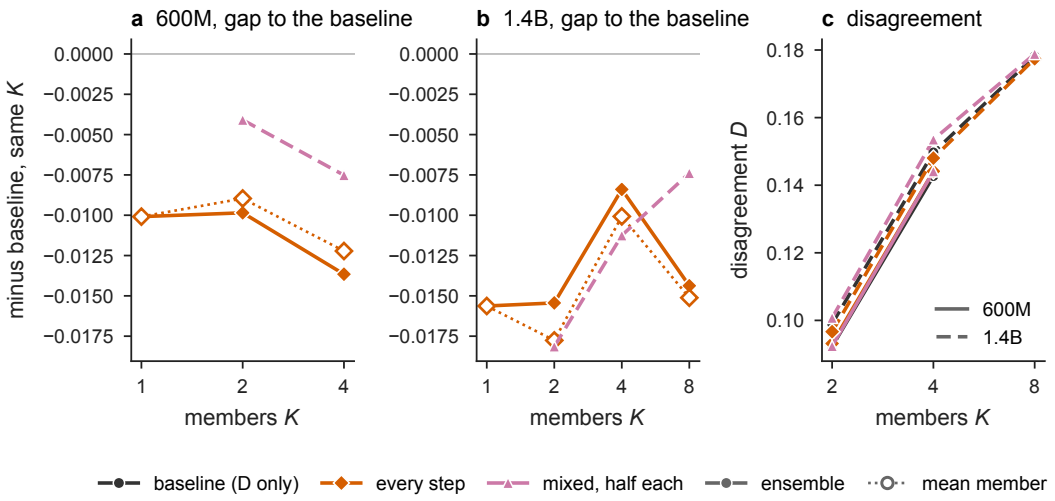


Figure 7: **The ensemble gap follows the member gap at 600M and 1.4B.** (a) 600M (seeds 0 to 3) and (b) 1.4B (seeds 0 to 7): the every-step and mixed probability-averaged ensembles minus the baseline ensemble at the same K (filled markers), and the every-step mean member minus the baseline mean member (open markers, dotted). (c) The disagreement term D against K at both sizes. The absolute losses are in Table 15.

I SHARPNESS-AWARE MINIMIZATION SWEEP

Sharpness-aware minimization sweep. Canonical SAM on the certified 300M recipe (seed 41, 16 passes): the perturbation is $\rho \nabla L / \|\nabla L\|$ over all parameters from the same batch, the update uses the perturbed gradient alone, and each step costs two forward-backward passes ($1.94\times$ the baseline step time). Validation losses are 3.5425, 3.5407, 3.5391, 3.5409, 3.5864, 3.5763, and 3.6615 at $\rho = 0.003$, 0.005, 0.01, 0.02, 0.05, 0.1, and 0.2 against 3.5459 for the baseline. The optimum is bracketed at $\rho = 0.01$: both neighbours are worse, by 0.0016 at 0.005 and by 0.0018 at 0.02, and the small-radius end returns toward the baseline as it must. Near the optimum the training curves are smooth, while $\rho \geq 0.05$ shows loss spikes of 0.5 to 0.9 between consecutive evaluations and ends with a training loss of 3.20 against 2.97. The best gain, 0.0068, is the size of the seed standard deviation on this task, and the sweep is single-seed; it bounds what SAM buys here rather than establishing it.

J ENSEMBLE EVALUATION AND MEMBER TABLES

Evaluation. Each member's final permanent checkpoint is loaded and evaluated on the validation set of Section 3.1; for an ensemble of K members the per-token probabilities are averaged (probability averaging) or the logits are averaged and renormalized (logit averaging), and the loss is the mean next-token cross-entropy of the averaged prediction. The disagreement term D of Equation (5) is the mean member loss minus the probability-averaged ensemble loss. For Tables 12 and 15, pairwise KL is averaged over ordered distinct member pairs and 16,380 valid target tokens from the first sequence of each of four probe batches. Model computation uses BF16; logits, normalized log-probabilities, and the KL reduction use FP32. Logit averaging is within 0.015 of probability averaging at every K; it is worse at 300M, 600M, and 1.4B for $K \geq 4$ and better at 150M, by up to 0.003 at $K = 4$ and 0.008 at $K = 2$.

Members at 300M. Table 14 lists every member loss. The two arms share model seeds and the data order; the every-step members use the certified recipe with the normalized step of 0.3.

Members at 600M. Table 16 lists the four members of each arm behind Table 15; the baseline and every-step arms use the certified 600M recipe (8 passes), the every-step members with the normalized step of 0.3.

Table 13: **T^3 ensembles stay below the baseline ensembles at 300M, 600M, and 1.4B.** Probability-averaged ensembles at the largest member count trained at each size, in nats per token; lower is better. Δ is T^3 minus the baseline, for the ensemble and for the mean member of the same ensemble, and D is the disagreement term of Equation (5).

		ensemble loss			mean member	disagreement D	
size	K	baseline	T^3	Δ	Δ	baseline	T^3
150M	4	3.5591	3.5575	-0.0016	-0.0045	0.159	0.156
300M	8	3.3446	3.3359	-0.0087	-0.0144	0.195	0.189
600M	4	3.3292	3.3156	-0.0137	-0.0122	0.143	0.144
1.4B	8	3.2553	3.2409	-0.0144	-0.0151	0.178	0.177

arm	s0	s1	s2	s3	s4	s5	s6	s7	mean	sd
baseline	3.5418	3.5393	3.5313	3.5370	3.5404	3.5518	3.5365	3.5392	3.5397	0.0059
T^3 every step	3.5270	3.5215	3.5218	3.5238	3.5241	3.5354	3.5224	3.5258	3.5252	0.0045

Table 14: Member losses at 300M and 16 passes (model seeds 0 to 7, data seed 42). Paired gaps to the baseline are -0.0144 ± 0.0026, negative on every seed.

size	members	K	mean member	ensemble	D	pairwise KL
1.4B	baseline	2	3.4332	3.3342	0.099	0.38
1.4B	baseline	4	3.4317	3.2820	0.150	0.37
1.4B	baseline	8	3.4334	3.2553	0.178	0.37
1.4B	T^3 every step	2	3.4154	3.3187	0.097	0.37
1.4B	T^3 every step	4	3.4217	3.2736	0.148	0.37
1.4B	T^3 every step	8	3.4183	**3.2409**	0.177	0.37
1.4B	mixed, baseline + every step	2	3.4168	3.3160	0.101	0.39
1.4B	mixed, baseline + every step	4	3.4243	3.2708	0.154	0.38
1.4B	mixed, baseline + every step	8	3.4267	3.2479	0.179	0.37
600M	baseline	2	3.4703	3.3780	0.092	0.36
600M	baseline	4	3.4719	3.3292	0.143	0.36
600M	T^3 every step	2	3.4614	3.3682	0.093	0.36
600M	T^3 every step	4	3.4597	**3.3156**	0.144	0.36
600M	mixed, baseline + every step	2	3.4664	3.3740	0.092	0.36
600M	mixed, baseline + every step	4	3.4658	3.3217	0.144	0.36
150M	baseline	2	3.7180	3.6156	0.102	0.43
150M	baseline	4	3.7182	3.5591	0.159	0.43
150M	T^3 every step	2	3.7147	3.6146	0.100	0.44
150M	T^3 every step	4	3.7137	**3.5575**	0.156	0.43
150M	mixed, baseline + every step	2	3.7162	3.6177	0.098	0.43
150M	mixed, baseline + every step	4	3.7163	3.5600	0.156	0.43

Table 15: Probability-averaged ensembles of the first K seeds of each arm at 1.4B (seeds 0 to 7), 600M, and 150M (seeds 0 to 3; 150M members run 16 passes); single members on seed 0 are 3.4246 (baseline) and 3.4090 (every step) at 1.4B, 3.4714 and 3.4613 at 600M, and 3.7152 and 3.7173 at 150M. Mixed ensembles take half their members from each arm; the best model of each size is bold.

Members at 1.4B. Table 17 lists the eight members of each arm behind Table 15; both arms use the certified 1.4B recipe (8 passes), the every-step members with the normalized step of 0.3.

Members at 150M. Table 18 lists the four members of each arm at 150M (certified recipe, 16 passes; every-step members with the normalized step of 0.3).

arm	s0	s1	s2	s3	mean	sd
baseline	3.4714	3.4692	3.4706	3.4766	3.4720	0.0032
T^3 every step	3.4613	3.4614	3.4581	3.4581	3.4597	0.0019

Table 16: Member losses at 600M and 8 passes (model seeds 0 to 3, data seed 42). Paired gaps to the baseline are -0.0122 ± 0.0046, negative on every seed.

arm	s0	s1	s2	s3	s4	s5	s6	s7	mean	sd
baseline	3.4246	3.4417	3.4324	3.4282	3.4350	3.4334	3.4354	3.4366	3.4334	0.0052
T^3 every step	3.4090	3.4218	3.4217	3.4341	3.4142	3.4161	3.4120	3.4174	3.4183	0.0078

Table 17: Member losses at 1.4B and 8 passes (model seeds 0 to 7, data seed 42). Paired gaps to the baseline are -0.0151 ± 0.0093, negative on seven of the eight seeds.

Training loss at θ against validation loss. Table 19 gives the paired gaps behind the regime comparison of Section L.2: the mean over the last 300 optimizer steps of the baseline's batch loss and of the every-step run's adaptation-half loss at θ on the same batches (same data order), next to the final validation gap.

