BACKPROPAGATED OUTPUT MOMENTUM: RELOCATING OPTIMIZER HISTORY FROM PARAMETERS TO TASK SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizer momentum is usually stored as a parameter-sized moving average of past gradients, which makes history costly and fixes each past signal in the coordinates in which it was computed. We introduce Backpropagated Output Momentum (BOM), which instead stores a compact moving average of prediction errors at the model output and reprojects that history through the current network at every step. A batch-level analysis characterizes the information retained and omitted by this relocation, while the implementation preserves the current supervised gradient and can replace the first-moment component of several adaptive optimizers. As a plug-in for momentum-based optimizers, including ones that already compress their state, BOM reduces parameter-shaped optimizer state by 49.7–99.8% in three compositions and, averaged over three language backbones, paired step time by 4.0%. It also improves mean validation performance across language and vision fine-tuning, by 1.42 points in the primary five-task comparison. Language and vision pretraining studies, together with matched mechanism controls, further test the construction across output spaces and model scales.

1 INTRODUCTION

AdamW is a reliable default for modern neural-network training because it combines a smoothed update direction with coordinate-wise adaptive scaling (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). The smoothing is carried by a *first moment*: a dense exponential moving average (EMA) m_t of the gradients up to optimizer step t, stored beside every trainable tensor, next to a second-moment EMA v_t used for adaptive scaling. This design duplicates temporal state at every tensor: the two moments together occupy two parameter-shaped buffers, twice the memory of the parameters themselves and often a dominant part of the training footprint.

Memory is the visible cost; the second cost is structural. For the mini-batch loss at a step $\tau \le t$, $\mathcal{L}_\tau(\theta) = B^{-1}\sum_{i=1}^{B}\phi\big(f(\theta; x_{\tau,i}), y_{\tau,i}\big)$, the chain rule gives the batch gradient $g_\tau = B^{-1}\sum_{i=1}^{B} J_{\tau,i}^\top r_{\tau,i}$, where $J_{\tau,i} = \partial f(\theta; x_{\tau,i})/\partial\theta$ is the output Jacobian and $r_{\tau,i} = \partial\phi/\partial f$ the output residual of example i, with ϕ the per-example loss and f the network output (the logits). This sum admits the exact decomposition

$$g_\tau = \bar{J}_\tau^\top s_\tau + c_\tau, \qquad s_\tau = \tfrac{1}{B}\textstyle\sum_i r_{\tau,i}, \quad \bar{J}_\tau = \tfrac{1}{B}\sum_i J_{\tau,i}, \quad c_\tau = \tfrac{1}{B}\sum_i (J_{\tau,i} - \bar{J}_\tau)^\top (r_{\tau,i} - s_\tau), \tag{1}$$

where $s_\tau \in \mathbb{R}^{d_{\text{out}}}$ is the batch-mean prediction error, $\bar{J}_\tau$ the batch-mean Jacobian, and c_τ the within-batch covariance between example-specific Jacobians and residuals. The first term maps the shared output error back to parameter space; the second is the example-specific part lost when errors are averaged before that mapping. AdamW's first moment stores an average of *already-projected* gradients, $m_t = \sum_{\tau\le t} w_{t,\tau}(\bar{J}_\tau^\top s_\tau + c_\tau)$ with EMA weights $w_{t,\tau}$: each past task signal s_τ stays tied to the Jacobian $\bar{J}_\tau$ through which it was first projected, together with its covariance contribution. As training moves the parameters and replaces the samples, these historical projections drift away from the map the current model would apply to the current examples; Section 3.3 states this precisely and Appendix A.6.5 measures both sources of drift. Momentum factorization and state quantization (Park & Lee, 2025; Dettmers et al., 2022) reduce storage while keeping the accumulated history in parameter space.

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.

We ask *which parts of past gradients must be retained, and how compact can that history be?* We develop a batch-shared output history for supervised classification and language modeling, replacing dense first-order state with a compact residual EMA. In these tasks, output coordinates have stable meanings even while the model Jacobian changes. Our approach, *Backpropagated Output Momentum* (BOM), therefore keeps the history in task space and defers the projection: it stores one EMA of the task-space error, $q_t = \sum_{\tau \le t} w_{t,\tau}\, s_\tau$, and at each step transports the accumulated history through the current batch-mean Jacobian, $h_t = \bar{J}_t^\top q_t$. While AdamW averages projected gradients, BOM averages task-space signals and projects them only once, through the current model on the current examples. The complete update retains the current supervised gradient (Section 3.2), and Section 3.3 makes the comparison to AdamW exact at batch level.

The relocation changes both the update rule and the stored state: current reprojection omits the historical-projection term, and one task-space vector replaces every dense local first-moment tensor, reducing first-order state from $O(P)$ to $O(d_{\text{out}})$. The decomposition identifies this structural difference, whereas its effect on optimization is evaluated empirically. Only the first-order history is relocated: the second moment stays parameter-local because elementwise squaring does not commute with the vector–Jacobian map, so no exact relocation of the same form exists (Section 3.4).

Contributions. Our contributions are summarized as follows.

1. *Introduce a composable first-moment replacement.* A compact task-space EMA replaces parameter-space momentum history, with current reprojection implemented in one backward pass. The construction composes with AdamW, Adam-mini and GaLore while retaining their adaptive-scaling rules.
2. *Explain and verify historical-projection drift.* An exact batch-level decomposition separates historical-projection drift from example-specific covariance. Probe measurements verify the presence of drift, and matched temporal-kernel and own-step controls support the contribution of current reprojection.
3. *Quantify fine-tuning gains and scale-dependent resource efficiency.* Five-task macros improve across three natural-language processing (NLP) backbones and three optimizer compositions. RoBERTa-base measurements show optimizer-state and peak-memory reductions in all three compositions. Pretraining measurements quantify training-checkpoint savings and, together with a calibrated cost model, characterize the observed efficiency crossover.

2 Related Work

Adaptive, sign, and matrix-momentum optimization. Adam maintains exponential moving averages of gradients and squared gradients, and AdamW decouples weight decay from the adaptive update (Kingma & Ba, 2015; Loshchilov & Hutter, 2019). Lion applies sign-based updates from a learned momentum rule (Chen et al., 2023), Muon orthogonalizes matrix-valued momentum for hidden weights (Jordan et al., 2024), and M+Adam combines additive and multiplicative updates (Liang et al., 2026). These methods change how a parameter-space update is formed; BOM instead changes where first-order history is stored and which Jacobian expresses it.

Memory-efficient optimizer states. Adafactor and CAME factor adaptive statistics (Shazeer & Stern, 2018; Luo et al., 2023); GaLore projects gradients into low-rank subspaces (Zhao et al., 2024); SMMF and a low-rank momentum approximation factorize momentum tensors (Park & Lee, 2025; Wang et al., 2026). SCALE replaces the parameter-local first moment of the backbone matrices with gradient normalization, retaining parameter-space first-order momentum at the output layer and on one-dimensional parameters (Glentis et al., 2026). Adam-mini reduces the number of distinct learning rates by sharing one second-moment scalar per block (Zhang et al., 2025). BOM instead removes parameter-local first moments entirely: one $O(d_{\text{out}})$ task-space EMA, stored before backpropagation and reprojected through the current full-network Jacobian, replaces them, while local second-moment scaling is kept.

Transported momentum: manifold and natural-gradient methods. Riemannian optimization accounts for the geometry of parameter-space search directions (Absil et al., 2008; Bonnabel,

2013). Riemannian momentum variants transport accumulated directions between tangent spaces (Bécigneul & Ganea, 2019), while natural-gradient and Kronecker-factored approximate-curvature (K-FAC) methods precondition gradients using the Fisher information metric or its approximation (Amari, 1998; Martens & Grosse, 2015). In these momentum variants, the retained history remains parameter-dimensional. Closest in spirit is SPRING (Goldshlager et al., 2024), which refreshes a carried stochastic-reconfiguration solution by projecting it onto the equations sampled at the current step, so that history is re-expressed through the current Jacobian; its carried state is parameter-shaped and the refresh is a per-sample regularized least-squares projection. BOM keeps the same ordering with none of that machinery: one $O(d_{\text{out}})$ task-space EMA enters through the loss, and a single backward pass performs the reprojection (Section 3.5).

Objective-side corrections at the output. Logit adjustment addresses class imbalance through prior-based logit offsets, applied either after training or within the training loss (Menon et al., 2021). BOM uses a running average of output residuals as the detached coefficient of a logit-linear surrogate.

3 Backpropagated Output Momentum

Notation. At step t, example $i \in \{1, \dots, B\}$ of the batch has logits $z_{t,i} \in \mathbb{R}^{d_{\text{out}}}$ and label $y_{t,i}$. The logit-loss gradient is the output residual $r_{t,i}$, with batch mean $s_t = B^{-1} \sum_i r_{t,i}$. For parameters $\theta \in \mathbb{R}^P$, let $J_{t,i} = \partial z_{t,i}/\partial \theta$ and $\bar{J}_t = B^{-1} \sum_i J_{t,i}$. A subscript ℓ restricts a Jacobian or gradient to trainable tensor $\theta_\ell \in \mathbb{R}^{P_\ell}$, $\ell = 1, \dots, L$. Residuals and their EMA lie in task space $\mathbb{R}^{d_{\text{out}}}$; projected gradients lie in parameter space. For historical steps $\tau \le t$, the EMA weights are $w_{t,\tau} = (1-\beta_1)\beta_1^{t-\tau}$, where $\beta_1 \in [0, 1)$ is the first-order EMA decay: AdamW applies it to m_t, and BOM applies the same value to its task-space EMA (Section 3.2). Norms $\|\cdot\|$ and $\|\cdot\|_{\text{op}}$ are Euclidean and induced operator norms.

3.1 Intuition: keep the history, refresh its projection

BOM stores task errors and recomputes their parameter-space direction at each step. It maintains the residual EMA $q_t = \beta_1 q_{t-1} + (1-\beta_1) s_t$ and applies the current projection $\bar{J}_t^\top q_t$. Figure 1 contrasts this order with AdamW; Section 3.3 gives their exact batch-level difference. The current vector–Jacobian product reaches every trainable tensor despite the compact stored history.

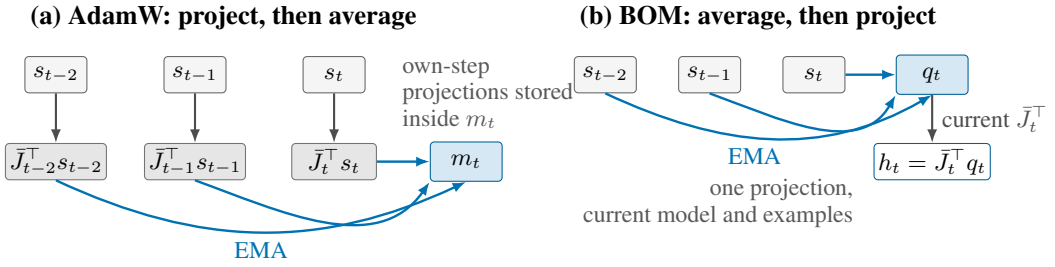

Figure 1: Where first-order history lives. AdamW projects each s_τ through its own step's model and sampled examples before averaging, storing historical projections inside m_t; BOM averages the task signals first and applies one projection through the current model and current examples, refreshing the same weighted history at every step (covariance terms omitted; Section 3.3 is exact).

How small the stored history is. For binary softmax classification, every residual $r_{t,i} = \text{softmax}(z_{t,i}) - \text{onehot}(y_{t,i})$ has zero coordinate sum, hence so do its batch mean s_t and the EMA q_t. The stored history is therefore a single scalar: $q_t = (a_t, -a_t)$ with $a_t = q_{t,1}$. Writing $\bar{J}_{t,k}^\top \in \mathbb{R}^P$ for the gradient of the batch-mean kth logit (the kth row of $\bar{J}_t$), $\bar{J}_t^\top q_t = a_t(\bar{J}_{t,1} - \bar{J}_{t,2})^\top = a_t \nabla_\theta [B^{-1} \sum_i (z_{t,i,1} - z_{t,i,2})]$. Here a_t is a running average of the batch-mean class-1 residual, i.e. of how far the mean predicted class-1 probability exceeded the class-1 label frequency in past

batches; the coefficient thus carries past label-conditioned prediction errors, while the parameter-space direction is the gradient of the current batch-mean logit margin under the current model (bias correction rescales a_t and leaves the direction unchanged). One scalar of first-order history reaches every trainable tensor, while the full current supervised gradient and the parameter-space second moment remain. This is a compact, batch-shared history, not per-example historical credit assignment or a fixed class-prior offset.

Assumption (stable output coordinates). The construction uses a fixed output dimension with stable coordinate meanings, as supplied by class labels in classification and vocabulary indices in language modeling. These shared coordinates allow residual history to accumulate across batches. Appendix A.6.5 measures how much residual magnitude survives batch averaging, and Section 4.6 evaluates the vocabulary-sized construction in language pretraining.

