TERMINAL SHRINKAGE AVERAGING REVEALS A SCHEDULE-ESTIMATOR INTERACTION IN LLM PRE-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) pretraining conventionally returns the raw final iterate. This couples two design choices: the learning-rate schedule that generates the parameter trajectory and the estimator that constructs the deployed model (e.g. the raw final iterate or a checkpoint average). A schedule that promotes optimization progress may differ from one that minimizes variation in the raw final iterate. Separating these choices creates an opportunity to maintain progress late in training while reducing variation in the returned model. To this end, we propose *Terminal Shrinkage Averaging (TSA)*, which interpolates between the raw final iterate and the average of recent checkpoints to balance recent progress against terminal variation. We analyze how TSA changes the preferred terminal learning-rate schedule under a local quadratic approximation and test this interaction through a sequence of controlled NanoChat experiments. Finally, we demonstrate that the resulting gains transfer to depth-22 NanoChat, where the combined schedule and estimator improve validation quality. A qualifying time-to-GPT-2 run also finishes faster than the public baseline used in our experiments, providing preliminary evidence of benchmark acceleration.

1 INTRODUCTION

Large language model (LLM) pretraining typically follows a parameter trajectory under a decaying learning-rate schedule and ultimately deploys the raw final iterate (Brown et al., 2020; Hoffmann et al., 2022; Touvron et al., 2023). This convention answers two design questions with a single choice: *which trajectory should be followed near the end of training*, and *which function of that trajectory should be deployed?*

The end of the schedule is usually a *cooldown*, a terminal reduction of the learning rate. Its length and specific implementation materially change endpoint loss and can even reorder optimizer comparisons (Hoffmann et al., 2022; Bergsma et al., 2025; Wen et al., 2025). A small terminal learning rate reduces the variation of the raw final iterate, and it also limits how far the parameters can still move. A trajectory that stays more active late in training can therefore keep making progress while arriving at a worse raw final iterate.

Weight averaging separates the trajectory from the model returned from it (Polyak & Juditsky, 1992; Izmailov et al., 2018). Recent checkpoint-averaging methods such as Latest Weight Averaging (LAWA) have also been used to accelerate pretraining (Kaddour, 2022); in LLM pretraining, averaging can become more effective at higher learning rates (Sanyal et al., 2024). These findings motivate the question we study: if the deployed model averages checkpoints, does a weaker cooldown become preferable, and do the strength of the averaging and the strength of the cooldown need to be chosen together?

To study these questions, we propose terminal shrinkage averaging (TSA), an interpolation between the raw final iterate and the average of the last K checkpoints, with a single shrinkage coefficient setting how much of the checkpoint average enters the returned model.

This interpolation allows us to have control over the strength of averaging. The raw final iterate and uniform LAWA are its two endpoints. Under a local quadratic approximation of the loss, we

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.

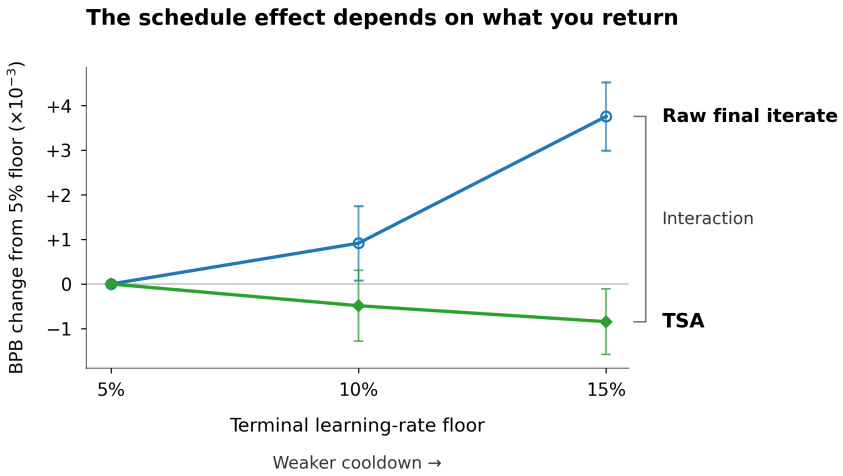

Figure 1: **Averaging weights enables more aggressive terminal optimization.** Raising the terminal learning-rate floor from 5% to 15% of peak worsens validation bits per byte (BPB) for the raw final iterate but improves it for TSA.

derive its exact surrogate risk, which is quadratic in the shrinkage coefficient. The risk separates the displacement of the mean output from the reduction in stochastic variation, demonstrating a bias-variance tradeoff, which gives conditions under which the minimizer is interior, so that neither the raw final iterate nor LAWA is optimal. We use the same approximation to show that an averaged output prefers a more active terminal trajectory than the raw final iterate does.

Our experiments are designed to isolate the effect of averaging, the effect of the terminal learning rate, and their interaction. Depth-12 NanoChat runs first show that the best shrinkage coefficient lies strictly between the two endpoints. A paired factorial design then crosses the terminal learning-rate floor (5%, 10%, and 15% of the peak learning rate) with the output estimator (the raw final iterate and TSA). Across five repetitions, raising the floor worsens validation BPB when the raw final iterate is deployed and improves it when TSA is deployed, and both effects grow with the size of the increase. A replication under pure AdamW reproduces the direction of this interaction, so this effect does not depend on NanoChat's Muon+AdamW optimizer. Finally, we apply the higher floor and TSA to the recipe behind the best depth-22 NanoChat time-to-GPT-2 run, which gives preliminary evidence that the interaction survives at scale, along with a qualifying record run.

Contributions. This work makes the following contributions.

- We separate two choices in pretraining: the learning-rate schedule, which determines the parameter trajectory, and the *output estimator*, which maps that trajectory to the deployed model. We introduce TSA, a shrinkage estimator between the raw final iterate and LAWA, and characterize its bias-variance tradeoff under a local quadratic approximation.
- Through controlled depth-12 NanoChat experiments, we identify that increasing terminal learning-rate activity is more favorable when TSA is returned, rather than the raw final iterate. Paired repetitions isolate this interaction from the effects of averaging or the terminal schedule alone.
- We demonstrate that the resulting gains transfer to the depth-22 NanoChat setting. At a matched training budget, the full schedule-estimator recipe improves validation quality over the underlying public recipe and produces a qualifying time-to-GPT-2 benchmark run.

2 Related Work

Iterate and checkpoint averaging. Averaging has classical asymptotic guarantees in convex problems (Polyak & Juditsky, 1992; Neu & Rosasco, 2018), while finite-sample analyses of tail averaging show how averaging can reduce the variance of the final output (Jain et al., 2018). In deep learning, stochastic weight averaging (SWA) returns an average over a late training trajectory and often finds solutions with improved generalization (Izmailov et al., 2018). Latest Weight Averaging

(LAWA) applies uniform averaging to a finite set of recent checkpoints and frames averaging as a way to accelerate training (Kaddour, 2022).

For language-model pretraining, Sanyal et al. (2024) show that checkpoint averaging becomes more effective at higher learning rates, while Ajroldi et al. (2025) characterize when averaging improves large-scale training. These results ask how the benefit of averaging changes under a given training trajectory. Our focus is complementary: we ask whether committing to an averaged output changes which terminal trajectory should be trained in the first place.

Lookahead also interpolates between fast and slow parameter states (Zhang et al., 2019). It performs this interpolation repeatedly during optimization and resets the fast weights after each inner loop. In contrast, TSA leaves the optimizer updates unchanged and performs a single terminal interpolation between the raw final iterate and a finite late-checkpoint average to construct the deployed model.

Learning-rate schedules designed with averaging. Several lines of work connecting sustained optimizer activity, cooldowns, and averaging. Warmup-stable-decay schedules retain a long constant-learning-rate phase before a terminal decay (Hu et al., 2024). This paper doesn't touch on averaging. Hägele et al. (2024) study constant-rate trajectories with cooldowns and show that SWA can be used in place of cooldowns. By allowing joint tuning of cooldowns and averaging, our method takes their observation that averaging plays a similar role to cooldowns and operationalizes it to improve optimization. Schedule-Free optimization asks the question of how to perform pretraining without knowing the length of the training a priori and strongly relies on learning-rate scheduling and iterate averaging (Defazio et al., 2024). Anytime pretraining falls in a similar regime (Meterez et al., 2026a). Our work is different from these two because it is in the setting where you know the training length in advance. Tian et al. (2026) more formally study the interchangeability of averaging and decay. Their Warmup-Stable and Merge framework uses weighted checkpoint merging to emulate decay from a stable-rate trajectory. However, they do not study the interpolation and interaction between these two techniques. Au & Block (2026) utilize averaging throughout training and develop PACE, which modifies the live AdamW trajectory by pulling parameters toward an exponential moving average. They do not study the interaction between the decay and the averaging, however. TSA leaves the live optimizer iterate unmodified and separately intervenes on terminal learning-rate activity. These are orthogonal directions.

Our setting is very controlled. We retain the underlying horizon-dependent pretraining recipe, alter only its terminal learning-rate activity, and separately vary the returned estimator. This design isolates whether the effect of the terminal schedule depends on which model is ultimately returned.

Pretraining optimizers. Adam and AdamW remain standard scalar-preconditioned optimizers (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). Matrix-preconditioned methods include Shampoo (Gupta et al., 2018), SOAP (Vyas et al., 2024), and Muon, which orthogonalizes momentum updates for matrix-valued parameters (Jordan et al., 2024; Liu et al., 2025). Optimizer rankings can depend on model scale, data ratio, tuning budget, and evaluation horizon (Wen et al., 2025; 2026). TSA does not alter the underlying optimizer update rule and is therefore complementary to optimizer design. Our primary experiments use NanoChat's native Muon+AdamW optimizer, and the appendix examines how the averaging gain is distributed across the corresponding parameter groups.

Compute-optimal pretraining and capability thresholds. Scaling-law studies relate model size, data, and compute (Kaplan et al., 2020; Hoffmann et al., 2022). NanoChat provides a compact full-stack training harness and a time-to-GPT-2 benchmark on one $8\times$H100 node (Karpathy & contributors, 2026). Qualification uses the 22-task DCLM CORE metric introduced with DataComp-LM (Li et al., 2024), with the original GPT-2 checkpoint as the reference capability (Radford et al., 2019). We begin from the exact PR #830 recipe available at the time of our experiments (Zinzi, 2026) and use it as the base for our matched-endpoint scale-transfer comparisons.

3 Method

The model returned from optimization depends on both the saved terminal checkpoints and the rule used to combine them. We first define that rule and analyze its risk for a fixed trajectory. We then ask how changing the rule can change the preferred terminal learning-rate activity.

3.1 TERMINAL SHRINKAGE AVERAGING

Consider optimization of a model with p parameters. Let $\theta_t \in \mathbb{R}^p$ denote the iterate at step t. At a terminal step T, we save K checkpoints separated by s optimizer steps and define their average as

$$\bar{\theta}_T := \frac{1}{K} \sum_{j=0}^{K-1} \theta_{T-js}. \tag{1}$$

Terminal Shrinkage Averaging returns

$$\widehat{\theta}_T(\alpha) := (1-\alpha)\theta_T + \alpha\bar{\theta}_T, \qquad \alpha \in [0,1]. \tag{2}$$

The endpoint $\alpha = 0$ returns the raw final iterate, while $\alpha = 1$ recovers uniform LAWA. We use LAWA as the averaging endpoint, although the interpolation can use any other averaging estimator in its place. Our goal is to characterize when the optimal model lies strictly between these two endpoints.