Loss spikes at 1.4B. On the certified 1.4B recipe some seeds of both arms show one early rise of the validation loss between consecutive evaluations (steps 1200 to 3300): baseline seed 3 by 3.0 and every-step seeds 1, 2, 3, and 6 by 2.6, 0.4, 4.1, and 1.1. Training recovers within a few hundred steps in every case and the final losses are compared as they stand. No member at 150M, 300M, or 600M shows a rise above 0.003 between consecutive evaluations.

Final checkpoints. The training framework numbers completed steps from zero and does not run its end-of-training checkpoint hook with the forcing flag that would save a permanent checkpoint, so a permanent final checkpoint requires a policy that fires at step 11,999 of a 12,000-step run. The first eight baseline members were trained before this was found. Five of them keep only a temporary checkpoint from the last ten minutes of training and were finished from it to the final step under a new run, which reproduces each member's own final loss to 10^{-4}; the other three were interrupted mid-training by a scheduler event and were resumed from their last temporary checkpoint (steps 2,977, 7,890, and 2,980) to the final step with the final checkpoint kept.

K Optimizer controls and external likelihood

Setting. These comparisons retain the 299,649,792-parameter Llama, fixed 196,608,000-token DCLM pool, 12,000 updates, 16 passes, batch 64×4096, model seed 41 and data seed 42. The optimizer settings are given in Appendix H.2. Computation is BF16 with FP32 parameters.

Common external protocol. The checkpoints reported here are evaluated on the same 64 documents per domain: English AO3, computer-science arXiv, physics arXiv, BBC news, C++, Python and English Wikipedia. The 448 documents contain 411,146 scored next-token targets. Teacher-forced negative log likelihood uses identical tokenization, document order and masks across checkpoints. Pooled loss divides total NLL by scored tokens; domain means retain their own token denominators. The 95% intervals use 10,000 paired document resamples, stratified by domain, with bootstrap seed 20260915. They condition on the checkpoints and fixed domains, have no multiple-comparison correction, and do not estimate training-seed or execution variation. Hyperparameters are selected using DCLM before interpreting these scores. Every retained checkpoint is the fixed-horizon terminal checkpoint at update index 11,999.

Table 20 reports four inner-direction and scope contrasts from the common evaluation.

Interpretation. The joint transfer rule requires an absolute DCLM difference of at least .003 and an external interval excluding zero in the same direction. Only the AdamW inner-Muon deterioration

arm	s0	s1	s2	s3	mean	sd
baseline	3.7152	3.7207	3.7230	3.7140	3.7182	0.0043
T^3 every step	3.7173	3.7121	3.7132	3.7123	3.7137	0.0024

Table 18: Member losses at 150M and 16 passes (model seeds 0 to 3, data seed 42). Paired gaps to the baseline are -0.0045 ± 0.0057, negative on three of the four seeds.

size	passes	pairs	training gap at θ	validation gap
150M	16	2	$+0.000$	-0.004
300M	16	6	$+0.014$	-0.013
300M	24	1	-0.002	-0.030
300M	32	1	$+0.005$	-0.012
600M	8	4	-0.025	-0.012
1.4B	8	4	-0.037	-0.020

Table 19: Every-step minus baseline, mean over the listed pairs (the 24- and 32-pass rows are the tuning-seed pass test). The 300M rows at 16 passes pool two members with the shared data order and the four data-order members.

satisfies it among the four contrasts reported here. The other three fail the DCLM margin or have an external interval crossing zero; failure of this joint rule does not mean every external interval crosses zero.

Inner Muon improves external loss by .012699 under outer Muon despite its small adverse DCLM difference. Repeated identical-checkpoint forward evaluations in the original common evaluation match exactly per document. Consistent evaluation does not establish bitwise reproduction of training trajectories, and the primary recipe is not reselected from these external rankings.

The older adapted-gradient response grid remains excluded after failing its repeated-reference check; passing forward-loss checks does not repair that failure.

candidate minus baseline	DCLM Δ	external Δ	document 95% interval
Inner Muon minus normalized (AdamW)	+.019739	+.116540	$[+.093010, +.141669]$
Inner Muon minus normalized (Muon)	+.001202	-.012699	$[-.020950, -.004553]$
All MLP minus latter-half MLP	+.000283	+.002367	$[-.004570, +.009323]$
Transformer matrices minus all MLP	+.002795	+.013364	$[+.004469, +.022495]$

Table 20: Reported pooled external contrasts. Negative differences favor the candidate. Scope rows use outer Muon. Intervals measure document uncertainty at retained checkpoints, not variation across independently trained models.

L Additional mechanism probes

L.1 Probe endpoints on the tiny track

Table 22 lists every tiny-track probe of Section 4 on both seeds. We report the two endpoints the tiny recipe evaluates, the exponential moving average of the weights and the checkpoint average; the final checkpoint alone is not listed. Diagnostic arms are compared with the record script run with the same diagnostics on the same seed, and each paired condition with the record script under that condition. In the 300M fixed-data setting, a double-batch pair with the learning rate scaled by $\sqrt{2}$ diverged in both arms (baseline 3.5707, T^3 3.6821) and is not used; the pair at the recipe's learning rate gives 3.5588 against 3.5458.

We measure gradient alignment and training loss, then vary batch size, episode period, pass count, and the outer recipe. These observations constrain possible explanations without identifying a unique mechanism for the normalized first-order update.

L.2 Gradient alignment and training loss

On the tiny SlowRun track we logged, every 10 steps of training, the cosine between the gradients of the two halves of the current batch over the Muon matrices, computed at the same weights in two backward passes. The MLP matrices moved by the temporary update are a subset of this probe's parameter scope. We measured this quantity for the record script and for every-step T^3 in the tiny-track configuration (MLP matrices, normalized step 0.5), on two seeds, 284 paired points per seed (Figure 8). These are separate runs with the logging on, and their final losses differ from Table 24 by run-to-run noise. The T^3 runs are better by the benchmark's margin (3.2933 against 3.3088 and 3.2951 against 3.3064), and the alignment is the same in both seeds to within 0.005. The mean cosine is 0.333 against 0.337 and 0.337 against 0.342; it falls from 0.58 to 0.60 in the first third of training to 0.11 in the last third in every run; the paired difference is -0.005 ± 0.075 (mean $\pm$ standard deviation over the 284 points) with 51 to 52 percent of points positive. The stochastic half-batch gradient norm is 11 percent higher under T^3 in both seeds (0.326 against 0.293 and 0.327 against 0.295). This norm does not estimate $\|\nabla L\|$ directly, and the measured cosine does not determine the projection in Proposition 1. The observations therefore do not exclude an agreement effect.

On seed 42, the final smoothed online training loss is 2.9139 under T^3 against 2.9249 for the baseline, while validation at the raw final checkpoint is 3.3448 against 3.3606. Validation using exponential-moving-average weights is 3.3136 against 3.3279, and checkpoint averaging gives the 3.2933 against 3.3088 reported above. The mean of the two diagnostic half-batch losses over the last third of training is 3.1567 against 3.1761. These training statistics average different batches and weights over time, so subtracting them from a terminal validation loss does not produce a matched train–validation gap. Training and validation losses both improve here; this does not establish the quadratic mechanism at scale. The direction control of Section 4.2 also holds in single runs of the submitted one-hour SlowRun configuration on seed 42: the descent step gives 3.1811, a random direction 3.1998, and the ascent step 3.2035, against 3.2005 for the record script. In the data-constrained setting at 300M on the certified recipe, a random temporary step of norm 0.3 gives 3.5433 and the ascent step 3.5482 against 3.5459 for the baseline, while the descent step gives 3.5213 (Table 10).

Training loss across regimes. The data-constrained runs let us repeat the training-loss comparison at four sizes, using the mean over the last 300 steps of the baseline's batch loss and of the every-step

Table 21: **Only a nonzero descent step reproduces the gain.** Losses in nats per token; each panel is one seed. (A) Direction of the temporary step, fixed-data 300M recipe, tuning seed, radius 0.3. (B) One-hour SlowRun recipe at a fixed 2,097 updates, seed 42; the checkpoint ensemble averages the checkpoints after passes 9 to 11.

(A) Direction, fixed-data 300M	
baseline	3.5459
T^3, normalized descent	**3.5213**
random direction, same length	3.5433
normalized ascent, same length	3.5482

(B) Displacement, SlowRun one hour		
configuration	final checkpoint	checkpoint ensemble
record script	3.2382	3.2011
zero control (radius 0)	3.2383	3.1993
T^3, radius 0.3	**3.2318**	**3.1903**

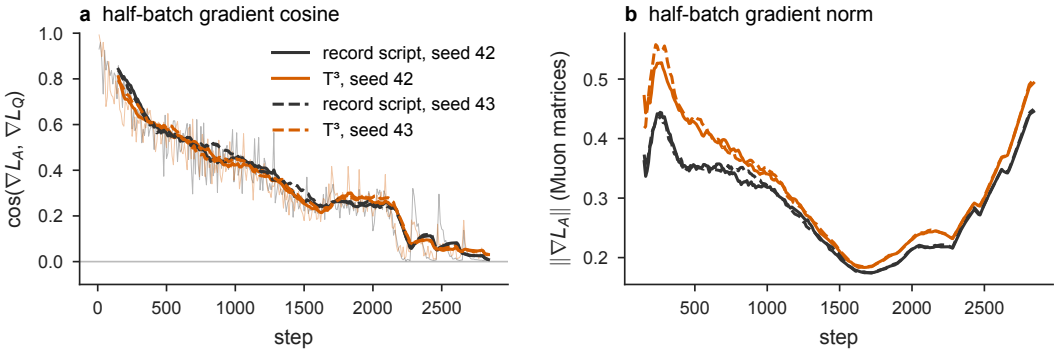


Figure 8: **The measured global cosine is similar along the two trajectories.** Tiny track, seeds 42 (solid) and 43 (dashed). (a) Cosine between the gradients of the two halves of each batch over the Muon matrices for the record script and every-step T^3, as a 15-point moving average of measurements every 10 steps; thin lines are the raw seed-42 values. (b) The half-batch gradient norm, with the same smoothing. This probe covers all Muon matrices, a larger set than the plastic MLP matrices, and does not measure the population-gradient projection directly.

run's adaptation-half loss at θ on the same batches. At 8 passes the every-step design lowers the training loss by more than the validation loss, as on the tiny track: by 0.025 against 0.012 at 600M (four pairs) and by 0.037 against 0.020 at 1.4B (four pairs). At 16 passes the picture changes: at 300M the training loss at θ is level or slightly higher (six pairs, mean $+0.014$) while the validation loss falls by 0.013, and at 150M both gaps are within noise (two pairs). At 300M and 16 passes the baseline's terminal validation loss exceeds its final-window online training average by about 0.55. The lower validation loss with similar online training loss under T^3 is consistent with reduced memorization, but these statistics do not establish that interpretation. At 8 passes both statistics improve; the adapted point's loss on the held-out half exceeds the loss at θ by 0.001 to 0.007 in every run. The training and validation changes depend on the regime (Table 19); neither pattern alone separates optimization effects from regularization. The online training averages are also not a fixed-checkpoint evaluation of the full training corpus.