3.2 Formal update

For cross-entropy (CE), the per-example residual is $r_{t,i} = \mathrm{softmax}(z_{t,i}) - \mathrm{onehot}(y_{t,i})$, and $s_t = B^{-1}\sum_i r_{t,i}$ is its batch mean. BOM stores its EMA, with the decay β_1 of the notation paragraph, and its bias-corrected form,

$$q_t = \beta_1 q_{t-1} + (1-\beta_1)s_t, \qquad \hat{q}_t = \frac{q_t}{1-\beta_1^t}. \tag{2}$$

To feed the task-space EMA through standard reverse-mode autodiff, BOM forms a mixed scalar objective that blends the current cross-entropy loss $\mathcal{L}_t^{\mathrm{CE}}$ with a logit-linear surrogate whose coefficient is the detached EMA $\hat{q}_t$; the mixture weight λ is fixed at 0.5 in every reported BOM run except the labelled sensitivity diagnostics:

$$\mathcal{L}_t^{\mathrm{mix}} = (1-\lambda)\mathcal{L}_t^{\mathrm{CE}} + \frac{\lambda}{B}\sum_{i=1}^{B}\langle z_{t,i}, \hat{q}_t\rangle. \tag{3}$$

Because $\hat{q}_t$ is treated as a constant with respect to the logits, one backward pass through $\mathcal{L}_t^{\mathrm{mix}}$ returns, for every trainable tensor, the per-step update numerator

$$u_{\ell,t} = \nabla_{\theta_\ell}\mathcal{L}_t^{\mathrm{mix}} = (1-\lambda)g_{\ell,t} + \lambda\bar{J}_{\ell,t}^{\top}\hat{q}_t. \tag{4}$$

The surrogate is a computational device: it yields $\bar{J}_{\ell,t}^{\top}\hat{q}_t$ through ordinary backpropagation, reprojecting the residual history through the current model and examples without materializing a Jacobian. The mixture and EMA coefficients of each experiment are listed in Appendix B.

What $u_{\ell,t}$ contains. Substituting the recursion for q_t into Eq. (4) (with bias correction negligible) separates the current and historical components:

$$u_{\ell,t} = \left[(1-\lambda) + \lambda(1-\beta_1)\right]\bar{J}_{\ell,t}^{\top}s_t + (1-\lambda)\,c_{\ell,t} + \lambda\beta_1\,\bar{J}_{\ell,t}^{\top}q_{t-1}, \tag{5}$$

whose three terms are the current batch-mean signal, the current covariance ($c_{\ell,t}$ is the ℓth block of c_t in Eq. (1)), and the reprojected history. At the standard setting $\lambda = 0.5$, $\beta_1 = 0.9$, the batch-mean component gives total weight 0.55 to the current signal and 0.45 to the past history q_{t-1}, whose tail decays at 0.9, while the current covariance keeps weight 0.5; AdamW instead weights the current full gradient 0.10 and its accumulated history 0.90. Section 4.5 therefore runs two separate controls: an AdamW variant that matches only this $0.55/0.45$ mass split ($\beta_1 = 0.45$), and a parameter-space control that matches BOM's complete temporal kernel.

3.3 Current-Jacobian reprojection and historical projection drift

The contrast of Section 3.1 can be stated exactly at batch level. Recall from Eq. (1) that at any step τ the batch gradient decomposes as $g_\tau = \bar{J}_\tau^{\top}s_\tau + c_\tau$, with s_τ, $\bar{J}_\tau$ and c_τ the batch-mean residual, batch-mean Jacobian and within-batch covariance of that step. For two examples with residuals r and $-r$, the batch mean is zero, but $g = \frac{1}{2}(J_1 - J_2)^{\top}r$ can be nonzero. Averaging residuals therefore loses example-specific information even when the output coordinates are well defined. On a common parameter and batch trajectory with the same first-order decay, ignoring bias correction

and recalling the EMA weights $w_{t,\tau}$, AdamW's first-moment construction and BOM's historical component are respectively

$$m_t^{\mathrm{AdamW}} = \sum_{\tau=1}^{t} w_{t,\tau} \left(\bar{J}_\tau^\top s_\tau + c_\tau \right), \qquad h_t^{\mathrm{BOM}} = \bar{J}_t^\top \sum_{\tau=1}^{t} w_{t,\tau} s_\tau. \tag{6}$$

Their difference isolates what BOM declines to preserve in historical state,

$$m_t^{\mathrm{AdamW}} - h_t^{\mathrm{BOM}} = \sum_{\tau=1}^{t} w_{t,\tau} \left[(\bar{J}_\tau - \bar{J}_t)^\top s_\tau + c_\tau \right], \tag{7}$$

whose norm is at most $\sum_{\tau=1}^{t} w_{t,\tau} \big(\|\bar{J}_\tau - \bar{J}_t\|_{\mathrm{op}} \|s_\tau\| + \|c_\tau\| \big)$. Summing the first term inside the bracket of Eq. (7) over τ gives $D_t := \sum_{\tau=1}^{t} w_{t,\tau} (\bar{J}_\tau - \bar{J}_t)^\top s_\tau$, the *historical-projection drift*: the part of AdamW's history that remains expressed through earlier Jacobians. Its norm is bounded by the operator-norm terms of that inequality, and $\bar{J}_\tau - \bar{J}_t$ changes for two reasons, parameter movement and sample replacement. To measure it, we maintain along one training trajectory both the bias-corrected EMA of own-step projections and BOM's current reprojection of the same history,

$$\hat{M}_t = \frac{1}{1-\beta_1^t} \sum_{\tau=1}^{t} w_{t,\tau} \, \bar{J}_\tau^\top s_\tau, \qquad h_t = \bar{J}_t^\top \hat{q}_t, \qquad \hat{M}_t - h_t = \frac{D_t}{1-\beta_1^t}, \tag{8}$$

and report the cosine $\cos(\hat{M}_t, h_t)$ and the relative drift $\|\hat{M}_t - h_t\| / \|\hat{M}_t\|$. Averaged over telemetry steps, the five evaluation seeds and the five RoBERTa tasks, the mean cosine is 0.81 in the classification head but 0.52 in the encoder, with relative drift 0.46 and 0.79 (Appendix A.6.5); fixed probes separate the two sources (Table 22). These measurements establish a structural difference; Section 4.5 compares the two projection rules by training quality. The covariance term c_τ is distinct from drift: BOM omits it from task-space history but retains the full current gradient, including c_t, at weight $(1-\lambda)$. The decomposition thus identifies exactly which historical information the construction changes.

3.4 Adaptive scaling and state complexity

BOM retains a parameter-space second moment of its actual numerator, where $\beta_2 \in [0, 1)$ is its decay,

$$v_{\ell,t} = \beta_2 v_{\ell,t-1} + (1-\beta_2) u_{\ell,t}^{\odot 2}. \tag{9}$$

With the bias-corrected $\hat{v}_{\ell,t} = v_{\ell,t} / (1-\beta_2^t)$, learning rate η_t, denominator constant ϵ, and decoupled weight-decay coefficient ω_ℓ (zero for bias and normalization tensors, Appendix B.1), each trainable tensor follows

$$\theta_{\ell,t+1} = \theta_{\ell,t} - \eta_t \frac{u_{\ell,t}}{\sqrt{\hat{v}_{\ell,t}} + \epsilon} - \eta_t \omega_\ell \theta_{\ell,t}. \tag{10}$$

Eliminating the dense first moment $m_{\ell,t}$ halves the optimizer state: AdamW keeps two parameter-shaped buffers per tensor, whereas BOM keeps one plus a single d_{out}-vector, so the state ratio is approximately $\frac{1}{2}$ whenever $d_{\mathrm{out}} \ll P$ and both buffers use the same storage precision. This ratio counts optimizer state only; peak allocated memory also includes parameters, gradients, activations, and any task-space workspace. Appendix A.8.1 (Table 30) reports the measured optimizer state and peak allocated memory of the bf16 fine-tuning runs.

The asymmetry is structural: vector–Jacobian products are linear in the output signal, whereas elementwise squaring does not commute with that map, $(J^\top s)^{\odot 2} \neq J^\top (s^{\odot 2})$. Both optimizers use the same scaling form and coefficients, but AdamW accumulates $g_{\ell,t}^{\odot 2}$ and BOM accumulates $u_{\ell,t}^{\odot 2}$, so their realized preconditioners can differ; the experiments evaluate the complete substitution under this matched rule. Algorithm 1 in Appendix A gives the full step.

Appendix C gives a conditional descent and stationarity guarantee for this update, with explicit terms for history tracking, covariance, and other update perturbations.

3.5 Substituting BOM into a base optimizer

The construction produces $u_{\ell,t}$ with the same shape as the ordinary gradient, allowing substitution into other first-moment update rules.

The substitution. Let a base optimizer maintain a parameter-shaped first-moment accumulator and update

$$m_{\ell,t} = \beta_1 m_{\ell,t-1} + (1-\beta_1) g_{\ell,t}, \qquad \theta_{\ell,t+1} = \theta_{\ell,t} - \eta\, \Phi(\hat{m}_{\ell,t},\, \Sigma_{\ell,t})\,, \tag{11}$$

where $\Sigma_{\ell,t}$ collects the base optimizer's remaining state and $\Phi(a, \Sigma_{\ell,t})$ is its update map: the function that takes the first-moment numerator a together with that state and returns the parameter-shaped step. For AdamW, $\Sigma_{\ell,t} = \{v_{\ell,t}\}$ and $\Phi(a, \Sigma_{\ell,t}) = a/(\sqrt{\hat{v}_{\ell,t}} + \epsilon)$; Adam-mini uses the same form with a block-shared $\hat{v}$; GaLore applies it in projected coordinates and reconstructs (Appendix B.3.1). Here $\hat{m}_{\ell,t}$ includes the base optimizer's applicable first-moment bias correction; for AdamW it is $m_{\ell,t}/(1-\beta_1^t)$, and without such a correction it equals $m_{\ell,t}$. The substitution deletes $m_{\ell,t}$ and uses $\theta_{\ell,t+1} = \theta_{\ell,t} - \eta\, \Phi(u_{\ell,t}, \Sigma_{\ell,t})$, with no additional local-first-moment correction applied to $u_{\ell,t}$. The task-space EMA and retained second moment keep their own bias corrections. Thus one backward pass through $\mathcal{L}_t^{\text{mix}}$ supplies the numerator without a local first-moment accumulator.

Scope. The formula above uses a dense representation. For a base with projected first moments, its original projection maps $u_{\ell,t}$ into the base's update coordinates, where it replaces the first-moment numerator; reconstruction is retained. Remaining adaptive state keeps its construction and bias correction but is updated from the new numerator, so its realized values can differ between arms. A base without a first-moment accumulator is outside this substitution's scope. Section 4.4 evaluates three compositions, while Appendix A.6.8 tests a separate output-only second-moment diagnostic.

4 Experiments

4.1 Setup and reporting

We evaluate BOM on RoBERTa-base (Liu et al., 2019), DeBERTa-v3-base (He et al., 2023), and Qwen3-1.7B (Qwen Team, 2025) across five General Language Understanding Evaluation (GLUE) tasks (Wang et al., 2019): Matthews correlation for CoLA, F1 for MRPC/QQP, and accuracy for RTE/SST-2. The macro weights the five task means equally; one point is 0.01. Hyperparameter optimization (HPO) uses deterministic 90/10 partitions of each official training split and two HPO seeds, never the official validation split. Final runs start afresh, train five epochs on the full training split, and report the official-validation score at the fixed epoch-5 endpoint over five consecutive evaluation seeds disjoint from the HPO seeds. Unless stated otherwise, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and BOM uses AdamW-style scaling. Timing is excluded from selection and paired within task, seed, GPU model, and implementation stack (Appendix B).

4.2 Natural-language processing (NLP) transfer across three backbones

The five-task macro improves by 1.42 points on RoBERTa-base, 1.38 on DeBERTa-v3-base and 1.35 on Qwen3-1.7B; 13 of 15 task means improve (Table 5). All seeds are retained. One low DeBERTa AdamW SST-2 seed contributes 44.8% of that backbone's margin, so the aggregate reflects both endpoint quality and stability (Table 7). Appendix A.1 reports taskwise effects and uncertainty.

Table 1: Primary epoch-5 NLP results. Scores average five tasks and five evaluation seeds; taskwise intervals are in Appendix A.1. ΔTime is the mean paired bf16 step-time change of BOM relative to AdamW; with FP32 master weights, RoBERTa-base changes by −4.3%.

Backbone	AdamW	BOM	ΔScore	ΔTime
RoBERTa-base	0.7962	**0.8104**	+0.0142	−0.7%
DeBERTa-v3-base	0.8329	**0.8467**	+0.0138	−4.8%
Qwen3-1.7B	0.8169	**0.8304**	+0.0135	−6.4%
Coverage	**3/3 backbones: higher mean score, lower mean step time**			

On matched RoBERTa-base runs with FP32 master weights, optimizer state falls from 951.0 to 475.5 mebibytes (MiB) and peak allocated memory by 329 MiB; in bf16, state falls from 475.5 to 237.7 MiB and peak memory by 232 MiB (Table 30). In bf16, the paired mean step-time ratio is

0.993 on RoBERTa-base, 0.952 on DeBERTa-v3-base, and 0.936 on Qwen3-1.7B, a mean reduction of 4.0% (Table 1); with FP32 master weights, the RoBERTa-base ratio falls to 0.957 (−4.3%), as the removed buffer doubles in size.