Because θ_T is itself included in $\bar{\theta}_T$, TSA assigns total weight

$$1 - \alpha + \frac{\alpha}{K}$$

to the final checkpoint and weight α/K to each of the other $K-1$ checkpoints.

Local quadratic model. To study how the interpolation affects performance, we approximate the loss over the short terminal region visited by the saved checkpoints using a positive-semidefinite quadratic model,

$$L_{\text{quad}}(\theta) := L(\theta^\star) + \frac{1}{2}(\theta - \theta^\star)^\top H(\theta - \theta^\star), \qquad H \succeq 0, \tag{3}$$

where $\theta^\star$ is a minimizer of the local model. Since H is positive semidefinite, this optimum may not be unique. This approximation is intended to describe the short late-training window used by TSA rather than the full optimization trajectory. Recent work finds that local linear and quadratic models become particularly accurate late in language-model pretraining (Meterez et al., 2026b).

For $x, y \in \mathbb{R}^p$, define the curvature-weighted inner product and associated seminorm

$$\langle x, y \rangle_H := x^\top H y, \qquad \|x\|_H^2 := x^\top H x. \tag{4}$$

Mean trajectory and stochastic residuals. To separate progress along the terminal trajectory from variation across training runs, we decompose each checkpoint into its mean and a stochastic residual. For notational convenience, write

$$\theta_j := \theta_{T-js}, \qquad j = 0, \dots, K-1,$$

so that $\theta_0 = \theta_T$. Define

$$\mu_j := \mathbb{E}[\theta_j], \qquad \epsilon_j := \theta_j - \mu_j, \qquad \mathbb{E}[\epsilon_j] = 0. \tag{5}$$

The expectation is over the training stochasticity, including the order of minibatches.

Define the average of the expected checkpoints and the average stochastic residual as

$$\bar{\mu} := \frac{1}{K} \sum_{j=0}^{K-1} \mu_j, \qquad \bar{\epsilon} := \frac{1}{K} \sum_{j=0}^{K-1} \epsilon_j. \tag{6}$$

We also define

$$m := \mu_0 - \theta^\star, \qquad A := \mu_0 - \bar{\mu}. \tag{7}$$

m is the displacement of the mean of the raw final iterate from the minimizer of the local quadratic model. A is the displacement between the mean of the raw final iterate and the expected checkpoint average.

Substituting these definitions into Eq. 2 gives the exact decomposition

$$\widehat{\theta}_T(\alpha) - \theta^\star = (1-\alpha)(\mu_0 + \epsilon_0) + \alpha(\bar{\mu} + \bar{\epsilon}) - \theta^\star \tag{8}$$

$$= m - \alpha A + \epsilon_0 + \alpha q, \tag{9}$$

where

$$q := \bar{\epsilon} - \epsilon_0. \tag{10}$$

Equation 9 separates the effect of TSA into two parts. The deterministic term $-\alpha A$ describes how averaging moves the mean output model relative to the local minimizer. The stochastic term αq describes how averaging changes the random deviation around that mean.

We measure performance under the local quadratic surrogate through

$$\mathcal{R}(\alpha) := \mathbb{E}\Big[L_{\text{quad}}\big(\widehat{\theta}_T(\alpha)\big) - L_{\text{quad}}(\theta^\star)\Big] = \frac{1}{2}\mathbb{E}\left\|\widehat{\theta}_T(\alpha) - \theta^\star\right\|_H^2. \tag{11}$$

Although $\theta^\star$ need not be unique when $H \succeq 0$, this quantity is invariant to the choice of local minimizer because distinct minimizers differ only along directions in the null space of H.

Theorem 1 (Risk of Terminal Shrinkage Averaging)**.** *Under the local quadratic model, let*

$$m := \mu_0 - \theta^\star, \qquad A := \mu_0 - \bar{\mu}, \qquad q := \bar{\epsilon} - \epsilon_0,$$

and define

$$V_q := \mathbb{E}\|q\|_H^2, \qquad C := \mathbb{E}\langle \epsilon_0, q\rangle_H.$$

Then the expected excess risk of TSA satisfies

$$\mathcal{R}(\alpha) = \mathcal{R}(0) + \alpha\big(-\langle m, A\rangle_H + C\big) + \frac{\alpha^2}{2}\left(\|A\|_H^2 + V_q\right). \tag{12}$$

Whenever $\|A\|_H^2 + V_q > 0$, this is a convex quadratic in α and

$$\alpha^\star = \Pi_{[0,1]}\left(\frac{\langle m, A\rangle_H - C}{\|A\|_H^2 + V_q}\right). \tag{13}$$

The linear term in Equation (12) exposes the central tradeoff. Since A points from the average of the saved checkpoints toward the raw final iterate, moving toward the checkpoint average can incur *lag* when the optimizer is still making useful progress. Define

$$G_{\text{lag}} := -\langle m, A\rangle_H.$$

At the same time, averaging can remove stochastic error from the raw final iterate. Define

$$G_{\text{noise}} := -C.$$

Then

$$\mathcal{R}(\alpha) = \mathcal{R}(0) + \alpha\left(G_{\text{lag}} - G_{\text{noise}}\right) + \frac{\alpha^2}{2}\left(\|A\|_H^2 + V_q\right). \tag{14}$$

A small amount of averaging therefore helps when $G_{\text{noise}} > G_{\text{lag}}$: the stochastic error removed by averaging outweighs the cost of moving toward older checkpoints. Full averaging need not be optimal because the quadratic term makes the marginal benefit of increasing α decrease. In particular, an interior optimum occurs when

$$0 < G_{\text{noise}} - G_{\text{lag}} < \|A\|_H^2 + V_q.$$

This gives the intuition behind TSA: retain enough weight on the raw final iterate to limit checkpoint lag while averaging strongly enough to reduce terminal variation.

The proof in Section C formalizes this fixed-trajectory result, and Section B.9 validates empirically that loss varies quadratically in α. The next question is whether the same tradeoff changes which trajectory should be trained.

3.2 A SUFFICIENT CONDITION FOR A MORE ACTIVE TERMINAL SCHEDULE

The previous analysis holds the training trajectory fixed and changes only the returned estimator. We next ask how changing the estimator can change which terminal trajectory is preferable.

Let ρ parameterize terminal optimizer activity; in our experiments, ρ corresponds to the terminal learning-rate floor. For fixed α, write the local risk as

$$\mathcal{R}_\alpha(\rho) = B_\alpha(\rho) + S_\alpha(\rho), \tag{15}$$

where

$$B_\alpha(\rho) := \frac{1}{2}\|m_\rho - \alpha A_\rho\|_H^2, \tag{16}$$

$$S_\alpha(\rho) := \frac{1}{2}\mathbb{E}\|\epsilon_{0,\rho} + \alpha q_\rho\|_H^2. \tag{17}$$

The first term captures mean displacement, while the second captures stochastic variation.

Let

$$\rho_{\mathrm{raw}} := \arg\min_\rho \mathcal{R}_0(\rho)$$

denote the terminal activity preferred when the raw final iterate is returned. $\mathcal{R}_0'(\rho_{\mathrm{raw}}) = 0$. Therefore,

$$\mathcal{R}_\alpha'(\rho_{\mathrm{raw}}) = \frac{d}{d\rho}\left[B_\alpha(\rho) - B_0(\rho)\right]_{\rho=\rho_{\mathrm{raw}}} - \frac{d}{d\rho}\left[S_0(\rho) - S_\alpha(\rho)\right]_{\rho=\rho_{\mathrm{raw}}}. \tag{18}$$

Consequently, if the stochastic benefit supplied by averaging grows with terminal activity faster than its additional deterministic cost, then $\mathcal{R}_\alpha'(\rho_{\mathrm{raw}}) < 0$. Under local convexity, the risk-minimizing averaged trajectory therefore lies at a larger value of ρ.

A one-dimensional corollary. We can demonstrate the schedule shift for a simple model using Equation (18). Consider one noisy terminal update on $L(x) = \frac{h}{2}x^2$ and construct TSA from the pre-update and post-update checkpoints. Whenever the risk-minimizing step sizes are interior, the preferred step size under shrinkage strength α is

$$\eta_\alpha^\star = \frac{\eta_{\mathrm{raw}}^\star}{1 - \alpha/2}. \tag{19}$$

Thus every $\alpha > 0$ strictly increases the preferred terminal step size, and full averaging of the two checkpoints doubles it. The exact risk and proof are provided in Section D.

This does not assert that increasing terminal activity is always beneficial under averaging, nor does it predict the magnitude of the optimal shift. Rather, it identifies the condition under which changing the returned estimator changes the preferred terminal schedule. The experiments in Section 4.3 test this interaction directly.

4 Experiments

We design the experiments as a sequence of controlled intervention tests. We first ask whether partial averaging improves the output model under a fixed training trajectory. We then ask whether increasing terminal learning-rate activity helps without averaging. Then, we show that our method is robust to choice of optimizer by replicating our results on AdamW. Finally, we combine the interventions to test the central prediction of our theory: an output estimator that reduces terminal variance can prefer a more active learning-rate schedule than the raw final iterate.

We use depth-12 NanoChat models for controlled ablations and a depth-22 model for a final time-to-GPT-2 experiment to demonstrate that our method scales. Unless otherwise stated, all depth-12 comparisons use the same model initialization, optimizer, training batches, and paired validation data so that differences can be attributed to the intervention being studied. We report validation bits per byte (BPB), where lower is better. For the depth-22 speedrun, we additionally report DCLM CORE, for which the qualification threshold is 0.256525. Full details are provided in Section A.

4.1 Partial shrinkage has a replicated interior optimum

We first hold each training trajectory fixed and vary only the returned estimator. To test whether the single-trajectory response from our preliminary sweep replicates, we evaluate a dense α grid on five independently ordered confirmation trajectories at the nanochat baseline 5% floor. Every value of α is evaluated on the same held-out batches within a trajectory. The grid is evaluated only after the confirmation estimator has been frozen and is not used to select that rule.

The response is broad and clearly interior. The best tested coefficient, $\alpha = 0.40$, improves BPB by 0.000794 ± 0.000087; the predeclared confirmation setting $\alpha = 0.55$ improves BPB by $0.000711 \pm$

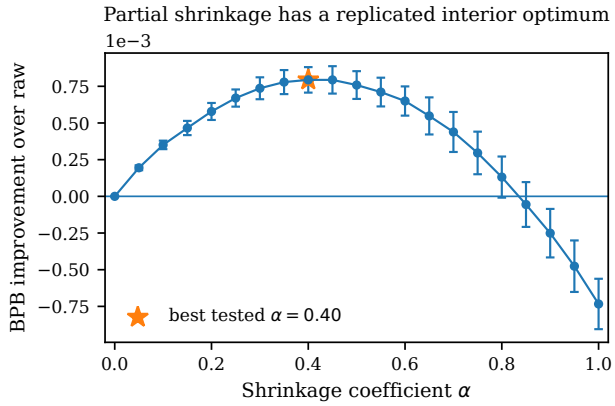

Figure 2: **Partial shrinkage has a replicated interior optimum.** Points show mean paired BPB improvement over the raw final iterate across five independently ordered depth-12 trajectories at the 5% terminal floor; bars are 95% t-intervals across trajectories. The best tested point is $\alpha = 0.40$, whereas full LAWA ($\alpha = 1$) is worse than the raw final iterate.