L.3 Batch-size control

On one seed of the tiny track, the gain disappears under the tested half-batch recipe. A natural null hypothesis for a step that adapts on one half of the batch and tests on the other is that it acts like a half-size batch. On the tiny track (seed 42) we trained the record script and every-step T^3 at half the microbatch size for twice the steps (the same tokens) with every learning rate scaled by $1/\sqrt{2}$. The half-batch baseline is almost as good as the full-batch one (3.3129 against 3.3088), so

Table 22: **Every tiny-track probe on both seeds.** Gain is the baseline's loss minus the run's, in nats per token, so positive values favor the run, at the exponential-moving-average (EMA) endpoint and at the checkpoint average (ckpt); λ is the final largest Hessian eigenvalue on the MLP matrices where it was measured. Diagnostic arms compare with the record script run with the same diagnostics; the paired conditions compare with the record script under the same condition, whose EMA loss is given in parentheses (seeds 42 / 43). Generated by `scripts/tab_probe_endpoints.py`.

	seed 42			seed 43		
run	EMA	ckpt	λ	EMA	ckpt	λ
Record script with diagnostics: EMA 3.3266 / 3.3242, ckpt 3.3078 / 3.3045, λ 4.44 / 5.34						
T^3, split batch, radius 0.5	+0.0115	+0.0132	1.51	+0.0080	+0.0081	1.59
within-document split	+0.0121	+0.0119	1.50	+0.0105	+0.0098	1.51
full batch tested on itself, fresh mask	+0.0013	+0.0110	2.35	−0.0022	+0.0025	5.10
full batch tested on itself, same mask	−0.0028	+0.0020	1.83	−0.0104	−0.0054	0.20
query after a step on its own gradient	−0.0147	−0.0145	19.31	−0.0167	−0.0162	23.03
same, fresh dropout mask	−0.0122	−0.0119	4.22	−0.0155	−0.0148	13.54
random direction	+0.0003	+0.0008	6.79	−0.0011	−0.0009	5.83
ascent direction	−0.0022	−0.0018	1.00	−0.0044	−0.0057	0.96
parallel part of the correction only	+0.0014	+0.0042	2.87	−0.0014	0.0000	3.21
perpendicular part only	+0.0095	−0.0048	1.87	+0.0017	−0.0163	1.85
radius 0.25	+0.0063	+0.0080	3.29	+0.0051	+0.0062	3.35
radius 1	+0.0153	+0.0158	0.82	+0.0105	+0.0085	0.85
radius 2	+0.0137	+0.0135	0.43	+0.0122	+0.0083	0.40
episodes in the 1st fifth only	+0.0045	+0.0065	3.99	+0.0010	+0.0019	4.98
episodes in the 2nd fifth only	+0.0023	+0.0044	4.00	+0.0039	+0.0044	4.35
episodes in the 3rd fifth only	+0.0021	+0.0045	3.39	+0.0018	+0.0025	3.72
episodes in the 4th fifth only	+0.0004	+0.0012	2.61	+0.0013	+0.0021	2.43
episodes in the 5th fifth only	−0.0001	−0.0029	1.39	+0.0009	−0.0027	1.54
episodes in the first half only	+0.0083	+0.0091	3.09	+0.0058	+0.0059	2.96
episodes in the second half only	+0.0039	+0.0049	1.63	+0.0034	+0.0026	1.45
Paired conditions (T^3 minus the record script under the same condition)						
half batch, twice the steps (3.3157 / 3.3125)	−0.0006	−0.0021	3.42	−0.0037	−0.0033	2.86
recipe batch, no diagnostics (3.3281 / 3.3249)	+0.0117	+0.0140	–	+0.0065	+0.0061	–
twice the batch, half the steps (3.3927 / 3.3939)	+0.0138	+0.0168	–	+0.0132	+0.0157	–
four times the batch (3.4960 / 3.4969)	+0.0003	+0.0045	–	+0.0020	+0.0062	–
learning rate $\times 0.5$ (3.3909 / 3.3929)	+0.0090	+0.0150	–	+0.0110	+0.0160	–
learning rate $\times 2$ (3.3603 / 3.3612)	+0.0136	+0.0128	–	+0.0124	+0.0130	–
no dropout (3.3398 / 3.3371)	+0.0162	+0.0183	–	+0.0114	+0.0148	–
no weight decay (4.0981 / 4.0922)	+0.0149	+0.0144	–	−0.0060	−0.0041	–

halving the batch is nearly free here, and yet the T^3 gain collapses from 0.0155 at full batch (3.2933 against 3.3088) to 0.0010 at half batch (3.3119 against 3.3129). Halving the baseline batch under this learning-rate rule does not reproduce the full-batch T^3 gain. The collapse is consistent with sensitivity to gradient noise, but the experiment also changes step count and learning rates and does not isolate the cause. These are the checkpoint-average endpoints of an earlier pair of runs. The later batch sweep of Section 4.4, with two seeds and the EMA endpoint, repeats the collapse at half the batch (-0.0006 and -0.0037) and adds twice and four times the batch (Table 22).

L.4 Episode period

The gain grows as episodes become more frequent (Figure 9; one seed per setting). On the one-hour SlowRun track, with a normalized step of 0.3 on all Muon matrices, the gain is 0.002 to 0.003 at periods 16, 8, and 4 and grows as the period shrinks below that (0.005 at period 2 and 0.009 at period 1). In the data-constrained setting, a 300M sweep holds split pairing, the normalized step of 0.3, and the second-half MLP plastic set fixed. Periods $1, 2, 4, 8$ give validation losses 3.5228, 3.5288, 3.5337, and 3.5424 on seed 41 at 16 passes, against the historical same-seed baseline of 3.5459. The new

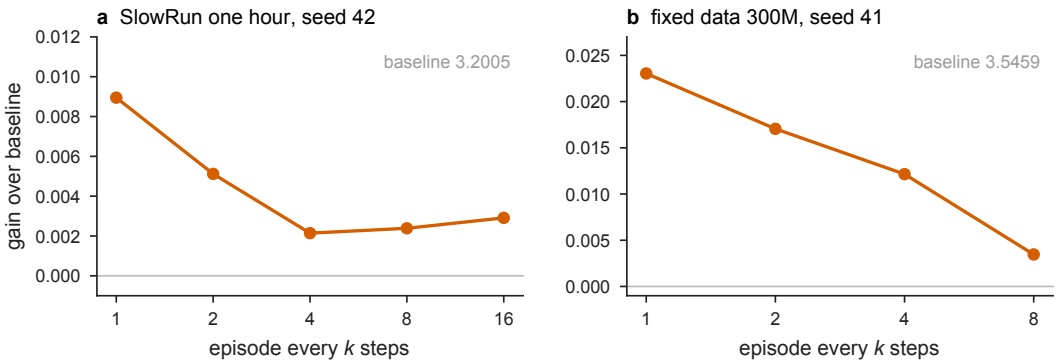


Figure 9: **The gain grows with episode frequency.** Gain over the same-seed baseline (baseline minus run, validation loss) against the episode period k. (a) SlowRun one-hour track (1.4B, normalized step 0.3 on all Muon matrices, split pairing, seed 42; baseline 3.2005). (b) Data-constrained 300M at 16 passes, seed 41 (baseline 3.5459), with a normalized step of 0.3 and split pairing, holding the outer recipe and second-half MLP plastic set fixed.

period-1 run differs from the earlier 3.5213 by 0.0015. This monotone single-seed result supports a frequency effect for the normalized design within the data-constrained setting; it does not establish that the temporary step's magnitude alone explains the difference, and the $1/k$ coefficient in a local objective expansion does not imply a linear dependence of final validation loss on frequency.

L.5 Pass-count control

T^3 does not make the tested extra passes worthwhile (one seed at 300M). On the same seed and data order at 300M, the baseline gives 3.5459, 3.5918, and 3.6205 at 16, 24, and 32 passes, and the every-step design 3.5213, 3.5623, and 3.6085 (Figure 5b). Both arms lose 0.04 to 0.09 to the extra passes, far more than T^3 gains; the every-step design keeps its lead at 24 passes (-0.030) but is not better at 24 passes than at 16. The weight-decay control at 1.4B (Section 3.1) points the same way. Below the certified weight decay the every-step design loses its gain on two seeds. In these tested settings T^3 does not compensate for the reduced weight decay, and neither extra repetition nor lower weight decay improves the selected result. These controls do not exclude regularization effects or locate a global optimum; fewer passes and higher weight decay were not jointly tuned.