4.3 Fixed-backbone optimizer comparison

Among six independently tuned fixed-backbone controls, BOM has the highest five-task validation macro (Appendix A.2, Figure 4). Its paired margin over AdamW is +0.0142 (descriptive 95% confidence interval (CI) [+0.0027, +0.0258], $p = 0.027$); the other five margins range from +0.0139 to +0.0513 and all remain significant at 0.05 after Holm correction. The appendix reports losses, state measurements, per-task results, and the retained degenerate SCALE seed; Figure 5 shows the training and validation trajectories.

4.4 Transfer: substituting BOM into other optimizers, and what it costs

Table 2 substitutes BOM into AdamW, block-adaptive Adam-mini, and low-rank GaLore. Their macros rise by 1.42, 1.23, and 1.14 points, respectively, with 14 of 15 task means improving. Parameter-shaped state falls by 50.0%, 99.8%, and 49.7%, while paired mean step time falls by 0.7% (4.3% with FP32 master weights), 2.0%, and 3.1%. Thus every composition improves all three mean estimates relative to its own base, including bases that already compress optimizer state.

Table 2: Pairwise BOM composition on RoBERTa-base: five-seed epoch-5 macros, parameter-shaped optimizer state (excluding the $O(d_{\text{out}})$ vector), and step-time changes paired within each base's implementation stack. Each base keeps a parameter-local first-moment EMA, which BOM replaces; Adam-mini shares v within blocks, and GaLore keeps its moments in a low-rank subspace. Macro-margin 95% CIs: AdamW [+0.0027, +0.0258], Adam-mini [−0.0203, +0.0449], GaLore [−0.0078, +0.0307]. All entries are measured in bf16; with FP32 master weights, the AdamW step-time change is −4.3%. Details: Appendices A.3, B.3.1 and B.8.

	Five-task macro			Optimizer state (MiB)		Step time
Base optimizer	Base	With BOM	Δ	Base	With BOM	Δ
AdamW	0.7962	**0.8104**	+0.0142	475.5	237.7	−0.7%
Adam-mini	0.7659	**0.7782**	+0.0123	238.2	0.42	−2.0%
GaLore	0.7739	**0.7854**	+0.0114	155.7	78.3	−3.1%

4.5 Mechanism ablations

Table 3 examines buffer removal, objective mixing, temporal weighting, projection choice, and adaptive scaling; Table 13 and Appendices A.6 and B.6 give questions, constructions and reporting protocols.

Table 3: RoBERTa-base mechanism controls (epoch-5 macro, five seeds; Δ from BOM's 0.8104). Details: Table 13.

Control	Macro	Δ	Control	Macro	Δ
CE, no local m	0.7841	−2.63	Own-step projection	0.7807	−2.97
Mixed + local m	0.7806	−2.98	BOM numerator, CE v	0.7892	−2.12
AdamW, $\beta_1 = 0.45$	0.8008	−0.96	Classifier-only history	0.7910	−1.94
Full-gradient kernel	0.7977	−1.28	$\beta_1 = 0$ (RTE acc.)	0.5235	4/5 deg.
APM anchor	0.7934	−1.70	Drift cos head/enc.	0.81/0.52	–

Temporal and objective controls. Under the matched HPO budget, removing the local first moment, mixing the objective, matching only the current/history mass, and matching the complete temporal kernel all finish below BOM (kernel: 95% CI [+0.66, +1.90]); none of these alone reproduces the margin.

Projection and scaling controls. Drift is larger in the encoder than the head, and independently tuned own-step projection finishes 2.97 points below current reprojection (95% CI $[+0.14, +5.80]$). The independently tuned multiclass STL10 control agrees in direction on both backbones: current reprojection leads by 0.79 points on ConvNeXt-Tiny (95% CI $[-1.21, +2.79]$) and 0.63 points on ViT-Tiny (95% CI $[-0.09, +1.35]$; Appendix A.7). Removing history makes four of five RTE seeds degenerate (Appendix A.6.2); numerator/second-moment cross-pairings and the output-only second-moment diagnostic further support retaining parameter-local adaptive scaling (Appendices A.6.4, A.6.7, and A.6.8).

History scope and precision. Full BOM exceeds the classifier-only historical control on all five task means and by 1.94 macro points under its protocol-selected configurations (95% CI $[-0.85, +4.74]$), consistent with a contribution of full-network history in this setting (Appendix A.6.6). With independently tuned 32-bit floating-point parameter/state storage and bf16 compute, the positive RoBERTa macro margin persists at $+1.67$ points (95% CI $[+0.73, +2.61]$; Appendix A.6.10).

4.6 Cross-domain validation: matched AdamW with and without BOM

Table 4 reports the matched AdamW–BOM comparison on STL10 fine-tuning on ConvNeXt-Tiny and ViT-Tiny (Coates et al., 2011; Liu et al., 2022; Dosovitskiy et al., 2021), ImageNet-1k pretraining on ResNet-50 (Deng et al., 2009; He et al., 2016), and Qwen3 language pretraining on Python code, FineWeb-Edu (Penedo et al., 2024), and C4 (Raffel et al., 2020). The independently tuned C4 comparisons span 55M, 110M, and 440M parameters with 3.0B, 5.0B, and 8.5B training tokens, and a 1.1B model is trained for 15.0B tokens with a preset peak learning rate (Figure 8). Other pretraining settings use five consecutive-seed reporting runs. C4-440M uses three such runs and one HPO seed per screening evaluation, with the same coarse grid, local refinement rule, and 10-evaluation budget, internal-holdout loss criterion and selection rule as the other language-pretraining studies and a screening horizon of 1.5B tokens; the arms are screened separately and both select 4×10^{-3}. The 1.1B study also uses three runs and fixes the peak learning rate at 10^{-3} for both arms before reporting.

Table 4: Matched AdamW–BOM cross-domain results; each group states its metric and Δ is the paired BOM$-$AdamW mean. Five consecutive-seed reporting runs except C4 440M and 1.1B ($n = 3$ each); pretraining entries are mean $\pm$ sample sd. Configurations: Appendices B.4 and B.5; paired intervals: Tables 11 and 12.

Setting	AdamW	BOM	Δ
Fine-tuning (vision), 40 epochs: validation accuracy $\uparrow$			
STL10, ConvNeXt-Tiny	0.9647	**0.9769**	$+0.0122$
STL10, ViT-Tiny	0.9567	**0.9669**	$+0.0102$
Pretraining (vision), 90 epochs: epoch-90 validation top-1 $\uparrow$			
ImageNet-1k, ResNet-50	0.7612 ± 0.0005	$\mathbf{0.7646 \pm 0.0008}$	$+0.0035$
Pretraining (language): final evaluation loss $\downarrow$			
Python code, Qwen3-55M, 3.0B tokens	1.7353 ± 0.0093	$\mathbf{1.7337 \pm 0.0106}$	-0.0016
FineWeb-Edu, Qwen3-55M, 3.0B tokens	$\mathbf{3.2931 \pm 0.0092}$	3.2948 ± 0.0045	$+0.0017$
C4, Qwen3-55M, 3.0B tokens	3.5587 ± 0.0105	$\mathbf{3.5506 \pm 0.0041}$	-0.0081
C4, Qwen3-110M, 5.0B tokens	3.7859 ± 0.0178	$\mathbf{3.7311 \pm 0.0034}$	-0.0547
C4, Qwen3-440M, 8.5B tokens	3.1336 ± 0.0004	$\mathbf{3.1292 \pm 0.0063}$	-0.0044
C4, Qwen3-1.1B, 15.0B tokens	3.0106 ± 0.0009	$\mathbf{3.0085 \pm 0.0008}$	-0.0021

BOM improves mean accuracy on both STL10 backbones and every ImageNet reporting run. On the three independently tuned C4 scales, it lowers mean final loss by 0.0081 at 55M, 0.0547 at 110M, and 0.0044 at 440M. The 110M paired interval excludes zero and all five runs improve; at 55M and 440M, three of five and two of three pairs improve, respectively, with intervals that include zero. In the 1.1B study with a preset peak learning rate, all three pairs improve and the mean final-loss difference is -0.0021. The Python-code and FineWeb-Edu mean differences are small and have opposite signs. Appendix Figures 7 and 8 report the loss trajectories.

Pretraining resource scaling. Figure 2 compares the C4 training measurements with the H800 single-step benchmark on a common parameter axis. From 55M through 1.1B, the mean paired C4 duration ratios, measured over complete runs at 55M and 110M and over matched intervals at 440M and 1.1B, are 1.066, 1.037, 1.032, and 0.996. At 1.1B, BOM saves 5.34 GiB of peak allocation and reduces checkpoint size from 12.29 to 8.19 GiB. The measured checkpoint sizes follow the FP32 tensor-count predictions, $12P$ bytes for AdamW and $8P$ for BOM. On H800, the step-time ratio crosses parity between 1B and 2B and reaches 0.938 at 4B, where peak-memory savings are 14.73 GiB. Calibrating the element-count model of Eq. (19) to the five H800 timing differences captures this shift from overhead to savings. Appendix B.8.1 gives the calibration; Appendix B.8 specifies the distinct timing protocols and matched C4 intervals.

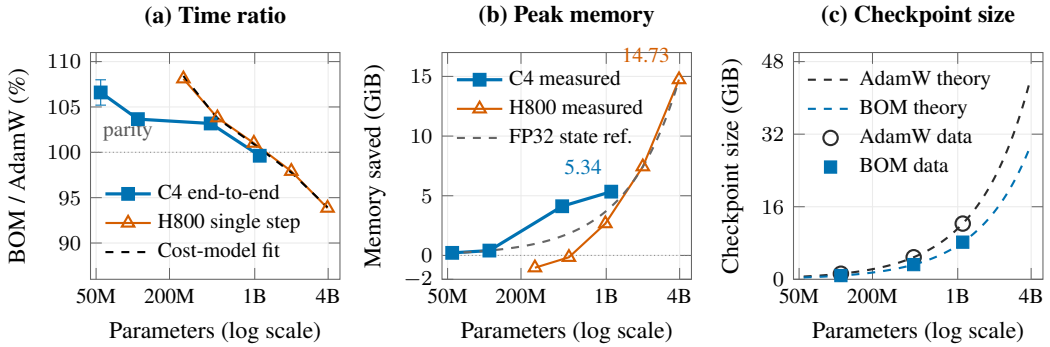

Figure 2: **Pretraining resource efficiency across scales.** (a) C4 end-to-end time ratios (paired mean $\pm$ SD), H800 single-step ratios, and the calibrated cost model (Appendix B.8.1). (b) Peak allocated-memory savings with the FP32 state reference $4(P - V)$ bytes. (c) Checkpoint sizes with FP32 predictions $12P$ (AdamW) and $8P$ (BOM) bytes. Details: Table 32, Figure 9, Appendix B.8.

5 Conclusion

BOM is a plug-in first-moment replacement for momentum-based optimizers: it relocates dense parameter-space history to an $O(d_{\text{out}})$ task-space EMA and reprojects it through the current network. An exact batch-level decomposition separates historical-projection drift from example-specific covariance; probe measurements establish the drift, and matched mechanism controls support the contribution of current reprojection. In fine-tuning, every composition, including those on bases that already compress state, cuts parameter-shaped state by 49.7–99.8% and lowers RoBERTa-base peak memory, while BOM also improves the five-task macro on all three NLP backbones and compositions. For pretraining, a calibrated cost model, the H800 ladder and C4 measurements locate where step-time overhead turns into savings and quantify peak-memory and checkpoint reductions. Together, these results position task-space history as a plug-in that makes momentum-based optimizers leaner and faster at scale, with better fine-tuning quality as an additional benefit.

6 Limitations

Language-pretraining quality reports held-out validation rather than an independent test set. The C4-440M and C4-1.1B comparisons each use three consecutive-seed reporting runs, and the 1.1B comparison fixes one shared, a priori learning rate rather than running method-specific HPO. The method assumes stable output coordinates; it retains within-batch covariance only through the current gradient, leaving changing output spaces and subgroup effects open. Systems timings are implementation-specific and use one unsharded device without gradient accumulation; distributed state, output, and communication costs require separate measurement (Appendix B.8.2).

AI USE STATEMENT

We used generative AI tools in four ways, matching the disclosures made at submission. (i) Writing assistance: English-language editing and polishing of the text and LaTeX restructuring. (ii) Retrieval and discovery: identifying related work and verifying bibliographic entries against their original sources. (iii) Research execution: the method and its design were conceived by the authors without generative AI; generative AI tools were used to implement the proposed optimizer and the experimental pipeline in code from the authors' specification, and for numerical and statistical checks of reported values, figure preparation, feedback on experimental reporting, and assistance in interpreting results. (iv) Drafting: drafting parts of the paper's sections, which the authors subsequently revised. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including the text, analyses, citations, and scientific claims produced with the aid of generative AI.

ETHICS STATEMENT

This work uses public benchmark datasets and involves no human subjects, personally identifying information, or new data collection. It changes optimizer state within standard training pipelines and introduces no identified application-specific risk. Dataset use and compute are documented in Appendix B.