0.000098. In contrast, uniform LAWA worsens BPB by 0.000734 ± 0.000172. Thus, the benefit of partial rather than full averaging persists across training streams and does not require precise tuning of α. The full response at all three terminal floors is reported in Section B.1 and we compare our method to other common forms of averaging in Section B.

4.2 More terminal activity worsens the raw final iterate

We next vary only the terminal learning-rate floor and always return the raw final iterate. Using the same five paired stream seeds, increasing the floor from 5% to 10% worsens BPB for the raw final iterate by 0.000915 ± 0.000836, while increasing it to 15% worsens BPB for the raw final iterate by 0.003757 ± 0.000769. All five paired differences are positive for both interventions.

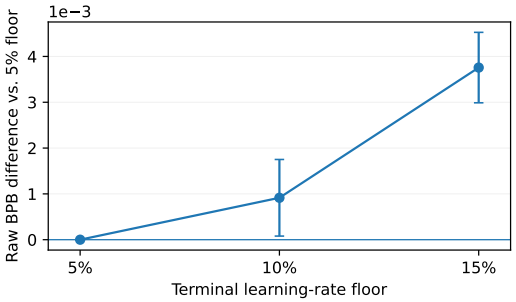

Figure 3: **More terminal activity worsens the raw final iterate.** Points show the mean change in BPB for the raw final iterate relative to the 5% floor across the five streams; bars are 95% t-intervals across streams. The 5% point is zero by construction. Increasing terminal activity degrades the raw final iterate at both tested intervention strengths, with a substantially larger effect at the 15% floor.

A denser preliminary single-trajectory sweep through 17.5% shows the same monotone, increasingly steep degradation of the raw final iterate. We report that descriptive sweep separately in Section B.8.

These comparisons establish the cost of a more active terminal schedule when the returned model is the raw final iterate. We next test whether TSA changes that cost.

4.3 THE OUTPUT ESTIMATOR CHANGES THE EFFECT OF TERMINAL ACTIVITY

We now vary the schedule and estimator together. We hold TSA fixed at $\alpha = 0.55$, $K = 8$, and $s = 32$, and compare the 5% baseline floor with both 10% and 15% floors over five paired data-order repetitions. For an active floor ρ, define the paired interaction

$$\mathcal{I}(\rho) := \big[L_{\mathrm{raw}}(\rho) - L_{\mathrm{raw}}(5\%)\big] - \big[L_{\mathrm{TSA}}(\rho) - L_{\mathrm{TSA}}(5\%)\big]. \tag{20}$$

Positive values mean the active schedule is more favorable under TSA than under the raw final iterate.

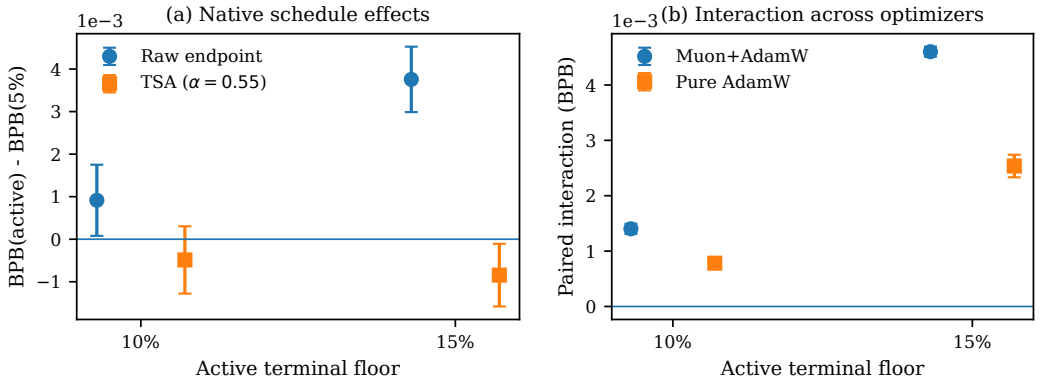

Figure 4: **The schedule–estimator interaction replicates across intervention strengths and optimizers. Left:** under NanoChat's native Muon+AdamW optimizer, points show the BPB effect of an active floor relative to the 5% floor; positive values mean worse BPB. The 15% floor worsens the raw final iterate but improves TSA. **Right:** paired interactions from Equation (20) under the native optimizer and pure AdamW, using the same fixed $\alpha = 0.55$. Positive values mean the active floor is more favorable under TSA. Bars are 95% t-intervals over five paired streams.

Table 1: **Two replicated schedule–estimator interventions.** Effects are active-floor BPB minus 5%-floor BPB, so positive simple effects indicate worse BPB. The interaction is the effect for the raw final iterate minus that for TSA. Intervals are 95% t-intervals over five paired repetitions.

Active floor	Raw final iterate	TSA	Interaction $\mathcal{I}$
10%	$+0.000915 \pm 0.000836$	-0.000487 ± 0.000791	$\mathbf{+0.001402 \pm 0.000092}$
15%	$+0.003757 \pm 0.000769$	-0.000844 ± 0.000737	$\mathbf{+0.004601 \pm 0.000091}$

The smaller 10% intervention already yields a clear interaction, even though the TSA simple effect is not individually resolved. At 15%, the simple effects themselves point in opposite and statistically resolved directions: the raw final iterate worsens, whereas TSA improves. The paired interaction is positive for every stream at both floors, showing that TSA makes a weaker cooldown more beneficial.

4.4 THE INTERACTION PERSISTS UNDER PURE ADAMW

The native optimizer assigns Muon to the main matrix parameter block. To test whether that assignment is necessary for the interaction, we repeat the same $5\%/10\%/15\%$ paired design under pure AdamW. We retain the fixed TSA coefficient $\alpha = 0.55$ and measure the within-stream interaction. It is 0.000782 ± 0.000089 BPB for the 10% floor and 0.002536 ± 0.000204 BPB for the 15% floor, and is positive in all five repetitions at both floors (Figure 4, right). The interaction is smaller than under the native optimizer, but persists without Muon updates, demonstrating that the phenomenon is not specific to Muon+AdamW.

4.5 SCALE TRANSFER TO THE NANOCHAT TIME-TO-GPT-2 BENCHMARK

The depth-12 results establish the interaction under controlled training conditions. We next test whether the combined schedule and estimator intervention of TSA transfers to the depth-22 NanoChat

speedrun setting. We start from the PR #830 recipe available at the time of our experiments (Zinzi, 2026), whose learning-rate schedule is calibrated for a data-to-parameter ratio of 9.4. We compare three configurations at the same physical ratio-9.0 endpoint (step 10,172): an early stop of the original ratio-9.4 schedule, the public recipe recalibrated directly to ratio 9.0, and the ratio-9.0 schedule with a 15% terminal learning-rate floor and TSA. For the treatment, we chose α based on the best-performing α in the depth-12 experiment at the 15% floor, which is 0.7 as seen in Section B.

Across three training repetitions, TSA improves validation BPB from 0.720235 ± 0.000735 to 0.719580 ± 0.000562 relative to the calibrated baseline (Table 2). The intervention improves BPB by 0.000654 ± 0.000184 in the paired comparison, with a favorable difference in all three seeds. [1]

Table 2: **Matched-endpoint depth-22 comparison over three seeds.** All configurations stop at step 10172. Time is accumulated training time. Values are means ± 95% Student-t intervals.

Configuration	Calibration	Floor	Time (min)	Val. BPB ↓	CORE ↑
PR #830 early stop	9.4	none	78.028 ± 2.023	0.721617 ± 0.000624	0.2479 ± 0.0200
PR #830 direct 9.0	9.0	none	77.999 ± 1.938	0.720235 ± 0.000735	0.2491 ± 0.0106
15% **floor + TSA**	**9.0**	**15%**	$\mathbf{77.896 \pm 0.029}$	$\mathbf{0.719580 \pm 0.000562}$	$\mathbf{0.2579 \pm 0.0159}$

Relative to the calibrated baseline, the seedwise CORE changes are $+0.0033$, $+0.0124$, and $+0.0108$. The NanoChat qualification threshold is 0.256525. While the average CORE value exceeds this, only one of the three seeds actually surpassed this threshold. For context, PR #830 reports 81.835 ± 0.138 minutes over six runs. Because CORE is discrete and noisy, and only one of our three repetitions individually qualifies, we treat this as a preliminary result with respect to an official record.

5 Conclusion

Pretraining need not return the raw final iterate. Once the deployed model is allowed to be a function of the terminal trajectory, the trajectory itself can be optimized differently. We study one simple realization of this idea: keeping optimization more active near the end of training while using Terminal Shrinkage Averaging (TSA) to construct the returned model. A local quadratic analysis characterizes the resulting tradeoff between optimization progress and stochastic variation and explains why partial, rather than uniform, averaging can be optimal.

Our controlled depth-12 experiments isolate the components of the method and establish the central schedule-estimator interaction: increasing terminal learning-rate activity can worsen the raw final iterate while improving the averaged output. At depth 22, the combined intervention improves validation BPB over a matched calibrated baseline in all three paired repetitions and yields one qualifying NanoChat time-to-GPT-2 run. More broadly, these results suggest that the optimization schedule and the rule used to construct the deployed model should be treated as a joint design problem rather than optimized independently.

Limitations. TSA explores only a small part of the space of possible output estimators and training trajectories. We use a single global shrinkage coefficient, although different layers or parameter tensors may benefit from different amounts of averaging; we do preliminary explorations of this in Section B.4 and Section B.5. Likewise, we modify terminal optimization through a simple learning-rate floor, but other changes to cooldown shape, duration, optimizer activity, or checkpoint placement could interact differently with the returned estimator. An issue with averaging is the memory required, which the observations in Section B.6 help mitigate by showing that averaging's benefit is heterogeneous between tensors. Our theoretical analysis uses a local quadratic approximation and is intended to explain the observed late-training tradeoff rather than provide a complete model of neural-network optimization. Finally, our experiments focus on NanoChat language-model pretraining, and establishing how these interactions transfer across architectures, optimizers, datasets, and substantially larger training regimes remains an important direction for future work.

[1] Training-time differences mostly reflect throughput variation across allocations; we do not interpret them as an algorithmic speed difference.

REFERENCES

Niccolò Ajroldi, Antonio Orvieto, and Jonas Geiping. When, where and why to average weights? In *International Conference on Machine Learning*, 2025.

Kwok Chun Au and Adam Block. Training for the model you return: Improving optimization for iterate-averaged language models, 2026. URL https://arxiv.org/abs/2606.25086.