L.6 Outer-recipe neighbours

At 300M we reran the four outer neighbours of the certified recipe (learning rate $\times\sqrt{2}$ and $/\sqrt{2}$, weight decay 1.2 and 2.0) with the every-step design at step norms 0.1 and 0.7, alongside the 0.3 neighbours of Section 3.1 (Table 23). The certified learning rate and weight decay are the best or tied within 0.003 at every step norm: raising the learning rate costs 0.08 to 0.15 and lowering it 0.01 to 0.02, weight decay 2.0 costs 0.009 to 0.036, and weight decay 1.2 costs $+0.005$, $+0.003$, and -0.001 at step norms 0.1, 0.3, and 0.7. The one trend is that the loss becomes flatter over the tested weight-decay values as the step norm grows. This local grid finds no clear improvement over the certified recipe. It supports trying the inner step as an addition to an already tuned recipe, but does not prove separability of the optima or rule out regularization.

L.7 Normalization and correction scope at a shared state

We inspect how inner normalization and the destination of the outer correction affect one update in the 300M Muon recipe. Five 80-update runs use model seed 41, data seed 42, 32/32 support/query halves, and sequence length 4,096. Temporary adaptation acts on the same 18 MLP matrices in layers 7–12. We retain their support gradients, inner directions, and parameters before and after updates 0 and 79. The comparison at update 0 has identical batches, random keys, retained initial parameters, and support gradients in all five runs. These short runs use their own cosine schedule, with FP32

step norm η	certified (lr 3×10^{-3}, wd 1.6)	lr $\times\sqrt{2}$	lr $/\sqrt{2}$	wd 1.2	wd 2.0
0.1	**3.5237**	3.6006	3.5424	3.5287	3.5596
0.3	**3.5213**	3.6687	3.5363	3.5244	3.5461
0.7	3.5249	3.6418	3.5352	**3.5236**	3.5336

Table 23: Every-step T^3 (split pairing, normalized step on the second-half MLP) on the certified 300M recipe and its four outer neighbours, at three step norms; seed 41 (baseline without the episode 3.5459). Bold marks the lowest loss per row; at 0.7 the two bold-candidate entries differ by 0.001, inside the 0.003 margin.

parameters and BF16 compute; their update magnitudes cannot be transferred to the 12,000-update training runs.

Let a, b, and c denote the support, unadapted-query, and adapted-query gradients, respectively. The field sent to the outer optimizer is

$$q_M = \frac{a+b}{2} + M\frac{c-b}{2},$$

where M retains the correction on all parameters, all MLP parameters, or non-MLP parameters. The ordinary gradient remains present on every parameter. The zero-displacement control uses radius zero; the other arms use radius 0.3. Three positive arms jointly normalize the support gradient on the 18 plastic matrices and vary M. A fourth normalizes each plastic matrix by its Frobenius norm, then jointly normalizes their concatenation, with the correction applied everywhere. All norms use a floor of 10^{-12}. The outer optimizer and its state initialization are shared across arms.

Matrix normalization changes the allocation of the inner displacement (Figure 10). At the shared initial state, each matrix receives 5.556% of the concatenated direction's squared norm, whereas global normalization assigns between 1.717% and 16.039%. The concatenated directions have cosine 0.936377 and relative L_2 difference 35.672%, even though their within-matrix orientations agree. Their actual persistent parameter changes have cosine 0.999295 and relative L_2 difference 3.756% on the retained coordinates. We compute these changes by subtracting saved FP32 parameters in FP64; separately retained optimizer updates give the same comparison to the reported precision. The smaller update contrast cannot be attributed specifically to Muon: the query-gradient response, averaging with the support gradient, and shared weight decay also intervene between the inner direction and the final update.

The receiving masks have their intended local effect at this initial state. On the retained MLP matrices, applying the correction only to non-MLP parameters produces exactly the zero-control change. Applying it only to MLP parameters produces exactly the all-parameter change. These equalities concern 37,158,912 retained coordinates, not the full model. At update 79 the arms have different parameters and optimizer histories, so their differences no longer isolate a shared-state effect. These measurements establish a local distinction between normalization rules and check the implementation of correction masks. They neither rank their generalization performance nor establish a mechanism for the full-training results.

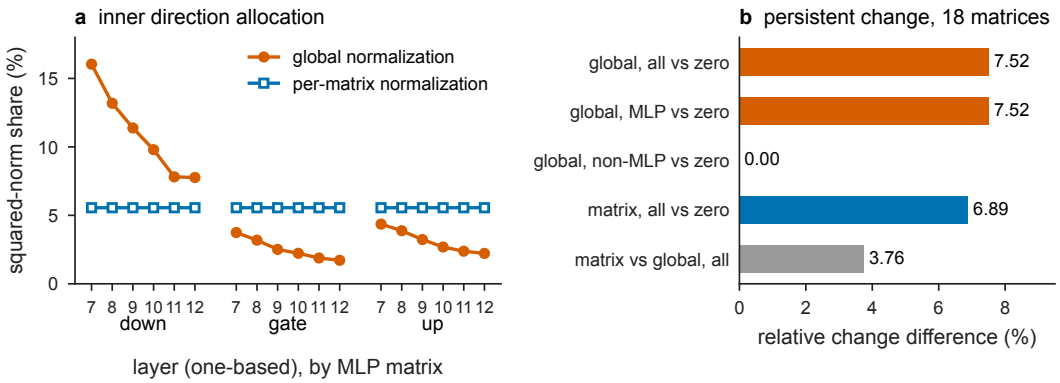


Figure 10: **Normalization changes inner allocation and the resulting update.** Measurements at a shared initial state of the 300M Muon recipe, on all 18 retained MLP matrices. (a) Each matrix's share of the squared inner-direction norm under global and per-matrix normalization. (b) The norm of the difference between actual persistent parameter changes, divided by the reference change norm named in each comparison. The coordinates are descriptive measurements from one initialization, not independent runs or estimates of training-seed uncertainty.

L.8 Compute precision and local adaptation episodes

We test whether the measured response to temporary adaptation depends on network compute precision at one fixed state of the 300M Muon recipe. We freeze the parameters and optimizer state after 1,501 updates on seed 41, and reuse the same batch of 64 sequences. The global episode adapts on 32 support sequences and evaluates 32 query sequences. The local construction divides each half into eight groups of four sequences, adapts separately within each support group, and averages the corresponding support/query gradients. Both constructions use the latter-half MLP mask and radius 0.3. Each temporary update moves in the negative support-gradient direction, normalized jointly over the selected coordinates to unit Euclidean norm with a norm floor of 10^{-12}. Each episode contributes half its unadapted support gradient and half its adapted query gradient to the outer field; the local construction averages its eight episode fields with weight $1/8$ each. We repeat the comparison with BF16 and FP32 network computation. Parameters remain FP32, and the configured output-dtype policy remains BF16. The outer optimizer's internal precision is unchanged. The latter policy is therefore not a full-FP32 reference.

The stages are the gradient field, the matrix input and output of Muon's Newton–Schulz transformation, and the actual change in persistent parameters. The matrix stages cover Muon matrices; the gradient and applied-update stages cover all trainable coordinates. At each stage, let $v_C(\rho)$ be its stored vector for construction $C \in \{\text{local}, \text{global}\}$ and radius ρ. The adaptation response is $\Delta_C = v_C(0.3) - v_C(0)$. We call $d = \|v_{\text{local}}(0) - v_{\text{global}}(0)\|_2$ the zero drift in Figure 11. All norms are Euclidean norms over the stated coordinates; panel (c) reports $\|\Delta_{\text{local}}\|_2/d$.

The local/global gradient-response norm ratio is 0.48117 under BF16 computation and 0.48134 under FP32 computation. The zero-radius gradient discrepancy falls from 8.0060×10^{-4} to 2.0239×10^{-8}. Thus the smaller local response persists under this precision intervention. After the outer update, the cosine between the local and global responses, computed separately within each precision policy, changes from 0.72060 to 0.87251. The local applied response is 1.83 and 22.26 times the corresponding zero-radius discrepancy, respectively. This downstream comparison is precision-sensitive even though the gradient-response norm ratio is stable.

These measurements establish neither a numerical error bound nor a cause of the full-training validation difference. We did not measure vector cosines across the two precision policies. As a separate control, scaling the local zero-radius gradient field to the norm of the local radius-0.3 gradient field leaves a discrepancy equal to about 87.7% of the local gradient response under either policy. The stored adaptation response is therefore not reproduced by this scalar rescaling, but its usefulness for generalization remains untested by this fixed-state probe.