REPRODUCIBILITY STATEMENT

Algorithm 1 specifies the optimizer. The released implementation provides BOM, its AdamW, Adam-mini and GaLore integrations, and the language-modeling construction, with training entry points for RoBERTa-base fine-tuning and Qwen3 pretraining. Appendix B documents the reported training settings, HPO policy, and selected hyperparameters; Appendix A gives task-level, control, ablation, precision, and aggregate pretraining results. Pretraining uses five consecutive-seed reporting runs except for the C4-440M and C4-1.1B studies, which each use three.

REFERENCES

P.-A. Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

Gary Bécigneul and Octavian-Eugen Ganea. Riemannian adaptive optimization methods. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1810.00760.

Silvère Bonnabel. Stochastic gradient descent on Riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013. doi: 10.1109/TAC.2013.2254619.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49205–49233, 2023. doi: 10.52202/075280-2140. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/9a39b4925e35cf447ccba8757137d84f-Abstract.html.

Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 215–223. PMLR, 2011. URL https://proceedings.mlr.press/v15/coates11a.html.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=shpkpVXzo3h.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Athanasios Glentis, Jiaxiang Li, Andi Han, and Mingyi Hong. Memory-efficient LLM pretraining via minimalist optimizer design. In *International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=prvGhNz39e.

Gil Goldshlager, Nilin Abrahamsen, and Lin Lin. A Kaczmarz-inspired approach to accelerate the optimization of neural network wavefunctions. *Journal of Computational Physics*, 516:113351, 2024. doi: 10.1016/j.jcp.2024.113351. URL https://arxiv.org/abs/2401.10190.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://openreview.net/forum?id=8gmWwjFyLj.

Xiaoyuan Liang, Sebastian Loeschcke, Mads Toftrup, and Anima Anandkumar. M+Adam: Low-precision training via additive–multiplicative optimization. In *International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2607.10611.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. URL https://arxiv.org/abs/1907.11692.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11976–11986, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Yang Luo, Xiaozhe Ren, Zangwei Zheng, Zhuo Jiang, Xin Jiang, and Yang You. CAME: Confidence-guided adaptive memory efficient optimization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4442–4453, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.243. URL https://aclanthology.org/2023.acl-long.243/.

James Martens and Roger Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417. PMLR, 2015. URL https://proceedings.mlr.press/v37/martens15.html.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=37nvvqkCo5.

Kwangryeol Park and Seulki Lee. SMMF: Square-matricized momentum factorization for memory-efficient optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(19): 19848–19856, April 2025. doi: 10.1609/aaai.v39i19.34186. URL https://ojs.aaai.org/index.php/AAAI/article/view/34186.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL https://arxiv.org/abs/2406.17557.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL https://www.jmlr.org/papers/v21/20-074.html.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4596–4604. PMLR, 2018. URL https://proceedings.mlr.press/v80/shazeer18a.html.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Zhengbo Wang, Jian Liang, Ran He, Zilei Wang, and Tieniu Tan. Taming momentum: Rethinking optimizer states through low-rank approximation. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=9Q0dNBYeEY.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P. Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2406.16793.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61121–61143. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhao24s.html.

A SUPPLEMENTARY RESULTS

This appendix reports every task-level result behind the aggregate claims of Section 4, together with the per-seed statistics and the control and ablation tables. Figure 3 first contrasts where the two optimizers hold their state, the structural picture summarized in Section 3.1, and Algorithm 1 lists the complete update step with AdamW-style scaling.

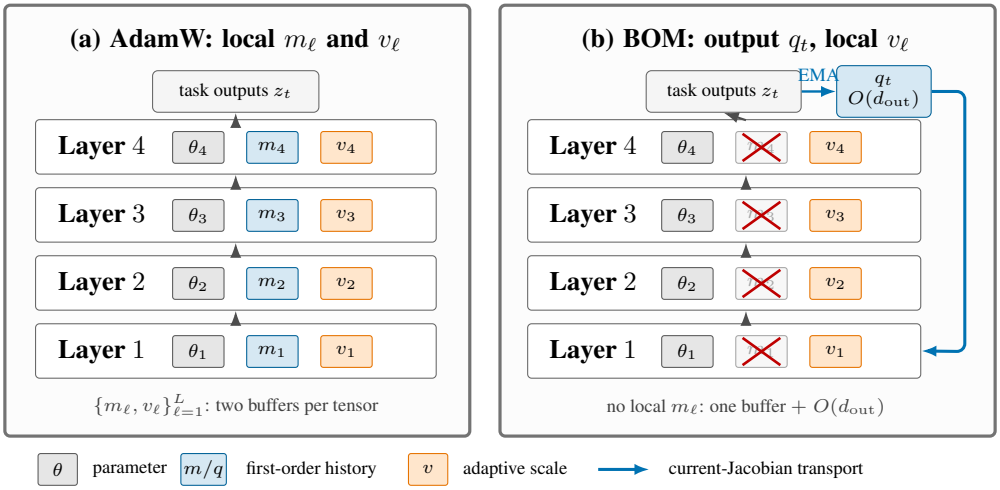

Figure 3: Optimizer-state placement in AdamW and BOM. AdamW retains dense first- and second-moment tensors for every trainable layer. BOM deletes every dense local first moment (crossed out), retains the local second moment used for adaptive scaling, and stores one compact task-space history q_t. The current backward graph re-expresses that history as updates for all trainable tensors, rather than preserving the parameter coordinates induced by earlier Jacobians. Parameter, gradient, and activation memory are omitted because the diagram isolates optimizer state.

Algorithm 1 Backpropagated Output Momentum with AdamW-style scaling

Require: Parameters θ_ℓ, step sizes η_t, coefficients $\lambda, \beta_1, \beta_2, \epsilon, \omega_\ell$.

Initialize one shared $q_0 = 0 \in \mathbb{R}^{d_{\text{out}}}$ and $v_{\ell,0} = 0$ for each trainable tensor.
for $t = 1, 2, \ldots$ **do**
  Let $\mathcal{I}_t$ index the supervised outputs of the batch, $N_t = |\mathcal{I}_t| > 0$: examples in classification, non-ignored target positions in language modeling. Compute their logits $z_{t,i}$, $i \in \mathcal{I}_t$, and mean cross-entropy $\mathcal{L}_t^{\text{CE}}$; all means below are over $\mathcal{I}_t$.
  $s_t \leftarrow N_t^{-1} \sum_{i \in \mathcal{I}_t} [\text{softmax}(\text{stopgrad}(z_{t,i})) - \text{onehot}(y_{t,i})]$.
  $q_t \leftarrow \beta_1 q_{t-1} + (1-\beta_1) s_t$; $\hat{q}_t \leftarrow q_t / (1-\beta_1^t)$; keep this state detached.
  $\mathcal{L}_t^{\text{mix}} \leftarrow (1-\lambda)\mathcal{L}_t^{\text{CE}} + \lambda N_t^{-1} \sum_{i \in \mathcal{I}_t} \langle z_{t,i}, \hat{q}_t \rangle$.
  One backward pass gives $u_{\ell,t} \leftarrow \nabla_{\theta_\ell} \mathcal{L}_t^{\text{mix}}$ for all trainable tensors.
  $v_{\ell,t} \leftarrow \beta_2 v_{\ell,t-1} + (1-\beta_2) u_{\ell,t}^{\odot 2}$; $\hat{v}_{\ell,t} \leftarrow v_{\ell,t} / (1-\beta_2^t)$.
  $\theta_\ell \leftarrow (1 - \eta_t \omega_\ell)\theta_\ell - \eta_t u_{\ell,t} / (\sqrt{\hat{v}_{\ell,t}} + \epsilon)$.
end for

A.1 NLP PER-DATASET RESULTS

Table 5 reports the NLP per-dataset results for the five-task main suite. Table 6 separates the descriptive RoBERTa-base task-level margins in panel (a) from the five-task macro estimate and descriptive interval in panel (b). On CoLA, RoBERTa-base improves Matthews correlation despite higher validation cross-entropy; DeBERTa-v3-base worsens on both measures, while Qwen3-1.7B improves on both. Matthews correlation summarizes discrete classification decisions, whereas cross-entropy also depends on predicted probabilities. Table 7 gives the per-seed validation scores behind the primary AdamW–BOM rows on all three backbones, with the paired difference and its interval for each task, so every mean can be read against its five constituent runs.

Table 5: Per-dataset results for AdamW and BOM on the three pretrained backbones, at the epoch-5 endpoint over the five evaluation seeds. Score is the metric named in each row; loss is cross-entropy.

Backbone	Dataset	Metric	Method	Train score	Val score	Train loss	Val loss
RoBERTa-base	CoLA	MCC	AdamW	0.8566	0.5743	0.1851	0.4777
RoBERTa-base	CoLA	MCC	BOM	0.9641	0.5970	0.0495	0.5794
RoBERTa-base	MRPC	F1	AdamW	0.9609	0.9077	0.1324	0.3240
RoBERTa-base	MRPC	F1	BOM	0.9862	0.9191	0.0616	0.3305
RoBERTa-base	QQP	F1	AdamW	0.8501	0.8463	0.2620	0.2683
RoBERTa-base	QQP	F1	BOM	0.9004	0.8626	0.1960	0.2481
RoBERTa-base	RTE	Acc	AdamW	0.7844	0.7155	0.4386	0.5894
RoBERTa-base	RTE	Acc	BOM	0.8906	0.7350	0.2936	0.5678
RoBERTa-base	SST-2	Acc	AdamW	0.9641	0.9372	0.0994	0.1975
RoBERTa-base	SST-2	Acc	BOM	0.9820	0.9385	0.0611	0.1873
DeBERTa-v3-base	CoLA	MCC	AdamW	0.8634	0.6628	0.1718	0.3906
DeBERTa-v3-base	CoLA	MCC	BOM	0.9531	0.6547	0.0698	0.4884
DeBERTa-v3-base	MRPC	F1	AdamW	0.9593	0.9102	0.1553	0.3382
DeBERTa-v3-base	MRPC	F1	BOM	0.9932	0.9246	0.0506	0.3223
DeBERTa-v3-base	QQP	F1	AdamW	0.8509	0.8597	0.2559	0.2518
DeBERTa-v3-base	QQP	F1	BOM	0.9103	0.8700	0.1693	0.2426
DeBERTa-v3-base	RTE	Acc	AdamW	0.9563	0.8123	0.1395	0.4962
DeBERTa-v3-base	RTE	Acc	BOM	0.9891	0.8318	0.0497	0.5100
DeBERTa-v3-base	SST-2	Acc	AdamW	0.9055	0.9195	0.2007	0.1812
DeBERTa-v3-base	SST-2	Acc	BOM	0.9844	0.9525	0.0560	0.1406
Qwen3-1.7B	CoLA	MCC	AdamW	0.9839	0.5596	0.0205	0.6892
Qwen3-1.7B	CoLA	MCC	BOM	0.9839	0.5804	0.0296	0.4849
Qwen3-1.7B	MRPC	F1	AdamW	1.0000	0.8841	0.0020	0.7610
Qwen3-1.7B	MRPC	F1	BOM	0.9994	0.9012	0.0080	0.4642
Qwen3-1.7B	QQP	F1	AdamW	0.9952	0.8730	0.0213	0.3986
Qwen3-1.7B	QQP	F1	BOM	0.9936	0.8852	0.0215	0.2874
Qwen3-1.7B	RTE	Acc	AdamW	0.9977	0.8137	0.0164	0.6187
Qwen3-1.7B	RTE	Acc	BOM	1.0000	0.8390	0.0116	0.4952
Qwen3-1.7B	SST-2	Acc	AdamW	1.0000	0.9539	0.0041	0.2184
Qwen3-1.7B	SST-2	Acc	BOM	0.9953	0.9461	0.0147	0.1962

Table 6: Paired BOM−AdamW validation-score margins on RoBERTa-base at the epoch-5 endpoint over the five evaluation seeds. Panel (a) reports the mean paired difference in each task's native metric with its descriptive paired 95% interval; panel (b) reports the five-task macro comparison of Section 4.3. Intervals use $t_{0.975,4} = 2.776$ on the five paired differences. Seeds are matched across the two arms, so the paired spread is the relevant one: on RTE each arm's own standard deviation (0.021 and 0.029) exceeds the margin, while the paired standard deviation is 0.0098. Per-seed values are in Table 7.

(a) Task-level paired margins

Dataset	Metric	Δ	95% CI
CoLA	MCC	$+0.0226$	$[-0.0226, +0.0679]$
MRPC	F1	$+0.0114$	$[-0.0079, +0.0306]$
QQP	F1	$+0.0163$	$[+0.0059, +0.0267]$
RTE	Acc	$+0.0195$	$[+0.0074, +0.0316]$
SST-2	Acc	$+0.0014$	$[-0.0116, +0.0144]$

(b) Aggregate comparison

Aggregation	Δ	95% CI
Five-task macro	$+0.0142$	$[+0.0027, +0.0258]$

Table 7: Per-seed validation scores for the primary AdamW–BOM comparison on all three backbones, at the epoch-5 endpoint. Score is the task-appropriate metric named with each block; the five evaluation seeds are disjoint from the two HPO seeds. Labels A–E identify the matched evaluation runs, and the Mean column is the value reported in Table 5. Δ is the mean paired difference with its descriptive 95% interval ($t_{0.975,4} = 2.776$ on the five paired differences); intervals use the variation of matched seed differences, which need not be smaller than either arm's marginal variation.