Shane Bergsma, Nolan Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Straight to zero: Why linearly decaying the learning rate to zero works best for LLMs. In *International Conference on Learning Representations*, 2025.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Aaron Defazio, Xingyu Alice Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. *arXiv preprint arXiv:2405.15682*, 2024.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, 2018.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.18392.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, Xinrong Zhang, Zheng Leng Thai, Kaihuo Zhang, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies, 2024. URL https://arxiv.org/abs/2404.06395.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, 2018.

Prateek Jain, Sham M. Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Parallelizing stochastic gradient descent for least squares regression: Mini-batching, averaging, and model misspecification. *Journal of Machine Learning Research*, 18(223):1–42, 2018.

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

Jean Kaddour. Stop wasting my time! saving days of ImageNet and BERT training with latest weight averaging. *arXiv preprint arXiv:2209.14981*, 2022.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Andrej Karpathy and contributors. nanochat: A minimal experimental harness for training language models. https://github.com/karpathy/nanochat, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Jeffrey Li et al. DataComp-LM: In search of the next generation of language model pretraining datasets. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. arXiv:2406.11794.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

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

Alexandru Meterez, Pranav Ajit Nair, Depen Morwani, Cengiz Pehlevan, and Sham Kakade. Anytime pretraining: Horizon-free learning-rate schedules with weight averaging. *arXiv preprint arXiv:2602.03702*, 2026a.

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

Gergely Neu and Lorenzo Rosasco. Iterate averaging as regularization for stochastic gradient descent. *arXiv preprint arXiv:1802.08009*, 2018.

Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992. doi: 10.1137/0330046.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Sunny Sanyal, Atula Tejaswi Neerkaje, Jean Kaddour, Abhishek Kumar, and Sujay Sanghavi. Early weight averaging meets high learning rates for LLM pre-training. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=IA8CWtNkUr.

Changxin Tian, Jiapeng Wang, Qian Zhao, Kunlong Chen, Jia Liu, Ziqi Liu, Jiaxin Mao, Wayne Xin Zhao, Zhiqiang Zhang, and Jun Zhou. WSM: Decay-free learning rate schedule via checkpoint merging for LLM pre-training. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=HhThhjKyfw.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025.

Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them II: Hyperball optimization. *arXiv preprint arXiv:2606.16899*, 2026.

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

Giovanni Zinzi. Speed up the 8xH100 GPT-2 run to 81.8 minutes. nanochat pull request #830, https://github.com/karpathy/nanochat/pull/830, 2026.

A EXPERIMENTAL DETAILS

This section specifies the training trajectories, estimator selection, evaluation protocol, and scale-transfer comparison underlying the main experiments.

A.1 DEPTH-12 MODEL AND COMMON TRAINING SETUP

The depth-12 experiments use the NanoChat architecture with 12 transformer layers and 286,261,730 trainable parameters. Runs use sequence length 512, device batch size 4, gradient accumulation 16, and therefore 32,768 tokens per optimizer step. Every trajectory is trained for $T = 3000$ optimizer steps, with the first 40 steps used for learning-rate warmup. The model initialization seed is 1337 in every run; training stochasticity is introduced by an explicit permutation of the 3000 stored optimizer-step batch blocks.

The native optimizer follows NanoChat's heterogeneous parameter assignment: Muon updates the main matrix-valued parameter block, while AdamW updates the remaining parameter groups. The corresponding learning-rate settings are matrix learning rate 0.02, embedding learning rate 0.2, unembedding learning rate 0.004, scalar learning rate 0.5, and weight decay 0.28. In the pure-AdamW control, the matrix block is instead updated by AdamW with learning rate 0.003, $\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-10}$, and matrix weight decay 0.1; the remaining group-specific settings are inherited from the same pack. We use the pure-AdamW arm only for within-optimizer estimator comparisons, not to rank the two optimizers by absolute BPB.

A terminal floor ρ means that we follow the original cooldown until its learning-rate multiplier reaches ρ and then hold the multiplier fixed for the remainder of training. Unless otherwise stated, TSA uses $K = 8$ checkpoints separated by $s = 32$ steps. The selected steps end at step 3000 and span 224 optimizer steps, approximately 7.5% of training. Snapshots are stored in bfloat16 and all returned candidates are materialized and evaluated in float32.

A.2 DEVELOPMENT AND CONFIRMATION SPLIT

The new replicated suite separates rule development from confirmation. The development stage contains five streams for each optimizer at the 10% floor, with stream seeds

$$\{6103, 7203, 8303, 9403, 10503\}.$$

On a calibration block, it evaluates a scalar grid $\alpha \in \{0, 0.05, \ldots, 1\}$ and a two-group grid $\alpha_A, \alpha_B \in \{0, 0.125, \ldots, 1\}$. Rules maximize mean paired BPB gain over the raw final iterate across the five development streams, with deterministic ties favoring less extreme coefficients.

The confirmation stage contains five new stream seeds,

$$\{11103, 12203, 13303, 14403, 15503\},$$

and crosses two optimizers with terminal floors 5%, 10%, and 15%, for 30 independently trained trajectories. Within each stream seed, all floor and optimizer arms begin from the same sampled initial parameter vector and use the same explicit data-order permutation. A smoke test also verifies matching sampled training and evaluation data fingerprints across the native and pure-AdamW packs. Confirmation trajectories are trained and saved before the rule-freeze job, but are not evaluated until the frozen-rule file exists.

The native development stage selects scalar $\alpha = 0.55$. This is also the pre-existing main-paper setting, and it is used unchanged in both optimizer regimes in Section 4.4. Pure-AdamW development separately selects $\alpha = 0.50$; optimizer-specific selected results are reported only in the structured-estimator appendix below. The distinction lets the main text test transfer of the same estimator while retaining the full development study for diagnostics.

A.3 EVALUATION PROTOCOL AND UNCERTAINTY

Each pack contains 256 stored evaluation batches. The replicated suite reserves batches 0–31 for the earlier curve block, batches 32–95 for coefficient calibration, batches 96–159 for floor-development

bookkeeping, and batches 160–255 as the untouched 96-batch confirmation holdout. All estimators constructed from a given trajectory are evaluated on exactly the same holdout batches.

For single-trajectory sweeps, intervals quantify paired variation across held-out evaluation batches and do not measure training-run uncertainty. For the replicated experiments, all intervals in the main text and this appendix are 95% Student-t intervals across five independently ordered training streams. Estimator gains and schedule–estimator interactions are computed within stream before averaging, preserving the paired design.

The schedule effects for the raw final iterate and TSA are strongly correlated across paired streams ($r = 0.995$ for the $5\% \to 10\%$ intervention and $r = 0.994$ for $5\% \to 15\%$ under the native optimizer). This common-mode variation is removed by the within-stream interaction, explaining why its uncertainty is substantially smaller than that of either simple schedule effect.

A.4 Depth-22 scale-transfer protocol

The depth-22 experiments start from NanoChat PR #830 at commit `e09bc164162f35da0b5b8315be791e9e974a4c3a`. The recipe uses a depth-22 model, a 49,152-token tokenizer, FP8 training, fused Liger cross-entropy, device batch size 32, and learnable RMSNorm scales. A training ratio denotes the target data-to-parameter ratio; ratio 9.0 corresponds to optimizer step 10,172, whereas the original PR #830 ratio-9.4 schedule has a horizon of 10,624 steps.

We compare three configurations at the same physical ratio-9.0 endpoint. First, the *early-stop* control follows the original ratio-9.4 PR #830 schedule but stops at step 10,172 and returns the raw final iterate. Second, the *direct*-9.0 control recalibrates the schedule itself to ratio 9.0, again with no added terminal floor, and returns the raw final iterate. Third, the treatment uses the same direct ratio-9.0 schedule, clamps the terminal learning-rate multiplier from below at 15% of peak, and returns TSA. Each configuration is evaluated for experiment seeds 42, 43, and 44.

For TSA, we use $K = 8$ checkpoints at

$$9381,\ 9494,\ 9607,\ 9720,\ 9833,\ 9946,\ 10059,\ 10172,$$

corresponding to a spacing of 113 optimizer steps and a total terminal window of 791 steps, approximately 7.8% of the ratio-9.0 endpoint horizon. The 113-step spacing was originally obtained by transferring the depth-12 spacing fraction,

$$\frac{32}{3000} \approx 1.07\%,$$

to the ratio-9.4 depth-22 schedule horizon. When recalibrating the treatment schedule to ratio 9.0, we intentionally retain these same physical checkpoint steps rather than rescaling the averaging window. Thus, the direct-9.0 control and treatment differ in terminal learning-rate activity and returned estimator, while the TSA checkpoint window remains fixed.

We select the depth-22 TSA coefficient by selecting the best-performing α in the depth-12 experiment for a 15% floor. We therefore used $\alpha = 0.70$ from Section B.

For each configuration, we report canonical validation BPB and DCLM CORE over the three experiment seeds. Training time is NanoChat's accumulated training-iteration time through step 10,172 rather than end-to-end wall-clock duration. TSA introduces no additional optimizer steps. Because the repetitions span separate allocations, small timing differences mostly include hardware-session variation; we therefore use training time primarily to establish the matched compute budget rather than to claim an algorithmic speed difference between configurations.

B Additional depth-12 results

The main text uses fixed scalar TSA to isolate the schedule–estimator interaction. Here we examine the coefficient response across conditions, sensitivity to checkpoint selection, and whether alternative estimators or parameter-specific rules change that picture.

B.1 Replicated shrinkage responses across floors and optimizers

After freezing the confirmation estimators, we evaluate the complete scalar α grid on the untouched holdouts. These curves are descriptive and do not alter the predeclared confirmation rule.

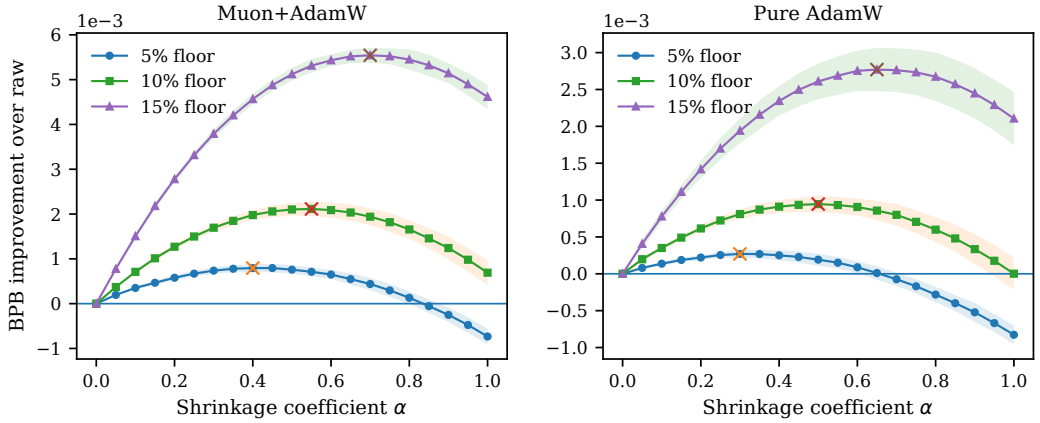

Figure 5: **The shrinkage response remains interior and shifts with terminal activity.** Curves show mean paired BPB improvement over the raw final iterate across five confirmation trajectories; shaded regions are 95% t-intervals across streams. Crosses mark the best tested coefficients. Under the native optimizer the best tested α moves from 0.40 to 0.55 to 0.70 as the floor rises from 5% to 10% to 15%; under pure AdamW it moves from 0.30 to 0.50 to 0.65.