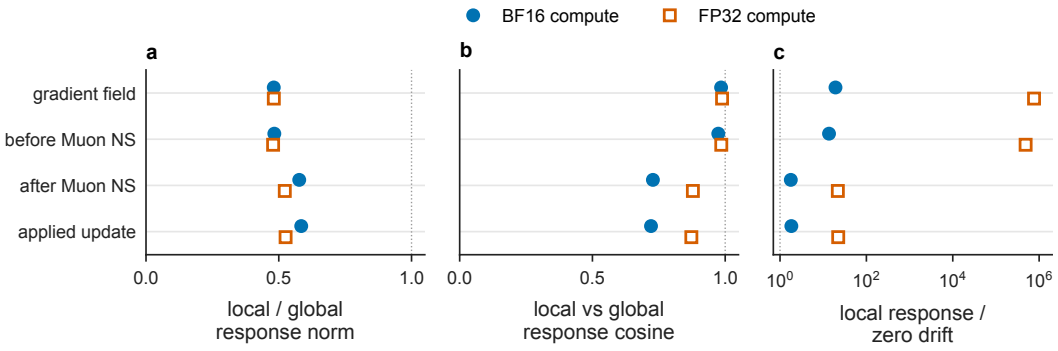


Figure 11: **The gradient-response magnitude is stable under a precision change; the downstream comparison is sensitive.** One fixed state and batch, with radius 0.3 and its own zero-radius control for each episode construction. (a) Local/global response norm ratio. (b) Cosine between the local and global responses within each precision policy. (c) Local response norm divided by the local/global zero-radius discrepancy (logarithmic axis). NS denotes Newton–Schulz: its two stages cover Muon matrix coordinates, whereas the gradient and applied-update stages cover all trainable coordinates. The output-dtype setting and outer-map precision remain unchanged. Points are deterministic summaries of these stored computations, not independent training runs or uncertainty estimates. The full record also retains optimizer-produced updates and parameter-rounding differences.

L.9 Synthesis

The descent direction, batch size, and episode frequency affect the observed gain, while the tested extra passes and outer-recipe neighbours do not improve the selected result. The measured global half-batch cosine stays similar even where training and validation losses both fall. These findings are compatible with an optimization effect from evaluating the query gradient at a nearby point. They do not identify the quadratic damping mechanism in the normalized adaptive-optimizer runs, and they do not exclude agreement or regularization effects. The empirical gains stand independently of this unresolved mechanism question.

M Complete SlowRun results and diagnostics

This appendix reports the SlowRun training comparisons, diagnostic sweeps, and ensemble evaluations, with detailed tables and pass-indexed figures. References to records and submitted entries refer to that campaign's historical scripts. The compact main comparison is in Section 3.2; the canonical ensemble decomposition is Equation (5).

M.1 Setting

T^3 added to the evaluated record scripts produced improved historical entries on both fixed-time tracks of NanoGPT SlowRun (qlabs-eng, 2026), a benchmark that fixes the data and the wall-clock budget and ranks static checkpoints by validation loss. The one-hour track fixes 100 million FineWeb tokens and 60 minutes of training time; the tiny track fixes 15 minutes and a model of about 300M parameters; an unlimited-compute track exists as well. Both fixed-time tracks rank static checkpoints by the loss on a fixed 10 million token validation split, and both had published records set by tuned scripts (3.195 at one hour, 3.307 on the tiny track). We take each record script (on the one-hour track a 1.4B transformer trained for eleven passes with Muon on the matrices and AdamW elsewhere (Jordan et al., 2024a), with stochastic-weight-averaging cycles and a three-checkpoint probability average) and add Algorithm 1 to every optimizer step. The plastic set is all Muon matrices on the one-hour track and the MLP matrices on the tiny track; the step norm follows the learning-rate schedule with a peak of 0.5 on the one-hour track and is a constant 0.5 on the tiny track; the two halves are the micro-batches the script already processes, so passes, schedules, optimizer state initialization and update rules, and evaluation are those of the record script. The submitted scripts also halve the

track, configuration	seed	val loss	timed min	same-seed baseline
one hour, first every-step configuration	42	3.1895	59.4	3.2005
one hour, first every-step configuration	43	3.1911	59.6	3.2013
one hour, first every-step configuration	44	3.1911	59.3	3.2034
one hour, submitted script	42	**3.1814**	59.2	3.2005
one hour, submitted script	43	3.1830*	59.3	3.2013
one hour, submitted script	44	3.1818	59.3	3.2034
one hour, submitted script	45	3.1824	59.0	3.2026
one hour, published record	–	3.195	59.0	–
tiny, submitted script	42	3.2954*	14.2	3.3095
tiny, submitted script	43	3.2958	14.6	3.3046
tiny, submitted script	44	**3.2939**	14.3	3.3070
tiny, submitted script	45	3.2984	14.3	3.3101
tiny, published record	–	3.307	14.1	–

Table 24: SlowRun one-hour track (cap 60 timed minutes) and tiny track (cap 15 minutes); losses after the script's checkpoint average, one row per run. Starred runs are the historical submitted entries and bold marks the lowest reported loss per track. The seed-45 one-hour baseline runs the submitted script with the episode off and the record script's regularizers. The comparisons use the baseline runs listed in this table.

one-hour recipe's dropout and remove its stochastic depth, additional changes evaluated jointly with the episode. Every T^3 run is reported next to the record script run with the same seed.

M.2 Historical submitted entries

T^3 lowers the loss on every listed seed of both tracks and produced the historical entries (Table 24). On the one-hour track the episode alone, on the unchanged recipe, gains 0.011 on seed 42 (3.1895 against 3.2005) and 0.010 and 0.012 on seeds 43 and 44; the submitted script reaches 3.1814 to 3.1830 on seeds 42 to 45 against 3.2005 to 3.2034 for the record script on the same seeds, a gain of 0.018 to 0.022, and the historical submitted entry is the seed-43 run, 3.1830 at 59.3 timed minutes. On the tiny track the submitted script reaches 3.2939 to 3.2984 on seeds 42 to 45 against 3.3046 to 3.3101, and the historical submitted entry is the seed-42 run, 3.2954 at 14.2 minutes. The extra work is the norm, the temporary step, and the restore, about 12 ms of a 1.5 to 1.9 s step on the one-hour track. The first every-step configuration (a constant step norm of 0.3, the recipe's regularizers unchanged; 3.1895 on seed 42) and the submitted script (step norm scheduled with a peak of 0.5, the recipe's dropout halved and its stochastic depth removed) are the configurations of Table 24.

M.3 Where the gain comes from

T^3 has lower loss at many measured intermediate passes, and the reported gain appears in the checkpoint average. Figure 12(a) tracks the validation loss after each pass relative to the seed-42 baseline for the first every-step run with the batch-averaged direction (3.1915, one seed). That run is 0.09 below the baseline after the first pass, 0.04 after the second and third, 0.02 through the middle, 0.005 to 0.008 below at passes nine and ten, and 0.003 above at the last pass (3.2413 against 3.2383). The evaluated benchmark quantity is the average of the last three checkpoints, and that average gains 0.050 over the final checkpoint for the T^3 run and 0.038 for the baseline; the run's 0.009 gain is that 0.012 difference less its 0.003 final-pass deficit.

Only the every-step form does this on this track (seed 42 throughout this subsection unless stated). Episodes every 16, 8, or 4 steps gain 0.002 to 0.003, every 2 steps 0.005, and every step 0.009 with the batch-averaged direction and 0.010 to 0.012 with the local one on three seeds (0.004 and 0.007 for the batch-averaged direction on seeds 43 and 44; Figure 12(b,c)). Episodes at every step but only in the first half of training keep 0.007 of the 0.011, and only during the averaging phase 0.003, so the gain accrues throughout with the larger share early.

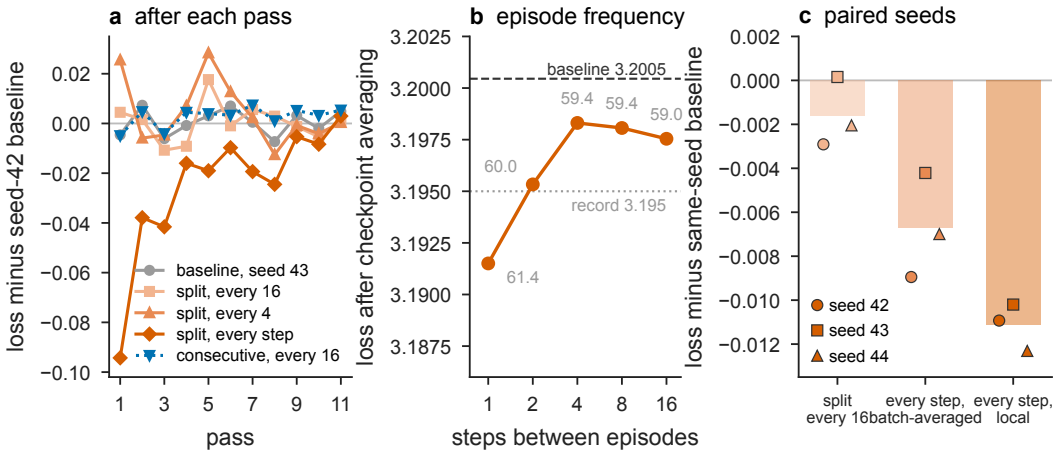


Figure 12: **The every-step episode lowers loss at many measured intermediate passes.** One-hour track. (a) Validation loss after each pass minus the seed-42 baseline, for a second baseline seed, the split episode every 16, 4, and 1 steps, and the consecutive episode every 16 steps (seed 42). (b) Loss after checkpoint averaging against the number of steps between split episodes, seed 42; labels give the timed minutes, and the lines mark the seed-42 baseline and the record at submission (3.195). (c) Loss minus the same-seed baseline on seeds 42 to 44 (markers) and their mean (bars), for split episodes every 16 steps and every step with the batch-averaged or the local direction.

members (tiny track)	K	mean member	ensemble	D	pairwise KL
baseline seeds 42,43 / 44,45	2	3.3070 / 3.3084	3.2493 / 3.2507	0.0578 / 0.0577	0.229 / 0.228
T^3 seeds 42,43 / 44,45	2	3.2956 / 3.2962	**3.2379** / **3.2384**	0.0577 / 0.0579	0.231 / 0.232
baseline seeds 42–45 / 46–49	4	3.3077 / 3.3086	3.2186 / 3.2193	0.0892 / 0.0893	0.229 / 0.230
T^3 seeds 42–45 / 46–49	4	3.2959 / 3.2959	**3.2069** / **3.2070**	0.0891 / 0.0890	0.231 / 0.233
baseline 42,43 + T^3 44,45	4	3.3017	3.2116	0.0901	0.233
baseline seeds 42–49	8	3.3082	3.2022	0.1059	0.229
T^3 seeds 42–49	8	3.2959	**3.1901**	0.1058	0.232

Table 25: Probability-averaged ensembles of T^3 members against record-script members on the tiny track, fixed 10M-token validation split; D is the disagreement term of Equation (38) and the pairwise Kullback–Leibler (KL) divergence is averaged over member pairs on eight probe batches.