Task	Method	A	B	C	D	E	Mean
RoBERTa-base							
CoLA (MCC)	AdamW	0.5435	0.5778	0.6131	0.5834	0.5539	0.5743
	BOM	0.5885	0.5536	0.6107	0.6111	0.6209	0.5970
				Δ (95% CI): $+0.0226$ $[-0.0226, +0.0679]$			
MRPC (F1)	AdamW	0.9004	0.9127	0.9171	0.9084	0.8998	0.9077
	BOM	0.9255	0.9308	0.9020	0.9206	0.9164	0.9191
				Δ (95% CI): $+0.0114$ $[-0.0079, +0.0306]$			
QQP (F1)	AdamW	0.8457	0.8486	0.8456	0.8448	0.8468	0.8463
	BOM	0.8653	0.8537	0.8661	0.8706	0.8572	0.8626
				Δ (95% CI): $+0.0163$ $[+0.0059, +0.0267]$			
RTE (Acc)	AdamW	0.7148	0.7076	0.6895	0.7473	0.7184	0.7155
	BOM	0.7292	0.7184	0.7076	0.7834	0.7365	0.7350
				Δ (95% CI): $+0.0195$ $[+0.0074, +0.0316]$			
SST-2 (Acc)	AdamW	0.9381	0.9346	0.9346	0.9392	0.9392	0.9372
	BOM	0.9461	0.9358	0.9472	0.9243	0.9392	0.9385
				Δ (95% CI): $+0.0014$ $[-0.0116, +0.0144]$			
DeBERTa-v3-base							
CoLA (MCC)	AdamW	0.6751	0.6676	0.6556	0.6505	0.6654	0.6628
	BOM	0.7090	0.6178	0.6192	0.6946	0.6331	0.6547
				Δ (95% CI): -0.0081 $[-0.0622, +0.0460]$			
MRPC (F1)	AdamW	0.9091	0.9211	0.9074	0.9062	0.9069	0.9102
	BOM	0.9319	0.9244	0.9170	0.9283	0.9215	0.9246
				Δ (95% CI): $+0.0145$ $[+0.0041, +0.0248]$			
QQP (F1)	AdamW	0.8581	0.8585	0.8603	0.8597	0.8619	0.8597
	BOM	0.8749	0.8568	0.8756	0.8773	0.8652	0.8700
				Δ (95% CI): $+0.0103$ $[-0.0007, +0.0212]$			
RTE (Acc)	AdamW	0.7906	0.8159	0.8303	0.8159	0.8087	0.8123
	BOM	0.8556	0.8484	0.8014	0.8267	0.8267	0.8318
				Δ (95% CI): $+0.0195$ $[-0.0229, +0.0618]$			
SST-2 (Acc)	AdamW	0.9530	0.9495	0.9484	0.7993	0.9472	0.9195
	BOM	0.9530	0.9484	0.9553	0.9541	0.9518	0.9525
				Δ (95% CI): $+0.0330$ $[-0.0516, +0.1176]$			
Qwen3-1.7B							
CoLA (MCC)	AdamW	0.5608	0.6041	0.5074	0.5470	0.5786	0.5596
	BOM	0.6141	0.5765	0.5925	0.5674	0.5514	0.5804
				Δ (95% CI): $+0.0208$ $[-0.0408, +0.0824]$			
MRPC (F1)	AdamW	0.8744	0.8897	0.8789	0.8776	0.8998	0.8841
	BOM	0.9129	0.8862	0.8982	0.9005	0.9082	0.9012
				Δ (95% CI): $+0.0171$ $[-0.0024, +0.0367]$			
QQP (F1)	AdamW	0.8753	0.8706	0.8711	0.8753	0.8725	0.8730
	BOM	0.8731	0.8890	0.8888	0.8888	0.8864	0.8852
				Δ (95% CI): $+0.0123$ $[+0.0018, +0.0227]$			

Table 7: Per-seed validation scores (continued).

Task	Method	A	B	C	D	E	Mean
RTE (Acc)	AdamW	0.8267	0.8159	0.7978	0.8159	0.8123	0.8137
	BOM	0.8664	0.8484	0.8339	0.8123	0.8339	0.8390
					Δ (95% CI): $+0.0253$ $[+0.0035, +0.0470]$		
SST-2 (Acc)	AdamW	0.9530	0.9576	0.9599	0.9484	0.9507	0.9539
	BOM	0.9507	0.9369	0.9495	0.9484	0.9450	0.9461
					Δ (95% CI): -0.0078 $[-0.0179, +0.0023]$		

A.2 Fixed-Backbone RoBERTa Control Results

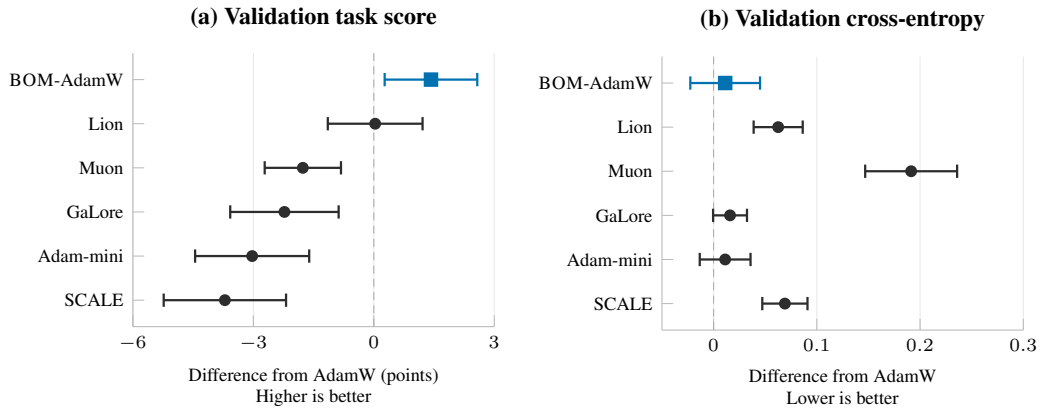

Figure 4: RoBERTa-base control effects relative to AdamW at the epoch-5 endpoint (Table 8). Each point is the mean method-minus-AdamW difference; bars are descriptive, unadjusted 95% paired t confidence intervals over five evaluation seeds ($df = 4$). For each seed, we first average the task-appropriate score or cross-entropy across the five tasks, then pair the resulting macro with AdamW at the same seed. Score differences are multiplied by 100; positive values favor the method. Negative cross-entropy differences favor the method. The dashed line marks no difference. These intervals describe seed variation conditional on the selected configurations and this fixed task set.

Table 9 gives the per-dataset RoBERTa-base results for the additional optimizer controls used in the fixed-backbone mechanism comparison; the matched AdamW and BOM rows are included for reference. The suite combines smaller datasets (CoLA, MRPC and RTE) with larger ones (SST-2 and QQP), while the vision and from-scratch studies extend coverage across domains and initialization regimes. The controls follow the shared selection policy in Appendix B; Muon additionally receives an update-scale search (Table 39). Implementations are documented in Appendix B.3.1.

Table 8: RoBERTa-base five-task macro endpoints for the seven methods in the fixed-backbone comparison, at the epoch-5 endpoint over the five evaluation seeds. Parameter-shaped optimizer state for the substitution pairs is reported with the composition comparison (Table 2).

Method	Train score	Val score	Train loss	Val loss
BOM-AdamW	0.9447	0.8104	0.1324	0.3826
AdamW	0.8832	0.7962	0.2235	0.3714
Lion	0.9302	0.7965	0.1340	0.4340
Muon	0.8833	0.7785	0.2029	0.5628
GaLore	0.8134	0.7739	0.3128	0.3873
Adam-mini	0.7967	0.7659	0.3406	0.3826
SCALE	0.8409	0.7591	0.2883	0.4405

Table 9: RoBERTa-base per-dataset results for every optimizer control and every BOM composition, at the epoch-5 endpoint over the five evaluation seeds. Score is the metric named in each row; loss is cross-entropy. Train quantities are measured on a fixed probe subset of the training set.

Dataset	Metric	Method	Train score	Val score	Train loss	Val loss
CoLA	MCC	AdamW	0.8566	0.5743	0.1851	0.4777
CoLA	MCC	BOM-AdamW	0.9641	0.5970	0.0495	0.5794
CoLA	MCC	Lion	0.9173	0.5780	0.0945	0.5178
CoLA	MCC	Adam-mini	0.6403	0.5348	0.3614	0.4639
CoLA	MCC	BOM-Adam-mini	0.7098	0.5478	0.3197	0.4568
CoLA	MCC	GaLore	0.6668	0.5495	0.3521	0.4386
CoLA	MCC	BOM-GaLore	0.6918	0.5561	0.3133	0.4607
CoLA	MCC	Muon	0.6628	0.5068	0.3568	0.4915
CoLA	MCC	SCALE	0.7209	0.5348	0.3058	0.5144
MRPC	F1	AdamW	0.9609	0.9077	0.1324	0.3240
MRPC	F1	BOM-AdamW	0.9862	0.9191	0.0616	0.3305
MRPC	F1	Lion	0.9938	0.9099	0.0350	0.3936
MRPC	F1	Adam-mini	0.9277	0.9073	0.2348	0.2956
MRPC	F1	BOM-Adam-mini	0.9285	0.9086	0.2453	0.3181
MRPC	F1	GaLore	0.9385	0.8948	0.1932	0.3572
MRPC	F1	BOM-GaLore	0.9480	0.9063	0.1787	0.3404
MRPC	F1	Muon	0.8700	0.8538	0.3947	0.4396
MRPC	F1	SCALE	0.9282	0.8763	0.2409	0.3853
QQP	F1	AdamW	0.8501	0.8463	0.2620	0.2683
QQP	F1	BOM-AdamW	0.9004	0.8626	0.1960	0.2481
QQP	F1	Lion	0.8284	0.8257	0.2910	0.2965
QQP	F1	Adam-mini	0.8185	0.8170	0.3256	0.3212
QQP	F1	BOM-Adam-mini	0.8264	0.8193	0.3275	0.3285
QQP	F1	GaLore	0.8448	0.8363	0.2808	0.2872
QQP	F1	BOM-GaLore	0.8526	0.8301	0.2767	0.2976
QQP	F1	Muon	0.9214	0.8737	0.1611	0.2469
QQP	F1	SCALE	0.8842	0.8349	0.2433	0.3288
RTE	Acc	AdamW	0.7844	0.7155	0.4386	0.5894
RTE	Acc	BOM-AdamW	0.8906	0.7350	0.2936	0.5678
RTE	Acc	Lion	0.9516	0.7372	0.1307	0.7726
RTE	Acc	Adam-mini	0.6531	0.6433	0.6188	0.6334
RTE	Acc	BOM-Adam-mini	0.6906	0.6787	0.5687	0.5989
RTE	Acc	GaLore	0.6648	0.6664	0.6186	0.6299
RTE	Acc	BOM-GaLore	0.7477	0.7025	0.5079	0.5751
RTE	Acc	Muon	0.9898	0.7155	0.0240	1.4303
RTE	Acc	SCALE	0.7039	0.6578	0.5258	0.6077
SST-2	Acc	AdamW	0.9641	0.9372	0.0994	0.1975
SST-2	Acc	BOM-AdamW	0.9820	0.9385	0.0611	0.1873
SST-2	Acc	Lion	0.9602	0.9319	0.1191	0.1893
SST-2	Acc	Adam-mini	0.9437	0.9271	0.1625	0.1988
SST-2	Acc	BOM-Adam-mini	0.9477	0.9367	0.1404	0.1933
SST-2	Acc	GaLore	0.9523	0.9227	0.1194	0.2238
SST-2	Acc	BOM-GaLore	0.9656	0.9319	0.0945	0.2010
SST-2	Acc	Muon	0.9727	0.9427	0.0779	0.2055
SST-2	Acc	SCALE	0.9672	0.8917	0.1256	0.3661
Macro mean	–	AdamW	0.8832	0.7962	0.2235	0.3714
Macro mean	–	BOM-AdamW	0.9447	0.8104	0.1324	0.3826
Macro mean	–	Lion	0.9302	0.7965	0.1340	0.4340
Macro mean	–	Adam-mini	0.7967	0.7659	0.3406	0.3826

Table 9: RoBERTa-base optimizer results (continued).

Dataset	Metric	Method	Train score	Val score	Train loss	Val loss
Macro mean	–	BOM-Adam-mini	0.8206	0.7782	0.3203	0.3791
Macro mean	–	GaLore	0.8134	0.7739	0.3128	0.3873
Macro mean	–	BOM-GaLore	0.8411	0.7854	0.2742	0.3750
Macro mean	–	Muon	0.8833	0.7785	0.2029	0.5628
Macro mean	–	SCALE	0.8409	0.7591	0.2883	0.4405

The comparison in the main text rests on the paired per-seed margins. On the five-task validation macro the paired per-seed margins favour BOM over every control. Against AdamW, the paired five-seed difference is $+0.0142$, with a descriptive 95% CI of $[+0.0027, +0.0258]$ and $t = 3.4$ ($p = 0.027$, uncorrected). The five remaining margins are $+0.0513$ over SCALE (95% CI $[+0.0363, +0.0663]$), $+0.0445$ over Adam-mini ($[+0.0204, +0.0687]$), $+0.0365$ over GaLore ($[+0.0180, +0.0549]$), $+0.0319$ over Muon ($[+0.0296, +0.0342]$), and $+0.0139$ over Lion ($[+0.0089, +0.0189]$); their uncorrected two-sided paired-t p-values (4 degrees of freedom) range from $< 10^{-4}$ (Muon) to 0.0069 (Adam-mini); the GaLore comparison has $p = 0.0054$, so all five remain significant at 0.05 after Holm correction. The corresponding means are in Table 9 and Table 1.