Table 3: **Descriptive optima of the post-freeze scalar grids.** The grids were evaluated on confirmation holdouts after the estimator rules had been frozen and were not used for selection.

Optimizer	Floor	Best tested α	Gain over the raw final iterate
Muon+AdamW	5%	0.40	0.000794 ± 0.000087
	10%	0.55	0.002113 ± 0.000154
	15%	0.70	0.005543 ± 0.000166
Pure AdamW	5%	0.30	0.000269 ± 0.000053
	10%	0.50	0.000945 ± 0.000116
	15%	0.65	0.002770 ± 0.000295

The systematic rightward shift is consistent with the paper's comparative statics: when the terminal trajectory is more active, the returned model can benefit from stronger shrinkage. It also clarifies why a single moderate value such as $\alpha = 0.55$ is robust across the tested settings even though it is not the pointwise optimum at every floor. Although the pointwise optimum shifts with terminal activity, the fixed $\alpha = 0.55$ rule remains effective across the native-optimizer conditions: it retains approximately 90%, 100%, and 96% of the best observed scalar-TSA gain at the 5%, 10%, and 15% floors, respectively. Thus the interaction does not depend on precise coefficient tuning.

B.2 Sensitivity to checkpoint count and spacing

We train five additional depth-12 trajectories with the 10% floor and stream seeds $\{1103, 2203, 3303, 4403, 5503\}$. Every estimator is evaluated post hoc on the same five trajectories. We fix $\alpha = 0.55$ and vary $K \in \{4, 8, 16\}$ and $s \in \{16, 32, 64\}$.

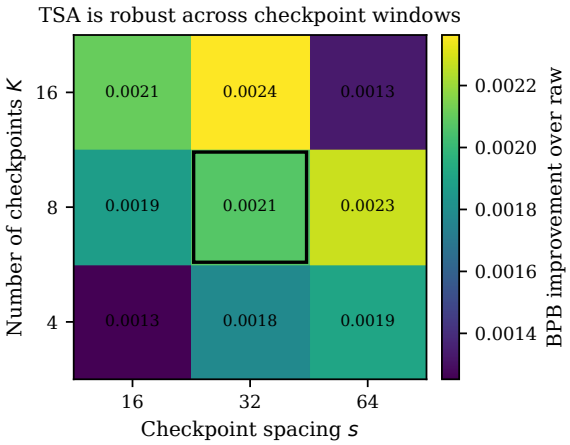

Figure 6: **TSA remains beneficial across checkpoint counts and spacings.** Cells show mean BPB improvement over the raw final iterate across five training repetitions. The outlined cell is the main-paper setting $K = 8, s = 32$.

TSA improves over the raw final iterate for all nine windows. The main $K = 8, s = 32$ setting gains 0.002070 ± 0.000205 BPB; the best tested window, $K = 16, s = 32$, gains 0.002363 ± 0.000207. Uniform LAWA is more sensitive: at $K = 8, s = 32$ it gains only 0.000684 ± 0.000261, and at $K = 16, s = 64$ it is worse than the raw final iterate by 0.005222 ± 0.000626. Partial shrinkage therefore broadens the useful range of checkpoint windows.

These trajectories and their evaluation block are separate from the main confirmation suite, so their nominally identical $K = 8, s = 32$ estimate need not numerically equal Table 3.

B.3 Alternative output estimators

The checkpoint-window results motivate a comparison with other ways to weight recent checkpoints. We use the following returned estimators.

Let $\theta^{(1)}, \ldots, \theta^{(K)}$ denote selected checkpoints ordered from oldest to newest, with $\theta^{(K)} = \theta_T$.

Raw final iterate. $\widehat{\theta}_{\text{raw}} = \theta_T$.

Latest Weight Averaging (LAWA).

$$\widehat{\theta}_{\text{LAWA}} = \frac{1}{K} \sum_{i=1}^{K} \theta^{(i)}.$$

Terminal Shrinkage Averaging.

$$\widehat{\theta}_{\text{TSA}}(\alpha) = (1 - \alpha)\theta_T + \alpha \widehat{\theta}_{\text{LAWA}}.$$

Thus the raw final iterate and LAWA are the endpoints $\alpha = 0$ and $\alpha = 1$.

Finite-window EWA.

$$\widehat{\theta}_{\text{EWA}}(\beta) = \frac{\sum_{i=1}^{K} \beta^{K-i} \theta^{(i)}}{\sum_{i=1}^{K} \beta^{K-i}}, \qquad 0 < \beta < 1.$$

Matched-window controls use the same $K = 8, s = 32$ checkpoints and $\beta \in \{0.50, 0.75, 0.90, 0.95\}$.

Checkpoint EMA. The checkpoint-EMA control uses the same normalized exponential rule over a denser, longer sequence with $\beta = 0.95, K = 16, s = 16$.

SWA-style late average. The SWA-style control is a broader uniform average with $K = 32, s = 16$. It is mathematically a late checkpoint average but spans a substantially longer terminal window than the main LAWA comparison.

More generally, shrinkage can be applied toward any fixed weighted checkpoint estimator. The same local quadratic decomposition follows after redefining the averaged endpoint. We focus on the uniform finite-window average to keep the method and controlled comparisons simple.

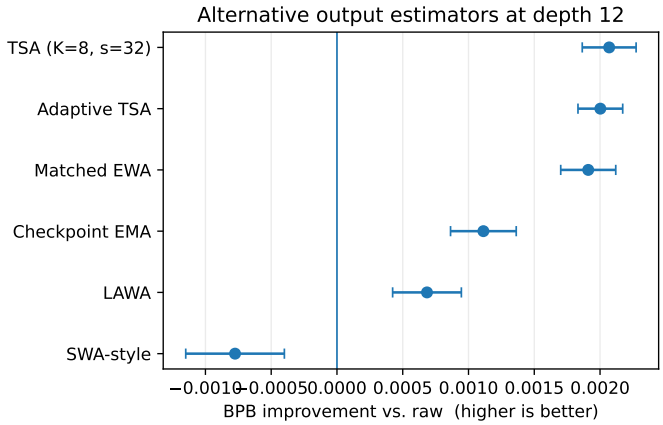

Figure 7: **Alternative output estimators at depth 12.** Points show mean paired BPB improvement over the raw final iterate, with 95% t-intervals across five training repetitions. Matched EWA denotes the best tested finite-window EWA ($\beta = 0.75$) on the same $K = 8, s = 32$ checkpoints as TSA.

The strongest matched EWA gains 0.001910 ± 0.000209 BPB. TSA gains 0.002070 ± 0.000205, with a paired TSA-over-EWA advantage of 0.000159 ± 0.000101. Checkpoint EMA gains 0.001113 ± 0.000250, while the long SWA-style average is worse than the raw final iterate by 0.000775 ± 0.000375. Recency weighting is useful, but the simple endpoint-shrinkage family remains competitive.

B.4 Exploratory tensorwise adaptive TSA

The preceding comparisons use one weighting rule across all parameters, implicitly treating their terminal lag–variation tradeoffs alike. To explore whether terminal updates can inform tensor-specific coefficients, let $\Delta_i^{(g)} = \theta_i^{(g)} - \theta_{i-1}^{(g)}$ be the update between adjacent saved checkpoints for tensor g, and let $\bar{\Delta}^{(g)}$ be the mean update. Define coherent drift energy

$$D_g = \|\bar{\Delta}^{(g)}\|_2^2$$

and residual update energy N_g after subtracting the mean drift. The empirical noise fraction is

$$\nu_g = \frac{N_g}{N_g + D_g + \epsilon}.$$

As a directional-persistence statistic, let c_g be mean cosine similarity between consecutive updates and define

$$o_g = \mathrm{clip}(1 - \max(c_g, 0), 0, 1)\,.$$

The heuristic adaptive rule is

$$\alpha_g = \mathrm{clip}\left(\frac{\nu_g + o_g}{2}, 0, 1\right), \qquad \widehat{\theta}_T^{(g)} = (1 - \alpha_g)\theta_T^{(g)} + \alpha_g \bar{\theta}_T^{(g)}.$$

It is not an estimator of the exact optimum in Theorem 1; it is a trajectory-statistic heuristic inspired by the same lag–variation intuition.

Across the five appendix trajectories, adaptive TSA over all tensors gains 0.002002 ± 0.000170 BPB. Applying the adaptive rule only to the Muon-managed tensors gains 0.001951 ± 0.000140, recovering approximately 97% of the all-tensor benefit. This observation motivated the initial optimizer-aligned two-group study below.

B.5 Initial optimizer-aligned structured TSA and parameter localization

The adaptive results suggest that shrinkage gains differ across tensors. We test a simpler, predeclared split based on NanoChat's optimizer parameter groups.

Let P_A project onto the flattened coordinates assigned to Muon by NanoChat's native optimizer, and let $P_B = I - P_A$. Group A contains 84,935,088 parameters, or 29.67% of the model. In pure-AdamW runs we retain the identical coordinate partition even though all parameters are trained with AdamW. Structured TSA is

$$\widehat{\theta}_T(\alpha_A, \alpha_B) = \theta_T + \alpha_A P_A(\bar{\theta}_T - \theta_T) + \alpha_B P_B(\bar{\theta}_T - \theta_T). \tag{21}$$

Scalar TSA is the diagonal restriction $\alpha_A = \alpha_B$.

The development grids select the same off-diagonal pair, $(\alpha_A, \alpha_B) = (0.625, 0.250)$, under both optimizers (Figure 8). The rule and group-schema hashes are written before any confirmation evaluation.

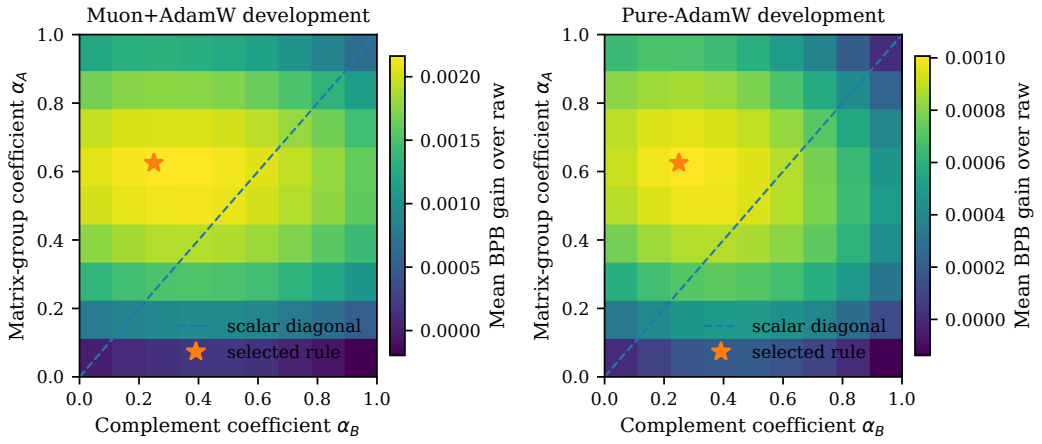

Figure 8: **Development surfaces for the initial optimizer-aligned two-group TSA.** Each panel reports mean paired BPB gain over the raw final iterate across five development streams at the 10% floor. The dashed line is scalar TSA. The same off-diagonal pair is selected under the native and pure-AdamW optimizers.