The choices that matter are the pairing and the direction. Split pairing works; consecutive pairing, which adapts on the current batch and tests on the next, is within -0.001 to $+0.009$ of the baseline at every setting tried here (3.1948 at every step, in twice the time). Step norms of 0.1 to 0.3 work (3.1931 and 3.1915 at every step) and 1.0 hurts (3.2088). All Muon matrices and the MLP matrices alone both work (3.1915 and 3.1912), and the local direction is at least as good as the batch-averaged one (3.1895 against 3.1915 and 3.1935 for two implementations of the batch-averaged direction) and 30 ms per step cheaper. At the submitted setting, a random direction of the same norm gives the unchanged-script loss (3.1998 against 3.2005), the ascent direction of sharpness-aware minimization is worse than the unchanged script (3.2035), and the descent direction gives 3.1811. These controls favor descent in the tested recipe; they do not identify a unique mechanism. Configurations that do not work at equal passes, with their loss relative to the same-seed baseline (positive is worse), are the post-adaptation gradient alone ($+0.021$), reusing it as the next step's gradient so that episodes are free ($+0.012$), temporary steps shaped like the Muon update ($+0.001$ to $+0.008$), and every consecutive-pairing arm of a seventeen-arm sweep over step geometry, norm, plastic subset, schedule, and weight decay (-0.001 to $+0.009$).

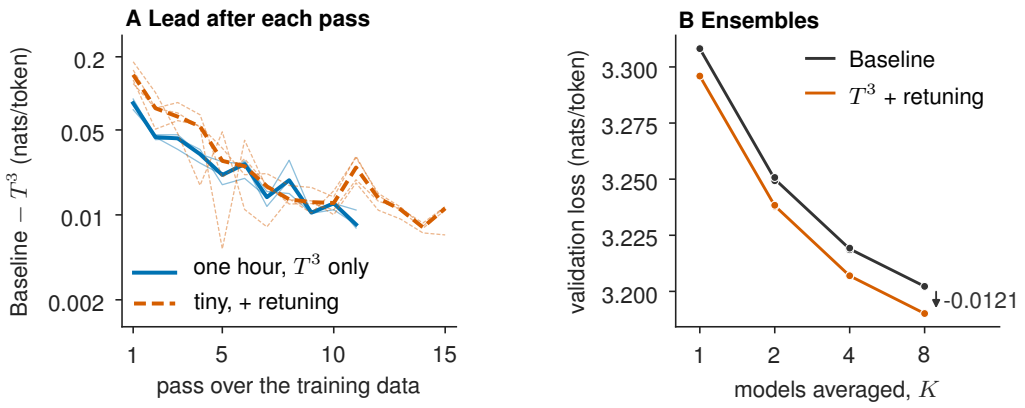


Figure 13: **The SlowRun gain persists across passes and after ensembling. (A)** Baseline minus T^3 validation loss at each pass, evaluated at the current weights. Thin lines show individual seeds and thick lines their median; solid lines are the one-hour episode-only runs and dashed lines are the tiny runs with T^3 and a retuned recipe. The loss-difference axis is logarithmic. **(B)** Tiny-track probability ensembles of models trained on seeds 42–49. At $K = 1$, the plotted loss is the mean member loss of the eight-member ensemble. Two disjoint member sets are shown at $K = 2$ and 4; lines connect their means. All losses are in nats per token.

M.4 The gain survives ensembling on the tiny track

Probability-averaged ensembles of independently trained models provide a strong reference when additional computation is available, so we test whether the single-model gain survives averaging. For members $p_1, \ldots, p_K$, the ensemble $\bar{p} = \frac{1}{K}\sum_j p_j$, and a target token y,

$$-\log \bar{p}(y) = \frac{1}{K}\sum_{j=1}^{K}\big[-\log p_j(y)\big] - \Big[\log \bar{p}(y) - \frac{1}{K}\sum_{j=1}^{K}\log p_j(y)\Big], \tag{38}$$

where the bracket is non-negative by concavity of the logarithm. Averaged over validation tokens, the bracket is the amount by which the ensemble beats its mean member, which we call the disagreement term D. Equation (38) is exact, so a training change helps an ensemble only through a lower mean member loss, a larger D, or both.

Every-step members lower the mean member loss and leave D approximately unchanged, so nearly all of the gain is retained. Eight tiny-track members trained with the submitted script on seeds 42 to 49 (member losses 3.2939 to 3.2984, mean 3.2959) are ensembled against eight record-script members on the same seeds (mean 3.3082) at matched K (Table 25). The disagreement term agrees to the third decimal between the two kinds of member at every K ($D = 0.058, 0.089, 0.106$ at $K = 2, 4, 8$ for both) and the pairwise KL between members (the KL divergence between the predictive distributions of two members, averaged over ordered pairs and over the tokens of eight probe batches) within 0.004, so the ensemble gain equals the member gain, 0.011 to 0.012 at every K, and the eight-member ensemble reaches 3.1901 against 3.2022. A mixed ensemble of two baseline and two T^3 members lands where its mean member puts it. The data-constrained counterpart of this experiment, with eight seeds per arm at 300M, is Section H.5.

M.5 A matched zero-adaptation control for SlowRun

To separate nonzero adaptation from execution of the episode, we ran three contemporaneous controls with the 1.44B SlowRun Muon recipe. Ordinary training uses the native update; the zero control executes the support/query episode with no temporary displacement; T^3 uses the same episode with a local normalized descent step of radius 0.3 on all Muon matrices. All three retain dropout 0.1 and

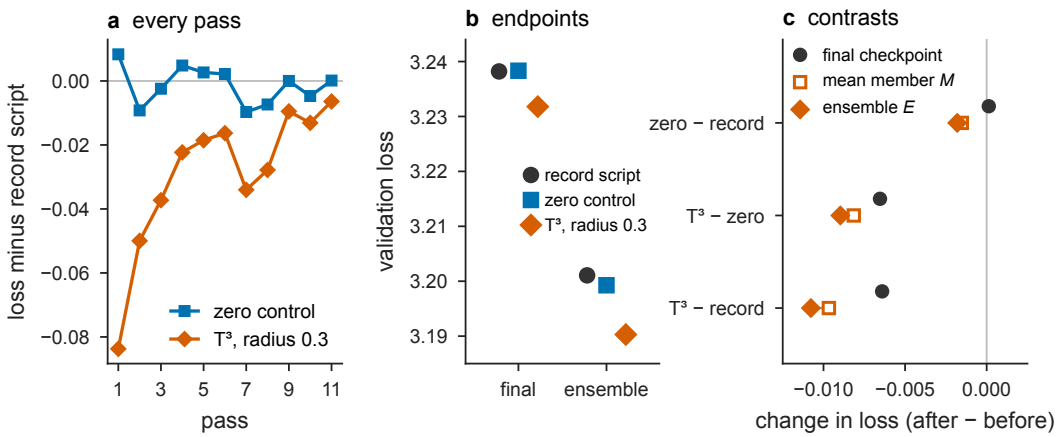


Figure 14: **The adaptation gain survives the zero control.** One-hour SlowRun recipe, seed 42, eleven passes and 2,097 updates: record script (black circles), zero control (blue squares), and T^3 with radius 0.3 (vermilion diamonds). (a) Pass-end validation loss minus the record script's, at every pass, without smoothing. (b) Final checkpoint (primary endpoint) and the fixed equal ensemble of passes 9–11 (secondary); Table 26 gives the values. (c) Each contrast is after minus before, for the final checkpoint, the mean member loss M, and the ensemble loss $E = M - J$; negative changes are favorable. The decomposition is arithmetic, without causal attribution or seed confidence intervals.

stochastic depth 0.05. This follow-up fixes 2,097 updates, eleven data passes, and seed 42. It is separate from the historical fixed-time records. Terminal negative log likelihood (NLL) is the primary endpoint. Equal probability averaging of checkpoints from passes 9–11 is secondary; the members are fixed in advance.

Against the zero control, T^3 lowers terminal NLL by 0.006537789 nats/token (Table 26). The zero control is 0.000130 worse than ordinary training at termination, but its equal probability ensemble is 0.001803 better. Thus, the observed ordinary-to-T^3 ensemble difference decomposes into a 0.001803 ordinary-to-zero difference and a 0.008969 zero-to-T^3 difference. This single execution per arm does not establish a systematic effect of the zero-adaptation path or identify the source of its difference from ordinary training. Let M be the mean member NLL, E the equal-ensemble NLL, and $J = M - E$ the ensemble gap. Relative to zero, the 0.008969 reduction in E comprises a 0.008146 reduction in M and a 0.000823 increase in J. Stronger late members account for most of this arithmetic decomposition.

Figure 14 retains every pass and all three contrasts. This single-seed result supports a nonzero-adaptation benefit in this recipe. It supplies no training-seed confidence interval, cross-setting explanation of Muon, causal diversity claim, or compute-speedup claim. The separately fixed recency-weighted ensemble remains an alternative endpoint and is not selected here.