SCALE uses the authors' official implementation (Glentis et al., 2026). Its backbone-gradient normalization and parameter-group update rules follow that implementation. During the two-seed search, every rate at or above 1×10^{-4} sent all five tasks toward majority-class solutions; CoLA and RTE selected the smallest rate of the coarse ladder (Table 36). The selected rates were frozen before evaluation. Across the five evaluation seeds, one RTE seed remains degenerate and CoLA spans 14 points. The all-seed five-task mean is 0.7591, against 0.8104 for BOM and 0.7962 for AdamW.

The cross-entropy control without a local first moment finishes 1.21 points below AdamW (Table 15), a smaller drop than SCALE's. Because SCALE also changes gradient normalization, its gap does not isolate the effect of deleting first-moment history. These runs evaluate SCALE after learning-rate tuning for encoder fine-tuning; its original application was language-model pretraining from random initialization. The results compare the methods under the shared encoder fine-tuning protocol.

A.3 BOM Composition Across Three Optimizers

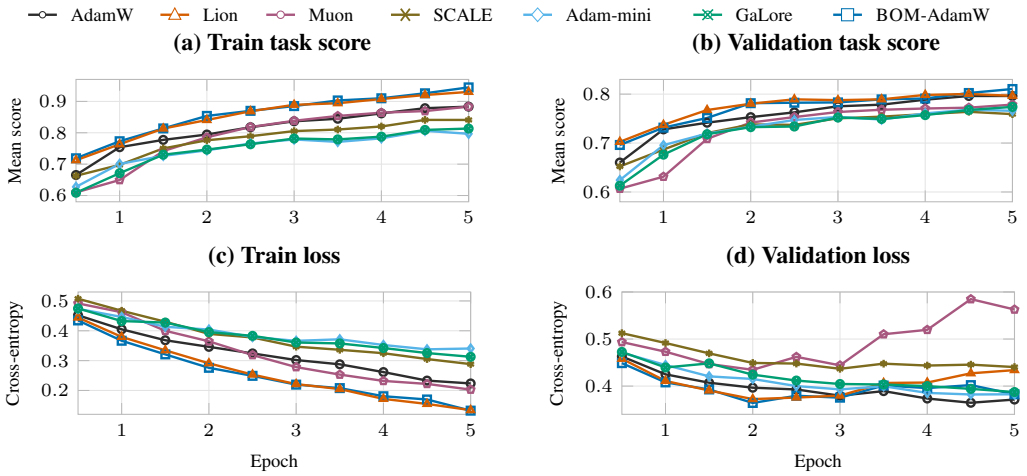

Figure 5: RoBERTa-base training and validation trajectories for the six baseline optimizers and BOM-AdamW. Each curve averages the five evaluation seeds within each task and then gives the five tasks equal weight. Training scores and losses use the training probe; validation scores use the task-appropriate metric and losses are cross-entropy. The logged intermediate evaluations describe training progress; the reported comparison uses the fixed epoch-5 endpoints in Table 8.

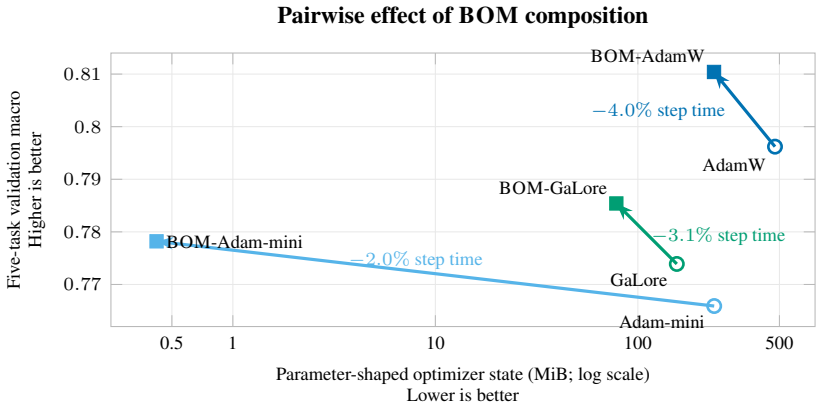

Figure 6: Pairwise effects of substituting BOM for the first-moment component of AdamW, Adam-mini and GaLore on RoBERTa-base. Each arrow starts at the base optimizer (open circle) and ends at its BOM composition (filled square); movement up and left denotes a higher five-task validation macro and less parameter-shaped optimizer state. The annotations give each method's mean paired bf16 step-time change over its main comparison: the three NLP backbones for AdamW, and RoBERTa-base for Adam-mini and GaLore. All three compositions move up and left and have lower mean step time, so each composition Pareto-dominates its own base on the three reported mean estimates. Averaged over the three methods' bf16 main comparisons, adding BOM raises the macro by 1.25 points and lowers paired step time by 3.0%. State bytes use the common accounting convention. Scores are epoch-5 means over five evaluation seeds. Full estimates and paired intervals are in Tables 2 and 30.

The composition study uses each base optimizer as its own reference, testing incremental savings across distinct update and state structures, including bases that already compress optimizer state.

Per-dataset results for the three pairs are in Table 9.

Figure 5 gives the seven-method training and validation trajectories alongside Figure 6, which summarizes the paired quality, optimizer-state and step-time changes for the three compositions. Detailed fine-tuning memory and timing measurements are grouped with the pretraining resource results in Appendix A.8.

GaLore composition and projection. The composition improves the mean macro score by 1.14 points, with a paired five-seed interval that includes zero (Table 2). Its state reduction removes both the projected first moment and the dense first moment in the unprojected branch. The surrogate numerator passes through the base optimizer's unchanged projection, so the retained direction still depends on its current basis. Implementation details are in Appendix B.3.1.

Table 10 compares the epochs needed to reach the epoch-5 AdamW training-probe macro of 0.8832. The interpolated BOM crossing is epoch 2.94. This diagnostic measures progress in the training-probe score; the final validation results and measured step times are reported separately.

Table 10: Training-probe score progress on RoBERTa-base, macro-averaged over five tasks and five evaluation seeds. The reference is AdamW's epoch-5 training-probe macro of 0.8832. The first BOM crossing is linearly interpolated between adjacent half-epoch measurements. This diagnostic uses only training-probe scores; the validation evaluations logged every half epoch are shown descriptively in Figure 5 and do not enter it, and the reported validation comparison uses the fixed epoch-5 endpoint.

Method	Epoch at reference training score
AdamW	5.00
BOM	**2.94**

A.4 Vision Transfer Results

Table 11 reports the STL10 transfer endpoints behind the cross-domain comparison of Section 4.6.

Table 11: STL10 transfer results for AdamW and BOM on two pretrained vision backbones, at the 40-epoch endpoint over the five evaluation seeds. Both arms share the same 40-epoch budget (Table 40) and are read under the same endpoint convention within each backbone pair. Score is accuracy; loss is cross-entropy. The paired BOM−AdamW mean accuracy differences and descriptive 95% t intervals are $+0.0122$ $[-0.0052, +0.0297]$ for ConvNeXt-Tiny and $+0.0102$ $[-0.0093, +0.0298]$ for ViT-Tiny. Intervals use the sample standard deviation of five matched seed differences and $t_{0.975,4} = 2.776$, without multiplicity correction.

Backbone	Method	Metric	Train score	Val score	Train loss	Val loss
ConvNeXt-Tiny	AdamW	Acc	0.9914	0.9647	0.0298	0.1487
ConvNeXt-Tiny	BOM	Acc	1.0000	0.9769	0.0007	0.0855
ViT-Tiny	AdamW	Acc	0.9930	0.9567	0.0255	0.1711
ViT-Tiny	BOM	Acc	0.9992	0.9669	0.0015	0.1262

A.5 From-Scratch Pretraining Results

Table 12 reports aggregate quality readings, and Figures 7 and 8 show validation trajectories. Configuration selection, budgets and data construction are specified in Appendix B.5. ImageNet reports epoch-90 validation accuracy; all language studies use final evaluation loss. C4-440M and C4-1.1B each use three consecutive-seed reporting runs; every other pretraining study uses five.

Table 12: Aggregate from-scratch pretraining endpoints, reported as mean ± sample sd over five consecutive-seed reporting runs except the C4 440M and 1.1B studies ($n = 3$ each). Δ is the paired mean difference (BOM − AdamW). ImageNet-1k reports epoch-90 top-1 and validation loss. For the language corpora, perplexity is computed as $\exp(\text{loss})$ for each run before averaging. All language rows use the final evaluation. Descriptive paired 95% intervals for the C4 loss differences are $[-0.0226, +0.0065]$ at 55M, $[-0.0766, -0.0329]$ at 110M, and $[-0.0199, +0.0111]$ at 440M. Budgets and schedules are in Table 42.

Setting	Metric	AdamW	BOM	Δ
ImageNet-1k, ResNet-50	Epoch-90 val top-1 ↑	0.7612 ± 0.0005	$\mathbf{0.7646 \pm 0.0008}$	$+0.0035$
	Val loss at that epoch ↓	0.9795 ± 0.0027	$\mathbf{0.9530 \pm 0.0011}$	-0.0265
Python code, Qwen3-55M	Final eval loss ↓	1.7353 ± 0.0093	$\mathbf{1.7337 \pm 0.0106}$	-0.0016
	Perplexity ↓	5.6706 ± 0.0529	$\mathbf{5.6619 \pm 0.0603}$	-0.0087
FineWeb-Edu, Qwen3-55M	Final eval loss ↓	$\mathbf{3.2931 \pm 0.0092}$	3.2948 ± 0.0045	$+0.0017$
	Perplexity ↓	$\mathbf{26.9262 \pm 0.2501}$	26.9723 ± 0.1221	$+0.0460$
C4, Qwen3-55M	Final eval loss ↓	3.5587 ± 0.0105	$\mathbf{3.5506 \pm 0.0041}$	-0.0081
	Perplexity ↓	35.1192 ± 0.3716	$\mathbf{34.8346 \pm 0.1437}$	-0.2846
C4, Qwen3-110M	Final eval loss ↓	3.7859 ± 0.0178	$\mathbf{3.7311 \pm 0.0034}$	-0.0547
	Perplexity ↓	44.0805 ± 0.7812	$\mathbf{41.7270 \pm 0.1416}$	-2.3536
C4, Qwen3-440M	Final eval loss ↓	3.1336 ± 0.0004	$\mathbf{3.1292 \pm 0.0063}$	-0.0044
	Perplexity ↓	22.9555 ± 0.0091	$\mathbf{22.8549 \pm 0.1432}$	-0.1006
C4, Qwen3-1.1B	Final eval loss ↓	3.0106 ± 0.0009	$\mathbf{3.0085 \pm 0.0008}$	-0.0021
	Perplexity ↓	20.3000 ± 0.0180	$\mathbf{20.2575 \pm 0.0166}$	-0.0425

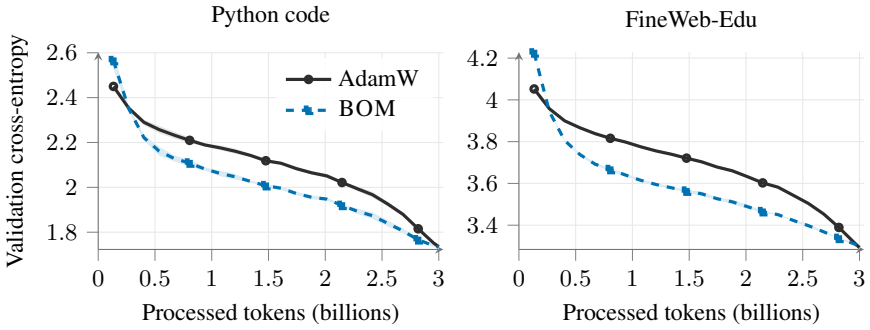

Figure 7: Qwen3-55M validation-loss trajectories on Python code and FineWeb-Edu. Each curve is the mean over five runs and each band is ±1 sample standard deviation; validation cross-entropy is evaluated on held-out batches. C4 validation-loss trajectories across four scales are reported separately in Figure 8.