Table 4: **Frozen structured TSA versus frozen scalar TSA.** Development selected $(\alpha_A, \alpha_B) = (0.625, 0.250)$ for both optimizers, scalar $\alpha = 0.55$ for the native optimizer, and scalar $\alpha = 0.50$ for pure AdamW. Positive values in the final column favor structured TSA.

Optimizer	Floor	Scalar gain	Structured gain	Structured − scalar
Muon+AdamW	5%	0.000711 ± 0.000098	0.000730 ± 0.000108	0.000019 ± 0.000049
	10%	0.002113 ± 0.000154	0.002153 ± 0.000120	0.000040 ± 0.000050
	15%	0.005312 ± 0.000155	0.005369 ± 0.000117	0.000057 ± 0.000070
Pure AdamW	5%	0.000192 ± 0.000081	0.000243 ± 0.000071	0.000051 ± 0.000033
	10%	0.000945 ± 0.000116	0.000992 ± 0.000105	0.000046 ± 0.000017
	15%	0.002609 ± 0.000242	0.002597 ± 0.000243	-0.000012 ± 0.000037

On the native confirmation runs, structured TSA provides only 0.000040 ± 0.000050 BPB beyond the best frozen scalar rule at the predeclared 10% decision gate. The interval includes zero, and the additional gains at 5% and 15% are similarly small. We therefore retain scalar TSA as the main method. Under pure AdamW, the structured increment is positive and small at 5% and 10%, but disappears at 15%. These findings show measurable heterogeneity without establishing a practically important advantage for the extra parameter.

The group-only ablation is more revealing. Shrinking only the 29.67% matrix group recovers most of the full TSA gain, whereas shrinking only the much larger complement produces a substantially

smaller improvement. This asymmetry persists when all tensors are trained with AdamW (Figure 9 and Table 5).

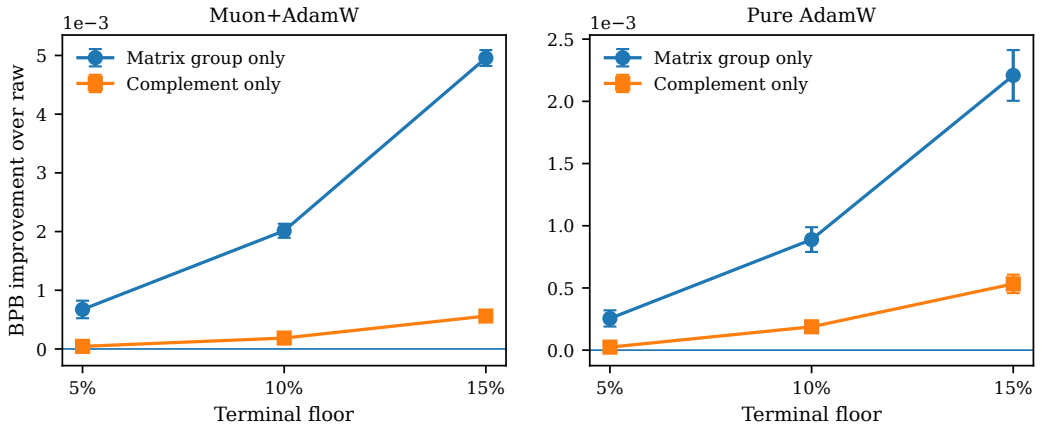

Figure 9: **The initial two-group split localizes most averaging gain to the matrix parameter group.** Points show paired BPB improvement over the raw final iterate when only one predeclared parameter group is shrunk. The same coordinate partition is used under both optimizers. Bars are 95% t-intervals across five confirmation streams.

Table 5: **Where the averaging gain is localized.** "Matrix group" is the 29.67% of parameters assigned to Muon in the native optimizer; the identical parameter partition is retained in pure-AdamW runs. All values are paired BPB improvements over the raw final iterate.

Optimizer	Floor	Matrix group only	Complement only	Matrix − complement
Muon+AdamW	5%	0.000673 ± 0.000149	0.000044 ± 0.000072	0.000629 ± 0.000192
	10%	0.002013 ± 0.000120	0.000186 ± 0.000049	0.001828 ± 0.000090
	15%	0.004957 ± 0.000136	0.000560 ± 0.000074	0.004396 ± 0.000150
Pure AdamW	5%	0.000255 ± 0.000065	0.000024 ± 0.000028	0.000231 ± 0.000056
	10%	0.000889 ± 0.000099	0.000188 ± 0.000032	0.000701 ± 0.000081
	15%	0.002208 ± 0.000204	0.000533 ± 0.000074	0.001676 ± 0.000170

Because the same pattern survives pure AdamW, it cannot be attributed solely to Muon updates. The optimizer-aligned matrix/complement partition is therefore best viewed as a useful first proxy for parameter-role or trajectory heterogeneity rather than as a causal Muon-versus-AdamW explanation. It establishes two facts that motivate the extension below: hidden matrices prefer substantially stronger shrinkage than the complement, while the corresponding two-parameter estimator adds only a small amount beyond scalar TSA.

B.6 Extending structured TSA to parameter-role groupings

The optimizer-aligned matrix/complement partition in Section B.5 provides a useful first localization: the hidden matrix block prefers substantially stronger shrinkage than the complement, and this asymmetry remains when all parameters are trained with AdamW. At the same time, the corresponding two-parameter estimator improves only modestly over scalar TSA. We therefore retain that split as an explicit baseline and ask whether a broader, but still low-dimensional family of parameter-role partitions explains additional out-of-sample gain.

Equation (21) is the $G = 2$ instance of a general groupwise estimator. Let $\{P_g\}_{g=1}^{G}$ be disjoint coordinate projections satisfying $\sum_{g=1}^{G} P_g = I$. We define

$$\widehat{\theta}_T(\boldsymbol{\alpha}) = \theta_T + \sum_{g=1}^{G} \alpha_g P_g(\bar{\theta}_T - \theta_T), \qquad \boldsymbol{\alpha} \in [0,1]^G. \tag{22}$$

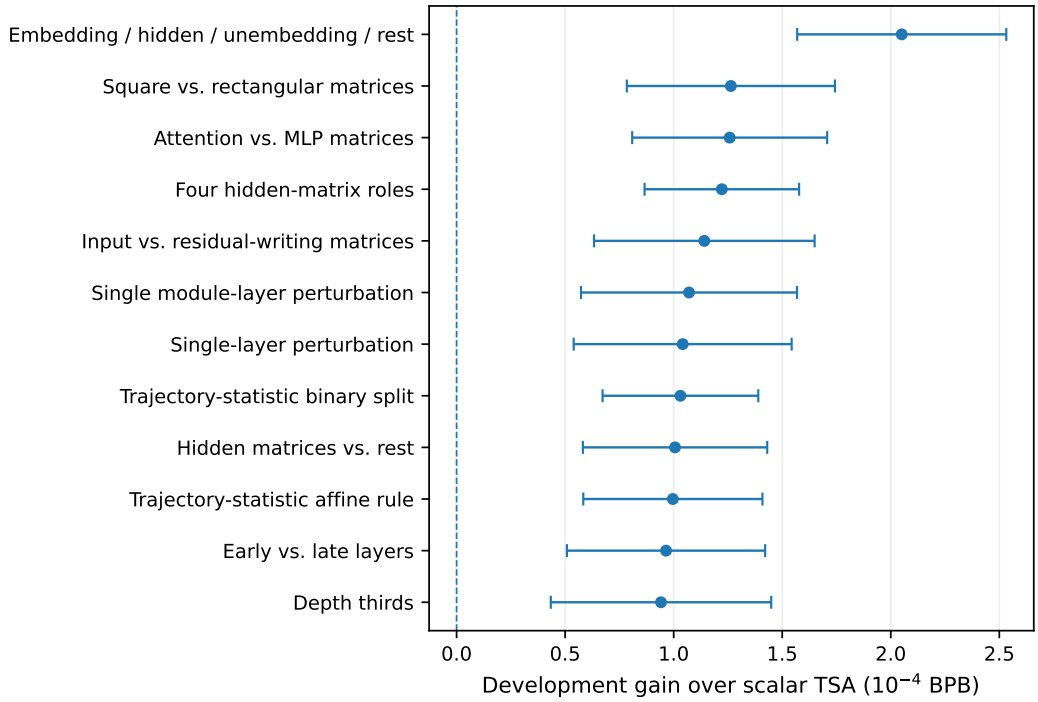

Figure 10: **Development search over parameter-group structures.** Each point is the best candidate within one predeclared structured family, measured relative to the development-selected scalar TSA rule; bars are 95% t-intervals across eight development trajectories. Because these data were used for selection, the estimates are descriptive rather than confirmatory. The "Hidden matrices vs. rest" row is the extension's direct version of the initial optimizer-aligned split in Section B.5; the embedding/hidden/unembedding/rest family produced the overall development winner.

Scalar TSA is the restricted family $\alpha_1 = \cdots = \alpha_G$; the matrix/complement rule above is one structured member of this larger family.

Development and confirmation protocol. All trajectories in this study use the 10% terminal floor and the main checkpoint window $K = 8, s = 32$. On eight native Muon+AdamW development trajectories, we evaluated 290 predeclared rules comprising a scalar baseline and 12 structured families. These included the earlier hidden-matrices-versus-rest partition, several functional and geometric subdivisions of the hidden block, trajectory-statistic rules, and partitions that separately expose embedding, unembedding, and remaining parameters. Rules were compared on a 32-batch calibration block. We then froze the scalar baseline, the winner from each structured family, and the overall structured winner before evaluating any new trajectory.

Confirmation used eight entirely fresh native trajectories and eight paired pure-AdamW trajectories, each evaluated on the untouched 96-batch holdout. The same parameter partitions and coefficients were transferred to pure AdamW without retuning. Unless noted otherwise, intervals in this subsection are 95% Student-t intervals across the eight trajectories.

The selected rule separates embedding parameters, hidden transformation matrices, the unembedding matrix, and the remaining scalar/other parameters:

$$(\alpha_{\text{emb}}, \alpha_{\text{hidden}}, \alpha_{\text{unemb}}, \alpha_{\text{rest}}) = (0, 0.65, 0.45, 0.25). \tag{23}$$

The initial localization is therefore preserved rather than overturned: hidden matrices still receive strong shrinkage. The additional gain comes primarily from refining the former complement, especially by separating the input embedding from the unembedding matrix and the remaining parameters.

Because the final checkpoint is included in the K-checkpoint average, its total weight in group g is $w_{T,g} = 1 - (K-1)\alpha_g/K$. For $K = 8$, the frozen rule therefore has the interpretation shown in Table 6.

Table 6: **Frozen parameter-role shrinkage rule.** The final column is the total weight placed on the final checkpoint after accounting for its inclusion in the checkpoint average.