Table 26: Matched SlowRun follow-up, seed 42 and 2,097 updates. Losses are nats/token; lower terminal, M, and E are better. M and E use passes 9–11; $J = M - E$ is an arithmetic gap. Values are rounded to six decimals, without seed confidence intervals.

configuration	terminal ↓	M ↓	E ↓	J
Ordinary	3.238200	3.254613	3.201066	0.053547
Zero adaptation	3.238330	3.253089	3.199263	0.053826
T^3, radius 0.3	3.231792	3.244942	3.190294	0.054648

N A ONE-PASS COMPUTE-EFFICIENCY PILOT

The fixed-data experiments measure generalization after repeated exposure. We separately test whether temporary adaptation improves quality at a fixed measured time when training positions are

not revisited. This single-seed pilot uses a different model, dataset and optimizer implementation, so it does not extend the primary setting's generalization claim.

Recipe and controls. We freeze the dense nanochat implementation at commit `b19b4f3e`, with 286,262,424 parameters, 12 layers, width 768, context 2,048, a 32,768-token vocabulary and the upstream SSSL attention pattern. Training uses a nonrepeating prefix of FineWeb-Edu shuffle shards 00000–00095, with shard 01822 held out, at dataset revision `4c8f30d6`. The tokenizer and token-byte table are hash pinned; their original training receipt is unavailable. Disjoint row groups retain the upstream beginning-of-sequence packing and cropping. Distinct corpus positions may contain identical text.

All four arms use seed 42, a global batch of 524,288 tokens and 2,378 updates, totaling 1,246,756,864 training token positions. The nominal recipe budget is 10^{18} FLOPs; conclusions use measured time. The upstream mixed Muon/AdamW optimizer has matrix learning rate .02, embedding/value-embedding learning rate .3, head learning rate .004, and residual/input-mixing scalar rates .005/.5. AdamW uses $\epsilon = 10^{-10}$ and zero weight decay, with betas $(.8, .95)$ except $(.96, .95)$ for input-mixing scalars. Matrix Muon retains five PolarExpress steps, factored second moment .95, momentum increasing from .85 to .95 over 300 updates, and cautious weight decay initially .2 with linear decay. There is no warmup; learning rates decay over the final half. Precision, PyTorch 2.9.1+cu128 and Flash Attention 3 are pinned across arms.

The native baseline uses the full-batch gradient. Split zero uses two equal halves of each microbatch, with the same consumed token stream and an ordinary gradient mean. The two adaptation arms use that split, total radii $\rho \in \{.03, .3\}$, and only the latter six layers' 12 MLP matrices (28,311,552 scalars). The full support gradient supplies the temporary direction; all model parameters receive the raw support/adapted-query gradient mean. Exact copies restore the weights before the persistent update. No gradient is differentiated through the inner step, no extra query backward is introduced, and no approximate data resume is accepted.

Evaluation and timing. Every 128 updates and at the terminal state, all arms evaluate the same 10,485,760 packed held-out positions. Bits per byte (BPB) divide total negative log likelihood by $\log(2)$ times the represented bytes, excluding special or ignored zero-byte targets: 10,477,045 scored tokens and 49,598,971 bytes. Consumed-stream digests, initial quality and scored counts agree across arms; checkpoints and all optimizer shards pass inspection.

The equal-time endpoint is the native baseline's final evaluation, 418.3 seconds after task startup. Each method contributes its latest evaluation completed within that time. Startup, data verification, compilation and preceding evaluations are charged; complete-attempt costs additionally include the final save and postprocessing. No interpolation is used to choose a quality point.

arm	terminal BPB	equal-time ΔBPB	complete attempt (s)
Native	0.900411	0.000000	425.1
Split zero	0.900392	+0.001469	434.4
Normalized .03	0.902633	+0.009243	439.7
Normalized .3	0.900162	+0.006530	440.5

Table 27: One-pass pilot, one seed; lower BPB is better. Differences are method minus native. Native completes update 2,378; the equal-time observations are update 2,304 for split zero and 2,176 for both adaptation arms. Times are elapsed seconds.

Outcome and stopping rule. Before execution, we require at least .005 better BPB at equal time and at least 5% conservative time-to-quality improvement at two of three targets: native terminal BPB plus .05, .02 and 0. We bracket each first recorded crossing by its previous and successful observations, without assuming monotonic learning. The ratio of native's previous-observation time to the selected method's successful-observation time supplies the conservative screen.

Radius .3 is selected by equal-time quality, but is .006530 BPB worse there and improves terminal equal-token BPB by only .000250. Its three conservative time ratios are .9068, .9153 and .9356; none passes 1.05. Split-zero terminal drift is only -0.000020, within the predeclared .005 execution-audit

margin. Early quality gains fade over the full horizon. Neither the acceleration criterion nor the .005 equal-token improvement criterion is met, so the predeclared null branch stops the larger-model extension.

These observations establish no compute acceleration for this pilot; they provide neither a scaling law nor a training-seed uncertainty estimate.

O ONE-PASS TRAINING AND TEST-TIME STEPS

O.1 ONE PASS OVER A LARGE CORPUS

We asked whether T^3 improves compute efficiency when no token is repeated. We used the optimization track of modded-nanogpt (Jordan et al., 2024b) on FineWeb10B (Penedo et al., 2024): a 12-layer, 768-wide transformer trained for 13,000 steps of 131,072 tokens or 3,250 steps of 524,288 tokens, about 1.7B tokens seen once. The recipes use Muon with decoupled weight decay (MuonW) or AdamW, with learning rates and weight decays tuned for each batch size before this study. Our 131K MuonW baseline reproduces its recorded loss to four decimals, and the 524K baselines are within 0.0012 of theirs. With the protocol's seed, the 131K AdamW baseline is 0.0125 above its record, as are two earlier reproductions with that seed; with two further seeds it agrees with the record (below). T^3 adapts the MLP matrices of the second half of the layers, 28.3M parameters. We chose radii 0.2 and 0.4 so that the displacement per coordinate matches the fixed-data and SlowRun settings, and we tested the support direction computed separately for each support partition or averaged over the batch. A baseline trained for 10% more steps, with its own full schedule, measures what a step-efficiency gain of that size would be worth. Table 28 lists every run; each is a single seed.

With MuonW, T^3 does not improve the one-pass loss at either batch size, and the larger radius is worse at 131K. Ten percent more baseline steps lower the loss by 0.012, so T^3, at 1.1 to 1.15 times the step time, is not a compute-efficient change for this recipe. With AdamW, single seeds suggested gains that further seeds did not confirm. At 524K, radius 0.4 lowered the loss by 0.006 with the first seed, about what 3.5% more steps buy (ten and twenty percent more steps lower it by 0.016 and 0.028), and by 0.0005 with the second. At 131K, every radius from 0.2 to 1.6 lowered the loss of the first seed by 0.008 to 0.016, and radius 0.4 by 0.0147, nearly as much as ten percent more steps (0.0162). The second and third seeds reversed it: radius 0.4 is 0.0037 and 0.0040 worse than their baselines. The first seed's baseline is the outlier among the three, 0.012 to 0.015 above the other two, which agree with the recorded values; from step 1,000 on it trails the second seed's by 0.012 to 0.016. Neither the direction nor the size of an early lead is consistent across seeds either. We therefore find no reliable one-pass gain for AdamW. T^3 processes the same tokens as the baseline, in two half-size passes; in our implementation a step takes 2 to 3% longer at 524K and about 12% longer at 131K. The tuned AdamW baseline remains 0.065 worse than the tuned MuonW baseline. An earlier one-pass pilot on a different architecture also found no gain at equal tokens (Appendix N).

Repeating the data in the same recipes. To separate data repetition from the recipe, we reran the 131K recipes for the same 13,000 steps on 400M unique tokens, four shards read 4.3 times in the same order, and on 100M tokens, one shard read 17 times; the validation set is unchanged (Table 29). Four passes raise the baseline loss by about 0.07. T^3 then changes the loss by -0.0012 and $+0.0031$ for MuonW at radii 0.2 and 0.4 and by -0.0008 for AdamW at radius 0.4, against $+0.0005$, $+0.0029$, and $+0.0037$ in one pass. With seventeen passes the recipes overfit: the validation loss reaches its minimum near step 3,000 and ends 0.8 higher, with differences between runs that change sign from one evaluation to the next. At the minimum, T^3 is within 0.0013 of the baseline for MuonW and 0.0049 lower for AdamW. Repetition in a recipe tuned for one pass therefore does not produce the gains of Sections 3.1 and 3.2, whose recipes were tuned for many passes over limited data.