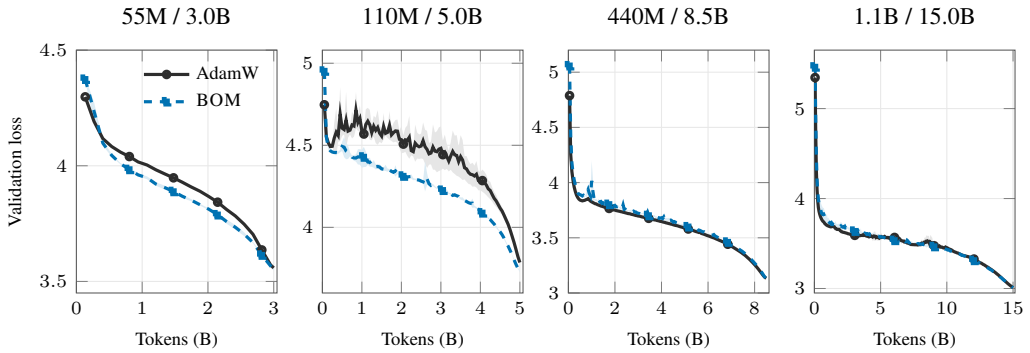

Figure 8: C4 validation-loss trajectories at four model and token-budget scales. Curves are means and bands are ± 1 sample standard deviation over five consecutive-seed reporting runs at 55M/110M and three at 440M/1.1B. The 1.1B setting uses 10^{-3} in both arms. Configurations are in Table 42; summary losses and perplexities are in Table 12.

A.6 RoBERTa Ablations and Diagnostics

Table 13 summarizes the mechanism controls of Section 4.5; the subsections below give their constructions and per-task results.

Table 13: RoBERTa-base mechanism controls. Most rows report the five-task epoch-5 macro over five seeds and their difference from BOM (0.8104); rows 4 and 7 report labeled diagnostics. The classifier-only row uses its protocol-selected configurations; rescue searches are reported in Appendix A.6.6. APM is the anchored per-tensor variant.

Question	Control	Reported quantity	Comparison / diagnostic	Reading
Is it buffer removal alone?	CE + no local m	0.7841	-2.63	no degenerate seeds
Is it the objective alone?	Mixed + local m	0.7806	-2.98	below AdamW
Is it just the current/history ratio?	AdamW $\beta_1 = 0.45$ $(0.55/0.45)$	0.8008	-0.96	vs. $\beta_1 = 0.9$: CI includes zero
Is historical projection drift real, and where?	$\cos(\hat{M}_t, h_t)$, head / encoder	0.81 / 0.52	–	rel. drift 0.46 / 0.79
Does a local scalar anchor help?	APM: per-tensor anchor, $+O(L)$ scalars	0.7934	-1.70	lower mean score
What changes under current reprojection?	same history, own-step projection	0.7807	-2.97	matched scaling rule; observed drop
Is the history itself load-bearing?	$\beta_1 = 0$ on RTE	0.5235	$4/5$ degenerate	$0/5$ at $\beta_1 = 0.9$; mixed objective retained
Is it the numerator alone?	BOM numerator, CE second moment	0.7892	-2.12	reverse pairing mostly degenerate
Is it the full temporal kernel alone?	full-gradient history, $\beta_1 = 0.9$	0.7977	-1.28	same kernel; lower mean score
Does the past-history term need the backbone?	past history on classifier tensors only	0.7910	-1.94	protocol-selected configurations; RTE seed-sensitive

Table 14: Update constructions behind the mechanism comparisons. g_t is the full CE gradient, $u_t = (1-\lambda)g_t + \lambda \bar{J}_t^\top \hat{q}_t$ is the BOM numerator, and $\widehat{\mathrm{EMA}}$ includes bias correction. The final column is the vector whose elementwise square feeds the second moment. The own-step buffer averages $\bar{J}_\tau^\top s_\tau$ and omits historical covariance; the full-kernel buffer averages g_τ and retains it. Each independently trained arm develops its own trajectory and adaptive statistics.

Construction	First-order history	Update numerator	Second-moment source
AdamW	Parameter-space full gradients	$\widehat{\mathrm{EMA}}(g)_t$	g_t
CE + no local m	None	g_t	g_t
Mixed + local m	Task residuals and parameter-space mixed updates	$\widehat{\mathrm{EMA}}(u)_t$	u_t
BOM	Task-space residual EMA $\hat{q}_t$	u_t	u_t
Own-step	Parameter-space mean-residual projections $\hat{M}_t$	$(1-\lambda)g_t + \lambda \hat{M}_t$	Own numerator
Full temporal kernel	Parameter-space full gradients $\hat{m}_t$	$(1-\lambda)g_t + \lambda \hat{m}_t$	Own numerator
Cell B	Task-space residual EMA	u_t	g_t
Cell C	Parameter-space gradient EMA; task residual EMA for scale	$\widehat{\mathrm{EMA}}(g)_t$	u_t

A.6.1 Objective–Momentum Factorial Control

Table 14 summarizes the update constructions behind the mechanism comparisons. Table 15 isolates the training objective and parameter-local first-moment storage. Both controls are independently tuned under the common protocol in Appendix B; every condition retains the parameter-local second moment.

Table 15: RoBERTa-base objective–momentum control. Entries are mean task-appropriate validation scores at epoch 5 over all five evaluation seeds. Both controls receive the same HPO budget as the primary pair; selected rates are in Table 43. Every condition retains parameter-local adaptive scaling and decoupled weight decay.

Dataset	Metric	AdamW	Mixed + local m	CE + no local m	BOM
CoLA	MCC	0.5743	0.5864	0.5635	**0.5970**
MRPC	F1	0.9077	0.9016	0.9011	**0.9191**
QQP	F1	0.8463	0.8452	0.8547	**0.8626**
RTE	Acc	0.7155	0.6318	0.7069	**0.7350**
SST-2	Acc	0.9372	0.9378	0.8945	**0.9385**
Macro mean	–	0.7962	0.7806	0.7841	**0.8104**

Interpretation. The mixed objective with the parameter-local buffer (*Mixed + local* m) reaches 0.7806 against AdamW's 0.7962, with the largest drop on RTE. In this arm, the local buffer averages numerators that already contain task-space history, adding parameter-space smoothing to the mixed update. Plain cross-entropy without the buffer (*CE + no local* m) removes first-order history and reaches 0.7841, 1.21 points below AdamW and 2.63 below BOM. It remains below BOM on all five tasks; its SST-2 mean is 0.8945, with no degenerate seed. Under this matched HPO protocol, neither control reproduces BOM's macro endpoint. SST-2 contributes about 0.88 of the 2.63 macro points, approximately 33.5%. The all-seed comparison captures training stability together with endpoint quality and evaluates the complete objective–momentum construction.

A.6.2 Fixed-Coefficient Sensitivity and Temporal-Mass Control

Mixing coefficient. Table 16 tests local sensitivity to the fixed mixture. The post-selection protocol is given in Appendix B.6.

Table 16: Post-selection local sensitivity of BOM's fixed mixing coefficient λ on RoBERTa-base. Entries are epoch-5 validation mean $\pm$ population standard deviation over the five evaluation seeds; Δ is relative to the paper configuration $\lambda = 0.5,\ \beta_1 = 0.9$, whose scores are 0.9191 on MRPC and 0.9385 on SST-2. Only non-default settings are shown.

Varied coefficient	Value	Fixed coefficient	MRPC F1	Δ	SST-2 Acc.	Δ
λ	0.25	$\beta_1 = 0.9$	0.9187 ± 0.0075	-0.0004	0.9353 ± 0.0019	-0.0032
λ	0.75	$\beta_1 = 0.9$	0.9156 ± 0.0076	-0.0035	0.9385 ± 0.0039	0.0000

The largest observed mean shift is 0.35 points within the tested λ neighborhood at $\beta_1 = 0.9$. These fixed-configuration results support local robustness of the mixture on the two tested tasks.

Task-space EMA decay: the full sweep. Table 17 varies β_1 alone over $\{0,\ 0.8,\ 0.9,\ 0.99\}$ on all five RoBERTa-base tasks, holding the $0.5/0.5$ mixture, BOM's selected learning rates and every other coefficient fixed, so that differences isolate the effect of β_1. At $\beta_1 = 0$ the bias-corrected $\hat{q}_t$ reduces to the current batch-mean residual s_t, so the mixed objective is retained but no history is kept. The $\beta_1 = 0.9$ row is the anchor for this diagnostic sweep.

Table 17: Task-space EMA decay β_1 across the five RoBERTa-base tasks, with the $0.5/0.5$ current/history mixture, the selected learning rates and all remaining coefficients held fixed. Entries are epoch-5 validation means over the five evaluation seeds; *deg.* counts seeds whose endpoint sits at the majority-class solution (0.5271 on RTE, 0.5092 on SST-2) with validation cross-entropy at chance, and no seed of any other task is degenerate at any setting. The $\beta_1 = 0.9$ row is the sweep anchor; the best entry in each column is bold.

	RTE (Acc.)		MRPC (F1)	SST-2 (Acc.)		CoLA (MCC)	QQP (F1)
β_1	score	deg.	score	score	deg.	score	score
0.00	0.5235	4/5	0.9057	0.8505	1/5	0.5848	0.8590
0.80	0.6383	2/5	**0.9210**	0.9378	0/5	0.5890	0.8581
0.90	0.7350	0/5	0.9191	**0.9385**	0/5	**0.5970**	**0.8626**
0.99	**0.7437**	0/5	0.9149	0.9358	0/5	0.5897	0.8529

CoLA, MRPC and QQP stay within 1.5 points of the sweep anchor at every setting. SST-2 has one degenerate seed at $\beta_1 = 0$; its four remaining seeds average 0.9358, while the table retains all five seeds in its reported mean. RTE shows the strongest dependence: its endpoint rises monotonically with β_1, and the number of seeds reaching the majority-class solution falls from four of five at $\beta_1 = 0$ to none at 0.9. Degeneracy therefore occurs on both RTE and SST-2, with the larger effect on RTE. Every cell uses BOM's selected task-specific learning rates; $\beta_1 = 0.9$ and 0.99 differ by at most one point on every task.

AdamW-only current/history ratio. The mass-matched control tests whether changing AdamW's current/history total weights explains the gain. Settings are specified in Appendix B.6.

Table 18: AdamW-only current/history-ratio control on RoBERTa-base. The main-experiment AdamW configuration is unchanged except for β_1, which moves from 0.9 to 0.45 and changes the asymptotic current/past total weights from $0.10/0.90$ to $0.55/0.45$. This matches the mass split of BOM's batch-mean component, while the past tail decays at 0.45 rather than 0.9. Per-dataset entries are epoch-5 means over the five evaluation seeds; the macro row averages the five task means. Score is the task-appropriate validation metric, loss is validation cross-entropy, and Δ is $\beta_1 = 0.45$ minus $\beta_1 = 0.9$.

Validation score				
Dataset	Metric	$\beta_1 = 0.9$	$\beta_1 = 0.45$	Δ
CoLA	MCC	0.5743	0.5827	+0.0083
MRPC	F1	0.9077	0.9039	−0.0038
QQP	F1	0.8463	0.8577	+0.0115
RTE	Acc.	0.7155	0.7191	+0.0036
SST-2	Acc.	0.9372	0.9408	+0.0037
Macro	–	0.7962	0.8008	+0.0047

Validation loss				
Dataset	Metric	$\beta_1 = 0.9$	$\beta_1 = 0.45$	Δ
CoLA	MCC	0.4777	0.4781	+0.0004
MRPC	F1	0.3240	0.3632	+0.0392
QQP	F1	0.2683	0.2558	−0.0126
RTE	Acc.	0.5894	0.6297	+0.0402
SST-2	Acc.	0.1975	0.1912	−0.0063
Macro	–	0.3714	0.3836	+0.0122

Interpretation. Matching the current/past mass split raises AdamW's validation-score macro from 0.7962 to 0.8008 (paired 95% CI for the change, $[-0.11, +1.04]$ points), while macro validation loss rises from 0.3714 to 0.3836. It does not reproduce BOM's 0.8104 endpoint. The past tail still decays at 0.45 rather than 0.9, so this control matches total mass rather than the complete temporal kernel; Table 19 tests the latter. Settings are in Appendix B.6.

A.6.3 Full Temporal-Kernel Parameter-History Control

The mass-ratio control changes the past-tail decay as well as the current/history split. To match the complete temporal kernel, we instead retain the full cross-entropy gradient history:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \qquad \hat{m}_t = m_t/(1-\beta_1^t), \quad m_0 = 0, \tag{12}$$

$$u_t^{\mathrm{FK}} = (1-\lambda) g_t + \lambda \hat{m}_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2)(u_t^{\mathrm{FK}})^{\odot 2}. \tag{13}$$

The update uses the bias-corrected second moment and decoupled weight decay, without another first-moment EMA on u_t^{FK}. With $\lambda = 0.5$ and $\beta_1 = 0.9$, the current coefficient is $0.5 + 0.05/(1-0.9^t)$ and the coefficient of lag $k \geq 1$ is $0.05\,0.9^k/(1-0.9^t)$. Thus it matches the temporal kernel of BOM's batch-mean component, including startup correction, while retaining full-gradient parameter-space history. Unlike the own-step control, this history also retains historical example-specific covariance. Each independently trained arm accumulates its second moment from its own numerator.

The control is independently tuned under the shared HPO policy; its final settings and history-buffer precision are recorded in Appendix B.6.

Table 19: Full temporal-kernel control on RoBERTa-base under the unified HPO policy (Appendix B.6). Scores are five-seed means from the one-shot final test after epoch 5. Δ is BOM minus the full-kernel control; intervals are descriptive paired 95% t intervals ($df = 4$), without multiplicity correction. The macro interval averages tasks within each seed before forming paired differences.