Parameter group	α_g	Newest-checkpoint weight $w_{T,g}$
Embedding parameters	0.00	1.000
Hidden matrices	0.65	0.431
Unembedding matrix	0.45	0.606
Scalars and other parameters	0.25	0.781

Fresh confirmation. On the eight new native trajectories, the frozen role-based rule improved over the suite-selected scalar baseline $\alpha = 0.50$ by 0.000153 ± 0.000051 BPB, with 95% interval $[0.000103, 0.000204]$. It also improved directly over the paper's established scalar setting $\alpha = 0.55$ by 0.000138 ± 0.000045 BPB and over the earlier hidden-matrix/rest rule by 0.000108 ± 0.000013 BPB. Relative to the raw final iterate of the same 10%-floor trajectories, the selected groupwise estimator gained 0.002316 ± 0.000093 BPB. Thus scalar TSA supplies most of the total averaging benefit, while parameter-role conditioning provides a smaller but resolved additional gain.

The same frozen rule improved over scalar $\alpha = 0.50$ by 0.000178 ± 0.000023 BPB under pure AdamW and improved over the hidden/rest rule by 0.000130 ± 0.000013 BPB. This transfer reinforces the conclusion of the initial localization experiment: the signal cannot be explained solely by Muon updates. It is instead consistent with heterogeneity associated with parameter role, tensor geometry, group-specific learning-rate scale, or the resulting terminal dynamics. Pure AdamW does not distinguish among those mechanisms because the parameter groups retain their different base learning rates.

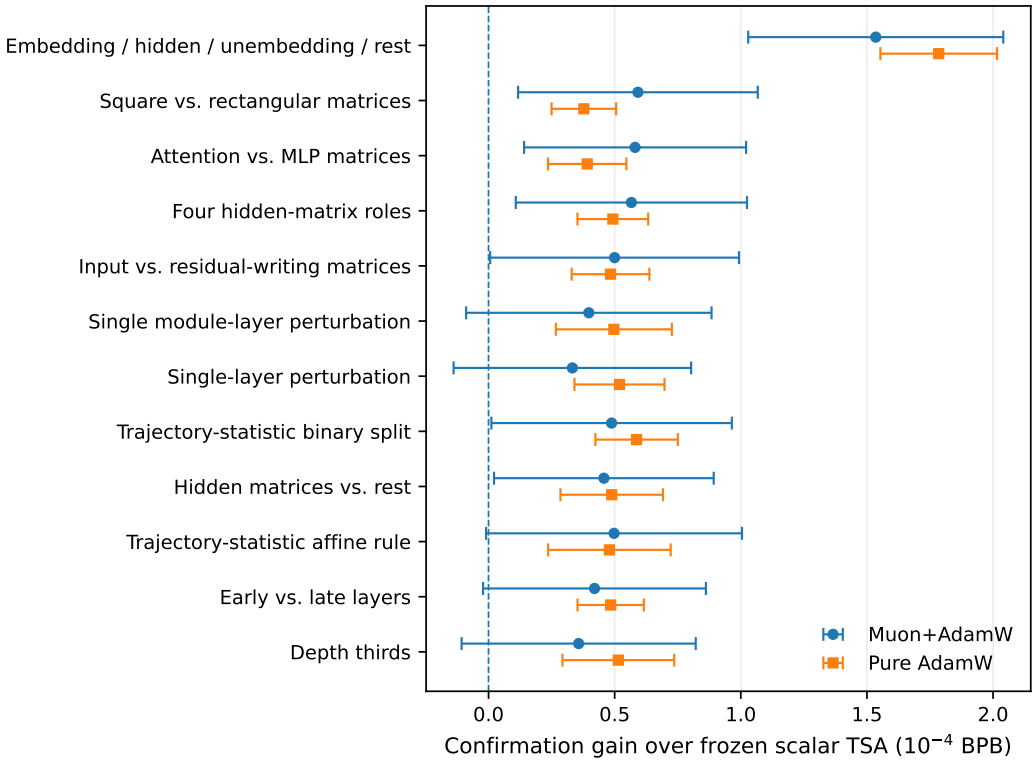

Figure 11: **Frozen groupwise rules on fresh confirmation trajectories.** Points show the gain of each development-selected family winner over the frozen scalar $\alpha = 0.50$ baseline. Circles use eight fresh native Muon+AdamW trajectories; squares transfer the identical rules to eight paired pure-AdamW trajectories without retuning. Bars are 95% t-intervals. The "Hidden matrices vs. rest" row carries forward the initial two-group experiment, while the role-based embedding/hidden/unembedding/rest rule provides a further confirmed gain in both optimizer regimes.

Several alternative partitions also carried smaller confirmation signals, including attention-versus-MLP, square-versus-rectangular, and four-way hidden-matrix role splits. Their gains were substantially below that of the selected role-based split. Consistently, the measured trajectory scores of QKV, attention-output, MLP-up, and MLP-down matrices were tightly clustered between 0.708 and 0.715, providing little support for a large functional separation within the hidden matrix block. The combined evidence therefore supports a progression rather than a replacement: the initial optimizer-aligned split correctly localized most averaging value to hidden matrices, and the broader search subsequently identified additional heterogeneity within the complement. These experiments do not establish that the selected grouping is universally optimal.

B.7 Groupwise terminal activity as a mechanism probe

The averaging experiments raise a separate question: if parameter groups prefer different returned estimators, should they also receive different amounts of terminal optimizer activity? We tested this as a fixed mechanism probe rather than a new hyperparameter search. The estimator in Equation (23) remained frozen, and its coefficients were mapped monotonically to terminal learning-rate floors via

$$\rho_g = 0.05 + 0.10\frac{\alpha_g}{0.65}. \tag{24}$$

This gives floors of 5.0% for embedding parameters, 15.0% for hidden matrices, 11.92% for the unembedding matrix, and 8.85% for the remaining parameters. The schedule implementation directly routed each optimizer parameter group to one of these four buckets.

We trained eight new paired native trajectories under four fixed schedules: uniform 5%, uniform 10%, the mapped groupwise schedule, and a uniform floor equal to the parameter-count-weighted

mean of the mapped floors (8.576%). Within each stream, all arms shared initialization, data order, checkpoint window, and the 96-batch holdout. For every trajectory, we evaluated the raw final iterate, scalar TSA at $\alpha = 0.55$, and the frozen role-based estimator. No schedule or estimator was selected using these trajectories.

For an estimator E, define the mapped-schedule effect relative to the parameter-count-matched uniform schedule as $\Delta_E = L_E(\text{mapped}) - L_E(\text{matched})$, so positive values mean that the mapped schedule is worse. The corresponding interaction is

$$\mathcal{I}_{\text{group}} = \Delta_{\text{raw}} - \Delta_{\text{groupwise}}. \tag{25}$$

Table 7: **Predeclared groupwise terminal-floor intervention.** Schedule effects are mapped-floor BPB minus the parameter-count-matched uniform-floor BPB; positive values indicate worse BPB. The interaction is the effect for the raw final iterate minus that for groupwise TSA. Intervals are 95% t-intervals over eight paired trajectories.

Returned estimator	Mapped-floor effect	95% interval
Raw final iterate	$+0.003404 \pm 0.000386$	$[+0.003019, +0.003790]$
Frozen groupwise TSA	-0.000112 ± 0.000294	$[-0.000407, +0.000182]$
Interaction $\mathcal{I}_{\text{group}}$	$\mathbf{+0.003516 \pm 0.000098}$	$[+0.003418, +0.003614]$

The tested heterogeneous schedule therefore did not establish a better final training recipe: under the frozen groupwise estimator, its difference from the matched uniform schedule was small and unresolved. Its effect nevertheless depended strongly on the returned estimator. The mapped allocation made the raw final iterate worse by 0.003404 BPB, while the groupwise estimator removed essentially all of that relative penalty, yielding the precisely estimated interaction in Table 7.

Parameter-specific averaging also became more valuable on the heterogeneous trajectory. Within the mapped arm, frozen groupwise TSA improved over the raw final iterate by 0.005114 ± 0.000174 BPB and over scalar TSA at $\alpha = 0.55$ by 0.000322 ± 0.000061 BPB. The complete mapped-schedule plus groupwise-TSA output improved over the paired uniform-5% raw final iterate by 0.000991 ± 0.000375 BPB. As an ancillary check, these eight new streams independently reproduced the ordinary uniform-5% to uniform-10% schedule–estimator interaction at 0.001474 ± 0.000086 BPB, close to the main confirmation estimate.

The parameter-count-matched control equalizes the mean floor across parameter coordinates, not optimizer-update energy, displacement, or noise injection; the corresponding optimizer groups have different base learning rates and update geometries. We therefore interpret this experiment as evidence that the value of parameter-specific output estimation depends on how terminal activity is distributed across parameter groups, not as evidence that the particular mapped cooldown should replace a uniform schedule.

B.8 Full preliminary single-trajectory sweeps

Before the replicated confirmation experiments, we performed two denser diagnostic sweeps on individual depth-12 trajectories. The first holds the training trajectory fixed and varies the scalar TSA coefficient α. The second returns the raw final iterate and varies only the terminal learning-rate floor. These experiments provide higher-resolution views of the response surfaces, but their uncertainty reflects evaluation batches within a single trained trajectory rather than variation across independent training runs. We therefore treat them as descriptive preliminary evidence; the corresponding replicated main-text experiments provide the stronger inferential results.

Scalar shrinkage sweep. Holding the baseline 5%-floor trajectory fixed, partial shrinkage produces a broad interior optimum, while full LAWA eventually becomes worse than the raw final iterate. This preliminary response motivated the replicated estimator sweep in Figure 2.

Terminal-floor sweep for the raw final iterate. We separately vary only the terminal learning-rate floor while returning the raw final iterate. The raw final iterate worsens monotonically over the tested range, and the degradation becomes steeper at the larger floors (Figure 12). This shape is consistent with the replicated 5%, 10%, and 15% comparison in Figure 3.

Table 8: **Preliminary fixed-trajectory scalar sweep.** Improvement is BPB for the raw final iterate minus estimator BPB. Intervals quantify paired evaluation-batch uncertainty for one trajectory.

α	Validation BPB ↓	Improvement vs. raw final iterate ↑
0.00	1.061224	—
0.10	1.060872	$+0.000352 \pm 0.000074$
0.20	1.060623	$+0.000602 \pm 0.000104$
0.30	1.060483	$+0.000741 \pm 0.000138$
0.40	1.060458	$+0.000766 \pm 0.000176$
0.50	1.060499	$+0.000725 \pm 0.000212$
0.55	1.060554	$+0.000670 \pm 0.000225$
0.60	1.060679	$+0.000546 \pm 0.000258$
0.65	1.060760	$+0.000465 \pm 0.000288$
0.70	1.060852	$+0.000372 \pm 0.000299$
0.75	1.060985	$+0.000239 \pm 0.000324$
0.80	1.061190	$+0.000035 \pm 0.000346$
0.90	1.061570	-0.000346 ± 0.000373
1.00	1.062068	-0.000844 ± 0.000424

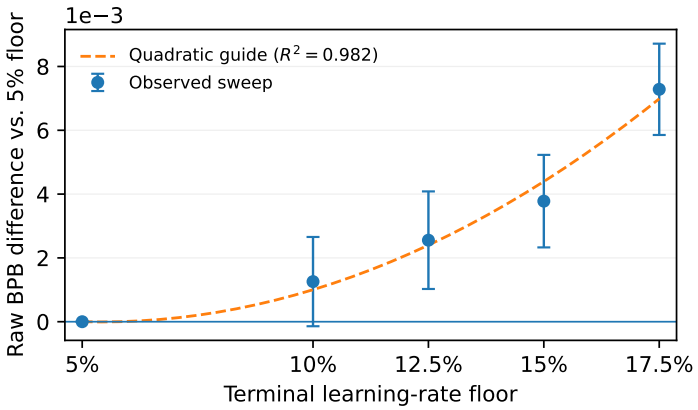

Figure 12: **Preliminary response of the raw final iterate across terminal floors.** Points show BPB differences for the raw final iterate relative to the 5% floor along one training trajectory; bars quantify 95% paired evaluation-batch uncertainty. The dashed curve is a least-squares quadratic guide, constrained to pass through the 5% baseline ($R^2 = 0.982$). The fit is descriptive only: unlike the exact quadratic dependence of the local TSA surrogate on shrinkage coefficient α, the theory does not imply that BPB for the raw final iterate must be quadratic in the terminal floor.