O.2 TEST-TIME STEPS ON EACH DOCUMENT

T^3 trains with one temporary step and deploys the model without adaptation. We asked whether a model trained with T^3 benefits more from test-time training (Sun et al., 2020) than an ordinary model. On the tiny SlowRun track we took the final weights of the record script, of T^3, and of the within-document variant of Section 4.2, two seeds each. For each validation sequence of 2,048 tokens, we took one or two normalized descent steps on the MLP matrices using the next-token loss of the

Table 28: **One pass over FineWeb10B.** Final validation loss (nats per token) on 10.5M tokens. Δ is the run minus the same-batch baseline of the same seed at the same number of steps. Time is the training wall time, excluding evaluation; repeated runs of one configuration vary by a few percent. Seed 1 unless noted.

optimizer	batch	steps	run	loss	Δ	time (s)
MuonW	131K	13,000	baseline	3.2682		652
		14,300	baseline, 1.1× steps	3.2562	-0.0121	751
		13,000	T^3 $r = 0.2$, local	3.2687	$+0.0005$	749
		13,000	T^3 $r = 0.4$, local	3.2706	$+0.0024$	749
		13,000	T^3 $r = 0.2$, averaged	3.2687	$+0.0005$	726
		13,000	T^3 $r = 0.4$, averaged	3.2711	$+0.0029$	757
MuonW	524K	3,250	baseline	3.2769		518
		3,250	T^3 $r = 0.2$, averaged	3.2789	$+0.0019$	532
		3,250	T^3 $r = 0.4$, averaged	3.2764	-0.0006	531
AdamW	524K	3,250	baseline	3.3418		470
		3,575	baseline, 1.1× steps	3.3258	-0.0160	550
		3,900	baseline, 1.2× steps	3.3135	-0.0284	562
		3,250	T^3 $r = 0.2$, averaged	3.3400	-0.0019	482
		3,250	T^3 $r = 0.4$, averaged	3.3358	-0.0060	513
		3,250	T^3 $r = 0.8$, averaged	3.3386	-0.0032	482
		3,250	T^3 $r = 1.6$, averaged	3.3407	-0.0011	483
		3,250	baseline, seed 2	3.3403		471
		3,250	T^3 $r = 0.4$, averaged, seed 2	3.3398	-0.0005	480
AdamW	131K	13,000	baseline	3.3265		592
		14,300	baseline, 1.1× steps	3.3103	-0.0162	688
		13,000	T^3 $r = 0.2$, averaged	3.3183	-0.0082	663
		13,000	T^3 $r = 0.4$, averaged	3.3117	-0.0147	662
		13,000	T^3 $r = 0.8$, averaged	3.3138	-0.0127	666
		13,000	T^3 $r = 1.6$, averaged	3.3106	-0.0159	664
		13,000	baseline, seed 2	3.3144		625
		13,000	T^3 $r = 0.4$, averaged, seed 2	3.3181	$+0.0037$	666
		13,000	baseline, seed 3	3.3113		592
		13,000	T^3 $r = 0.4$, averaged, seed 3	3.3152	$+0.0040$	666

first 1,024 positions, and we measured the loss of the remaining positions at the adapted weights. Each sequence is adapted separately and the weights are restored afterwards; 4,864 sequences in all. Controls are a random direction, the ascent direction, and a step along the prefix gradient of a different sequence.

With the prefix in the attention context, one step of radius 0.25 lowers the suffix loss by 0.005 for all model types. Models trained with T^3 gain about 0.001 more: -0.0048 and -0.0056 for T^3 and -0.0056 and -0.0054 for the within-document variant, against -0.0047 and -0.0046 for the record script. At radius 0.5 the record script's models lose 0.0006 to 0.0008 while the T^3 models gain up to 0.0013.

Removing the prefix from the attention context raises the suffix loss by about 0.08. One step on the prefix at radius 0.5 then recovers 0.015 for every model type (-0.0148 for both seeds of the record script, -0.0153 and -0.0163 for T^3), and a step along another sequence's prefix gradient recovers at most 0.001. The recovered information is therefore specific to the document. The T^3 models recover 0.0005 to 0.0015 more. They are hurt less by large steps: two steps of radius 2 raise the suffix loss by 0.28 to 0.30 for T^3 and by 0.32 to 0.35 for the record script, consistent with the flatter end of training in Section 4.3. We then trained a model for exactly this use of the step, in the manner of end-to-end test-time training (Tandon et al., 2025): the support set is the first 1,024 positions of each sequence and the query set the remaining positions, forwarded without the prefix, so that the prefix can lower the query loss only through the temporary step. Episodes alternated with ordinary steps, so that the model still trained on the full context. Its endpoint is 0.0099 below the record script's, close to T^3's 0.0115, but the step it was trained for is no more useful at test time. With the suffix

Table 29: **The one-pass recipes on repeated data** (131K tokens per batch, 13,000 steps). Final and minimum validation loss. MuonW seed 1, AdamW seed 2.

unique tokens (passes)	run	final	Δ final	minimum	Δ minimum
1.7B (1)	MuonW baseline	3.2682		3.2682	
	MuonW T^3 $r = 0.2$	3.2687	+0.0005	3.2687	+0.0005
	AdamW baseline	3.3144		3.3144	
	AdamW T^3 $r = 0.4$	3.3181	+0.0037	3.3181	+0.0037
400M (4.3)	MuonW baseline	3.3369		3.3369	
	MuonW T^3 $r = 0.2$	3.3356	−0.0012	3.3356	−0.0012
	MuonW T^3 $r = 0.4$	3.3400	+0.0031	3.3400	+0.0031
	AdamW baseline	3.3861		3.3861	
	AdamW T^3 $r = 0.4$	3.3853	−0.0008	3.3853	−0.0008
100M (17)	MuonW baseline	4.6660		3.8484	
	MuonW T^3 $r = 0.2$	4.6521	−0.0139	3.8497	+0.0013
	MuonW T^3 $r = 0.4$	4.6461	−0.0199	3.8477	−0.0007
	AdamW baseline	4.6774		3.9316	
	AdamW T^3 $r = 0.4$	4.6391	−0.0384	3.9267	−0.0049

scored without the prefix, one step at the training radius recovers 0.0148, the same as for the record script, and the best of two steps 0.0159, against 0.0166. The coupling between T^3 and test-time training is consistent but small in this setting, even when training targets the test-time step directly. End-to-end test-time training (Tandon et al., 2025) takes many inner steps over a long context, with sliding-window attention; a single normalized step on one prefix is a much weaker inner loop.

Author TODOs — working draft

Status: 2026-09-16, 16:38 EDT. Original checklist: 4 complete, 3 open, 2 withdrawn. T1–T9 preserve all nine original items. **Primary setting:** 300M, fixed 196.6M-token DCLM pool, 12,000 updates, batch 64×4096, every-step with a 32/32 support/query split; 18 latter-half MLP matrices, normalized inner direction at total radius 0.3, and tuned outer Muon. Frozen before reading the Muon–T^3 endpoint; Adam is auxiliary. No new every-16. The bounded single-seed stages are accepted. Factual manuscript updates continue.

T1. **Ordinary-training optimizer comparisons: WITHDRAWN from the manuscript by the author.**

T2. **SAM baseline: COMPLETE (2026-09-23).** The radius sweep is in Appendix I, with its best row in Table 10: gain 0.007 at $2\times$ compute, 0.018 above the every-step design. Two radii below the original grid ($\rho = 0.003$ and 0.005) were added on 2026-09-23 and bracket the optimum at $\rho = 0.01$, so the reported best is no longer an edge of the grid. One weakness is left standing deliberately: the best gain is the size of the seed standard deviation and the sweep is single-seed, so the claim in the text bounds what SAM buys rather than establishing it. A second seed at the best radius would settle it and is the cheapest remaining experiment in this list.

T3. **Inner-step geometry and independent inner state: EXPERIMENTS COMPLETE.** Direction controls are in Table 10; separability is Table 23. The remaining independent-momentum experiment is a negative result: $\beta = .9/.99$ gives DCLM 3.544227/3.550689 versus memory-free T^3 at 3.522827 (seed 41). Both checkpoints verify separate inner state. Functional continuation passed; the original bitwise trajectory check failed and remains recorded. No wider beta sweep.

T4. **Every-16 follow-up: WITHDRAWN by the author (2026-09-15).** No new runs, replications, or equal-budget follow-up. Existing results retain their budget qualifications. Withdrawn does not mean experimentally complete.

T5. **Scale-ensemble compression: COMPLETE.** The every-16 paragraph is three sentences, and Table 15 is in Appendix J, alongside the member tables.

T6. **Unverified train–validation statistic: COMPLETE, corrected.** Section L.2 distinguishes raw, EMA, and checkpoint-averaged validation. Neither proposed numeric pair was reproducible; the mixed-window gap inference was removed.

T7. **Mechanism tests on a second seed: OPEN, deferred.** Half-batch collapse, pass-count tests, and the completed normalized period sweep remain single-seed. Historical every-k seed-0 checkpoints are partial. Selected replication must use the primary every-step, split-half setting; no every-16 replication.

T8. **Template and page limit: PARTIAL.** ICLR 2027 style is active, both tables moved, and the AI-use statement is drafted. Nine-page compression remains unfinished; factual updates continue. *2026-09-24:* the main text was restructured to the approved outline of 22 September and ends on page 8 before the references; the TODO pages and the experiment inventory now print after the appendices and are removed by `\workingdraftfalse` in `main.tex`.

T9. **Prose sweep: PARTIAL, deferred.** MLP and KL are expanded at first use. Remaining "not X but Y" constructions, sentence-internal colons, and announcing sentences still need the final prose pass.

Additional investigations (not substitutes for T1–T9)

- **Gradient-response diagnostic: OPEN, failed acceptance.** The 18-output grid and eight identical-command repeats failed the invariant check. Quarantine these outputs and resolve consistency before any response-curve claim.
- **Muon-shaped inner direction: COMPLETE, single seed.** At fixed 18 matrices/radius 0.3, loss is 3.504430 under outer Muon (+0.001202 versus normalized) and 3.542567 under AdamW (+0.019739). Branch B retains normalized inner; no new radius sweep. Both fulls and their source/config/norm/checkpoint audits passed.
- **Primary-setting plastic scope: COMPLETE, single seed.** Latter-half MLP / all MLP / transformer matrices give 3.502605 / 3.502888 / 3.505683 at radius 0.3. Both predeclared adjacent differences are below .003 (branch A); the widest-minus-primary difference is +.003078, so there is no equivalence claim. Retain the primary mask; the conditional earlier-half control is not released.
- **One-pass compute efficiency: COMPLETE PILOT, null branch.** All four full runs are accepted (Appendix N). Normalized .3 improves terminal BPB by only .000250 and is .006530 worse at equal time. The preregistered acceleration screen fails; no d16 or new radius is released. Original run records remain retained.