B.9 Replicated quadratic response

Under the local quadratic surrogate, the TSA risk is exactly quadratic in the shrinkage coefficient α for a fixed trajectory. We therefore ask whether the observed validation response has the corresponding shape. For each optimizer and terminal floor, we fit

$$g(\alpha) = b\alpha + c\alpha^2, \qquad g(0) = 0, \tag{26}$$

to the mean paired BPB improvement over the raw final iterate across the five confirmation trajectories. Because g denotes improvement rather than risk, a beneficial interior optimum corresponds to a concave response. The fitted vertex is $\widehat{\alpha}^\star = -b/(2c)$.

The quadratic description is accurate across all six optimizer-floor conditions, with $R^2 \geq 0.99950$. The fitted vertices closely track the independently observed grid optima. Under Muon+AdamW, the fitted optimum moves from 0.419 to 0.547 to 0.701 as the terminal floor rises from 5% to 10% to 15%; under pure AdamW it moves from 0.330 to 0.499 to 0.668. Thus the near-quadratic response is not specific to one trajectory, floor, or optimizer regime, and the preferred amount of shrinkage systematically increases with terminal activity. This provides an empirical consistency check for the

Table 9: **Single-trajectory floor sweep without averaging.** Differences are relative to the 5% floor. Intervals quantify paired evaluation-batch uncertainty for one trajectory.

Floor	Validation BPB ↓	Difference vs. 5% ↓
5%	1.061224	—
10%	1.062482	$+0.001257 \pm 0.001399$
12.5%	1.063779	$+0.002555 \pm 0.001529$
15%	1.065003	$+0.003779 \pm 0.001450$
17.5%	1.068508	$+0.007284 \pm 0.001431$

Table 10: **Quadratic fits to the replicated shrinkage responses.** For each optimizer and terminal floor, the best tested coefficient on the post-freeze scalar grid is compared with the vertex of the fitted quadratic. Fits use the mean paired BPB improvement over the raw final iterate across five confirmation trajectories.

Optimizer	Floor	Best tested α	$\widehat{\alpha}^\star$	R^2
Muon+AdamW	5%	0.40	0.419	0.99977
	10%	0.55	0.547	0.99976
	15%	0.70	0.701	0.99970
Pure AdamW	5%	0.30	0.330	0.99984
	10%	0.50	0.499	0.99950
	15%	0.65	0.668	0.99993

local quadratic description and its comparative-static interpretation; it does not imply that global neural-network training dynamics are quadratic.

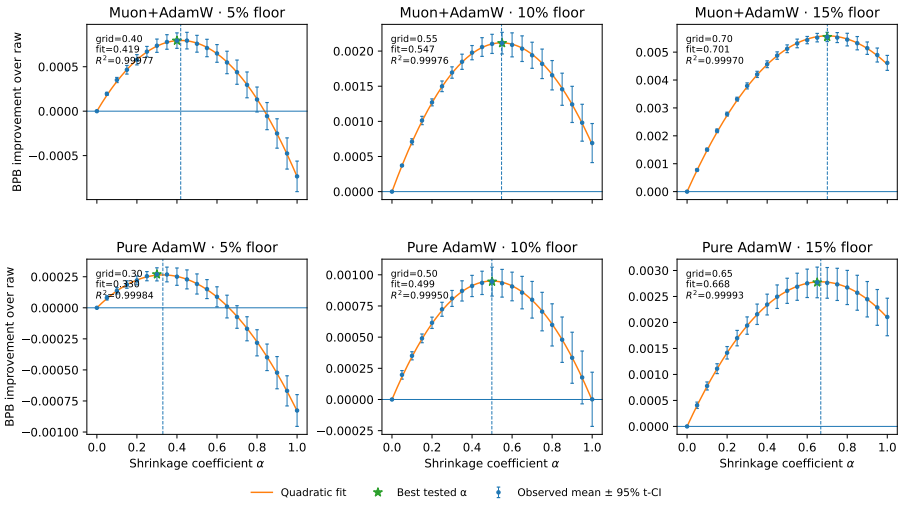

Figure 13: **The replicated shrinkage responses are nearly quadratic.** Each panel shows mean paired BPB improvement over the raw final iterate across five confirmation trajectories; bars are 95% t-intervals across trajectories. Solid curves are fits of $g(\alpha) = b\alpha + c\alpha^2$ constrained by $g(0) = 0$. Stars mark the best tested scalar-grid coefficients, and dashed vertical lines mark the fitted vertices.

C Proof of the TSA risk theorem

Proof of Theorem 1. By Equation (9),

$$\widehat{\theta}_T(\alpha) - \theta^\star = m - \alpha A + \epsilon_0 + \alpha q.$$

The random component $\epsilon_0 + \alpha q$ has mean zero. Therefore,

$$2\mathcal{R}(\alpha) = \mathbb{E}\,\|m - \alpha A + \epsilon_0 + \alpha q\|_H^2 \tag{27}$$

$$= \|m - \alpha A\|_H^2 + \mathbb{E}\|\epsilon_0 + \alpha q\|_H^2 \tag{28}$$

$$= \|m - \alpha A\|_H^2 + \mathbb{E}\|\epsilon_0\|_H^2 + 2\alpha C + \alpha^2 V_q. \tag{29}$$

Expanding the deterministic term,

$$\|m - \alpha A\|_H^2 = \|m\|_H^2 - 2\alpha\langle m, A\rangle_H + \alpha^2\|A\|_H^2.$$

Since

$$\mathcal{R}(0) = \frac{1}{2}\left(\|m\|_H^2 + \mathbb{E}\|\epsilon_0\|_H^2\right),$$

dividing by two yields

$$\mathcal{R}(\alpha) = \mathcal{R}(0) + \alpha\big(-\langle m, A\rangle_H + C\big) + \frac{\alpha^2}{2}\big(\|A\|_H^2 + V_q\big).$$

Differentiating gives

$$\mathcal{R}'(\alpha) = -\langle m, A\rangle_H + C + \alpha\big(\|A\|_H^2 + V_q\big).$$

When $\|A\|_H^2 + V_q > 0$, the unconstrained minimizer is

$$\alpha_{\text{unc}} = \frac{\langle m, A\rangle_H - C}{\|A\|_H^2 + V_q}.$$

Projection onto $[0, 1]$ gives Equation (13). The optimum is strictly interior exactly when

$$0 < \langle m, A\rangle_H - C < \|A\|_H^2 + V_q.$$

□

D ONE-STEP QUADRATIC ANALYSIS

Consider the one-dimensional quadratic loss

$$L(x) = L(0) + \frac{h}{2}x^2, \qquad h > 0. \tag{30}$$

Let $x^- \neq 0$ be a deterministic parameter value immediately before one terminal stochastic-gradient update, and let

$$x^+ = (1 - \eta h)x^- - \eta\xi, \qquad \mathbb{E}[\xi] = 0, \qquad \mathbb{E}[\xi^2] = \sigma^2. \tag{31}$$

Using the two checkpoints x^- and x^+, TSA returns

$$\begin{aligned} \widehat{x}_\alpha &= (1-\alpha)x^+ + \frac{\alpha}{2}\left(x^- + x^+\right) \\ &= \left(1 - \frac{\alpha}{2}\right)x^+ + \frac{\alpha}{2}x^-, \qquad \alpha \in [0,1]. \end{aligned} \tag{32}$$

Define $a_\alpha := 1 - \alpha/2$. Substitution gives

$$\widehat{x}_\alpha = (1 - a_\alpha\eta h)\, x^- - a_\alpha\eta\xi. \tag{33}$$

Consequently, the expected excess quadratic risk is exactly

$$R_\alpha(\eta) := \mathbb{E}[L(\widehat{x}_\alpha) - L(0)] = \frac{h}{2}\left[(1 - a_\alpha\eta h)^2\,(x^-)^2 + a_\alpha^2\eta^2\sigma^2\right]. \tag{34}$$

Corollary. Suppose η is selected from $[0, \eta_{\max}]$. Then

$$\eta_\alpha^\star = \Pi_{[0,\eta_{\max}]}\left(\frac{h(x^-)^2}{a_\alpha\left(h^2(x^-)^2 + \sigma^2\right)}\right). \tag{35}$$

If the optima for the raw final iterate and TSA are not clipped by $\eta_{\max}$, then

$$\eta_\alpha^\star = \frac{\eta_0^\star}{1 - \alpha/2}. \tag{36}$$

In particular, $\eta_\alpha^\star > \eta_0^\star$ for every $\alpha \in (0,1]$, and $\eta_1^\star = 2\eta_0^\star$.

Proof. Differentiating (34) yields

$$R'_\alpha(\eta) = h a_\alpha\left[-h(x^-)^2 + a_\alpha\eta\left(h^2(x^-)^2 + \sigma^2\right)\right], \tag{37}$$

while

$$R''_\alpha(\eta) = h a_\alpha^2\left(h^2(x^-)^2 + \sigma^2\right) > 0. \tag{38}$$

The unconstrained minimizer is therefore

$$\eta^\star_{\alpha,\mathrm{unc}} = \frac{h(x^-)^2}{a_\alpha\left(h^2(x^-)^2 + \sigma^2\right)}. \tag{39}$$

Projection onto $[0, \eta_{\max}]$ gives (35). Since $a_0 = 1$ and $a_\alpha = 1 - \alpha/2$, the interior ratio follows. □

Remark. In this minimal one-update model, $a_\alpha\eta_\alpha^\star$ is constant in α, so the jointly optimized minimum risk is also constant. The result therefore isolates a change in the preferred schedule rather than claiming an automatic improvement in the best attainable risk. Multi-step trajectories, longer checkpoint windows, and nonstationary late-training dynamics can break this exact rescaling equivalence.

E LLM DISCLOSURE

We used LLMs in three ways. First, essentially all experiments were set up and implemented with LLM coding, followed by a separate LLM review of code correctness. Second, we asked LLMs to critique manuscript drafts and incorporated suggestions when we judged them appropriate. Third, we used LLMs to find related papers; we read each paper before citing it to confirm its relation to our work.