Task	Metric	AdamW	Full kernel	BOM	Δ	95% CI
CoLA	MCC	0.5743	0.5625	0.5970	$+0.0345$	$[+0.0053, +0.0636]$
MRPC	F1	0.9077	0.9209	0.9191	-0.0018	$[-0.0159, +0.0122]$
QQP	F1	0.8463	0.8584	0.8626	$+0.0042$	$[-0.0030, +0.0114]$
RTE	Acc.	0.7155	0.7235	0.7350	$+0.0116$	$[-0.0376, +0.0607]$
SST-2	Acc.	0.9372	0.9232	0.9385	$+0.0154$	$[-0.0115, +0.0423]$
Macro	–	0.7962	0.7977	0.8104	$+0.0128$	$[+0.0066, +0.0190]$

The full-kernel macro is 0.7977, compared with AdamW's 0.7962, the own-step control's 0.7807, and BOM's 0.8104. BOM has a higher mean on four tasks; MRPC favors the full-kernel control. All 25 full-kernel endpoints have nondegenerate predictions. These results disfavor temporal weighting alone as an explanation for the observed macro gain under this protocol; the descriptive paired macro interval excludes zero.

A.6.4 Historical-Projection Control

The control of Section 4.5 keeps BOM's task-space accumulation but projects each signal through the model and batch-mean Jacobian of its own step, maintaining a parameter-shaped $M_t = \beta_1 M_{t-1} + (1-\beta_1)\bar{J}_{\ell,t}^{\top} s_t$ and forming $u_{\ell,t} = (1-\lambda) g_{\ell,t} + \lambda \hat{M}_t$. Both arms share λ, β_1, β_2, ϵ, weight decay, and the AdamW-style adaptive-scaling construction; each updates its scale statistic from its own numerator. Evaluated on a common trajectory, the two numerator constructions differ by $\lambda(\hat{M}_t - \bar{J}_{\ell,t}^{\top}\hat{q}_t)$, the bias-corrected historical-projection drift, and neither accumulates covariance in its history. On the classifier bias, $\bar{J}_{b,t} = I$ implies $\hat{M}_t = \hat{q}_t$ within that trajectory, so both constructions reduce to $(1-\lambda)s_t + \lambda\hat{q}_t$. This is equality of update formulas: independently trained arms can develop different residual histories and scale statistics even when their selected learning rates agree. The own-step control uses the shared independent HPO protocol (Appendix B). It is a diagnostic requiring an extra backward pass and a dense buffer, so its step time and state are not compared with the compositions. For a linear model with a fixed batch, the Jacobian is constant and the two rules coincide when initialized identically and run at the same learning rate.

Table 20: Historical-projection control on RoBERTa-base, evaluated once after five epochs on each of the five evaluation seeds under the final-test convention of Appendix B. The control keeps BOM's task-space history but transports it through each step's own model and examples rather than the current ones. On a common trajectory, its numerator differs from BOM's by λ times the bias-corrected historical-projection drift term of Eq. (7). Both arms share the coefficients, temporal weighting, covariance treatment, AdamW-style adaptive-scaling rule and weight decay; each updates its scale statistic from its own numerator. Its independently selected per-task learning rates are listed in Table 36. Δ is the mean paired BOM-minus-control score difference. Intervals are descriptive, unadjusted 95% paired-t intervals over five matched evaluation runs ($df = 4$), using the sample standard deviation of the paired differences. Macro differences first average the five tasks equally within each matched run before estimating uncertainty.

Dataset	Metric	AdamW	Own-step control	BOM	Δ	Paired 95% CI
CoLA	MCC	0.5743	0.5632	**0.5970**	$+0.0338$	$[-0.1003, +0.1678]$
MRPC	F1	0.9077	0.9162	**0.9191**	$+0.0029$	$[-0.0113, +0.0171]$
QQP	F1	0.8463	0.8561	**0.8626**	$+0.0065$	$[-0.0044, +0.0173]$
RTE	Acc	0.7155	0.6440	**0.7350**	$+0.0910$	$[+0.0161, +0.1658]$
SST-2	Acc	0.9372	0.9241	**0.9385**	$+0.0144$	$[+0.0037, +0.0252]$
Macro mean	–	0.7962	0.7807	**0.8104**	$+0.0297$	$[+0.0014, +0.0580]$

Interpretation. The independently tuned current-reprojection endpoint is 2.97 macro points above the own-step control ($\Delta = +0.0297$, descriptive paired 95% CI $[+0.0014, +0.0580]$). RTE contributes $0.0910/5 = 0.0182$ to the macro difference, approximately 61.2% of the total. The macro interval and the RTE and SST-2 task intervals exclude zero; the other three task intervals include zero. These intervals are unadjusted for multiple comparisons. The control also lands 1.55 points below AdamW on the macro, whereas BOM finishes above it. The matched-HPO own-step–BOM pair compares historical and current projection under a common update family.

A.6.5 Measuring historical projection drift

The control of Appendix A.6.4 maintains both projection accumulations on one trajectory. At each telemetry step, $\hat{M}_t$ averages the task-space signals after their own-step projections, while $h_t = \bar{J}_t^\top \hat{q}_t$ reprojects the same residual history through the current model and examples. The latter is BOM's historical component, which enters its numerator as $(1-\lambda)g_t + \lambda h_t$. Their difference is the bias-corrected historical-projection drift vector on the same parameter and batch sequence, with no matching across runs required.

Table 21 reports the comparison split by depth over the five evaluation seeds, using telemetry steps from step 100 to the epoch-5 endpoint. Each task is weighted equally rather than each record, since the number of telemetry steps scales with dataset size.

The own-step accumulation stays closer to current reprojection in the classifier head than in the encoder: mean cosine 0.81 against 0.52, and relative drift $\|\hat{M}_t - h_t\|/\|\hat{M}_t\|$ of 0.46 against 0.79. These averages over the retained telemetry steps show a substantial encoder departure. The compact history remains d_{out}-dimensional while the current model and examples determine its parameter-space projection at every step.

The last column measures within-batch cancellation. The ratio $\|s_t\|/B^{-1}\sum_i \|r_{t,i}\|$ is the fraction of the average per-example residual magnitude retained by batch averaging; complete cancellation would make it zero. Its task means exceed 0.13 and average 0.25. This quantifies the magnitude that survives within each batch; the projection-drift statistics above measure how the resulting historical direction changes across steps.

What the drift is made of. The difference $\bar{J}_\tau - \bar{J}_t$ in Eq. (7) changes for two reasons at once: the parameters have moved, and the samples defining the mean Jacobian have been replaced. Table 22 separates them on BOM trajectories for all five RoBERTa tasks with two fixed probe batches and a fixed task-space direction, so that each single-factor comparison varies exactly one source. At lag 20, the mixed encoder cosine lies between 0.456 and 0.577 on every task. RTE and MRPC show substantial departure from both parameter movement (cosine 0.702–0.740) and sample replace-

Table 21: Historical-projection drift measured inside the own-step control on RoBERTa-base, over the five evaluation seeds and all telemetry steps from step 100 to the epoch-5 endpoint. $\hat{M}_t$ accumulates the task-space signals through each step's own model and examples; $h_t = \bar{J}_t^\top \hat{q}_t$ reprojects the same history through the current model and examples. *head* is the classification head, *encoder* every other trainable tensor. Cancellation is $\|s_t\|/B^{-1}\sum_i \|r_{t,i}\|$. Tasks are weighted equally in the mean.

	$\cos(\hat{M}_t, h_t)$		$\|\hat{M}_t - h_t\|/\|\hat{M}_t\|$		Cancellation
Task	head	encoder	head	encoder	ratio
CoLA	0.84	0.46	0.45	0.86	0.27
MRPC	0.63	0.26	0.71	0.96	0.28
QQP	0.81	0.58	0.47	0.75	0.25
RTE	0.98	0.84	0.20	0.50	0.14
SST-2	0.79	0.44	0.49	0.88	0.33
Mean	0.81	0.52	0.46	0.79	0.25

ment (0.655–0.686). On CoLA, SST-2 and QQP, sample replacement is larger over this horizon (0.475–0.619) than parameter movement (0.830–0.850); at longer lags the parameter-only cosine decreases consistently. The measured term is therefore historical-projection drift jointly induced by the model that expressed a signal and the examples on which it was expressed. Current reprojection removes both components together, while the fixed probes quantify their respective contributions to historical-projection drift.

Table 22: Decomposing historical-projection drift on BOM trajectories across the five RoBERTa-base tasks (the five evaluation seeds, probes every 20 steps from step 100 to the epoch-5 endpoint). Fixed probe batches A and B and a fixed unit task-space direction s^* define $v_X(\theta) = \bar{J}_X(\theta)^\top s^*$. *Batch-only* compares v_A and v_B at the same parameters; *parameter-only* compares v_A at parameters Δ steps apart; *mixed* changes both. Entries are encoder cosines averaged over seeds and probe steps; lower values indicate larger departure. The EMA at $\beta_1 = 0.9$ has a mean lag of nine steps.

Task	Batch-only	Param. $\Delta = 20$	Param. $\Delta = 100$	Param. $\Delta = 300$	Mixed $\Delta = 20$	Mixed $\Delta = 300$
RTE	0.686	0.740	0.588	0.349	0.577	0.294
MRPC	0.655	0.702	0.579	0.493	0.545	0.432
CoLA	0.619	0.830	0.704	0.622	0.559	0.480
SST-2	0.548	0.850	0.736	0.656	0.507	0.432
QQP	0.475	0.848	0.750	0.672	0.456	0.402

A.6.6 Classifier-Only Historical Contribution

This control tests whether the mechanism can be reduced to an output-level correction. Let Γ_{cls} retain the four weight and bias tensors under the RoBERTa classifier and set every other parameter coordinate to zero. The control restricts only the past-history contribution to those tensors:

$$u_t^{\mathrm{cls}} = (1-\lambda)g_t + \lambda\frac{1-\beta_1}{1-\beta_1^t}\bar{J}_t^\top s_t + \lambda\frac{\beta_1}{1-\beta_1^t}\Gamma_{\mathrm{cls}}\bar{J}_t^\top q_{t-1}. \tag{14}$$

The backbone remains trainable, and both the full-network supervised gradient and the current-step task-space term are unchanged. Each arm accumulates its parameter-local second moment from its own numerator. Training settings and the primary and rescue selection procedures are specified in Appendix B.6.

Table 23: Classifier-only past-history control on RoBERTa-base. Each row retains all five evaluation runs (A–E) at epoch 5. Score is Matthews correlation for CoLA, F1 for MRPC/QQP and accuracy for RTE/SST-2; loss is cross-entropy. *Primary* marks the protocol-selected configurations used in the primary macro and *Rescue* the configurations selected by the three-HPO-seed rescue searches. Selection rules are in Appendix B.6.

Task	LR	A	B	C	D	E	Mean score	Mean loss	Status
CoLA	2.5×10^{-5}	0.6407	0.4624	0.6547	0.5417	0.6007	0.5800	0.4775	Primary
MRPC	2.0×10^{-5}	0.9242	0.9078	0.9158	0.8538	0.9154	0.9034	0.3237	Primary
MRPC	3.0×10^{-5}	0.7722	0.9154	0.9007	0.9061	0.9144	0.8818	0.4517	Rescue
QQP	3.0×10^{-5}	0.8633	0.8561	0.8631	0.8627	0.8292	0.8549	0.2624	Primary
QQP	2.0×10^{-5}	0.8537	0.8440	0.8588	0.8396	0.8556	0.8503	0.2694	Rescue
RTE	2.0×10^{-5}	0.5632	0.7365	0.7401	0.7978	0.5740	0.6823	0.5960	Primary
RTE	1.5×10^{-5}	0.7545	0.7076	0.7148	0.7112	0.6354	0.7047	0.6008	Rescue
RTE	1.0×10^{-5}	0.6751	0.5560	0.6751	0.6715	0.5993	0.6354	0.6413	Rescue
SST-2	3.0×10^{-5}	0.9346	0.9381	0.9392	0.9415	0.9186	0.9344	0.1939	Primary

Selection and interpretation. The primary score uses the protocol-selected configurations. Rescue searches on MRPC (one), QQP (one) and RTE (three, yielding two distinct alternative rates) use a three-HPO-seed mean; none replaces a primary configuration (Appendix B.6). Full BOM exceeds the protocol-selected control on all five task means and by 1.94 macro points (descriptive paired 95% CI $[-0.85, +4.74]$). The rescue configurations do not recover full BOM performance. No selected endpoint predicts only one class: QQP has one weaker run, and two RTE runs have near-majority predictions while still using both classes. Across the tested tasks, these results are consistent with a contribution from full-network history.

A.6.7 Numerator and Second Moment

BOM replaces two things at once relative to AdamW: the numerator handed to the update map, and the tensor whose elementwise square accumulates into $v_{\ell,t}$. Table 24 reports the two off-diagonal cells of that factorial. Cell B keeps BOM's numerator $u_{\ell,t}$ but builds $v_{\ell,t}$ from the plain cross-entropy gradient; cell C hands AdamW's first-moment numerator a second moment built from $u_{\ell,t}$. Both need $\nabla_\theta \mathcal{L}_t^{\mathrm{CE}}$ and $\nabla_\theta \mathcal{L}_t^{\mathrm{mix}}$ in the same step, so they take one forward pass and two backward passes; like the control of Appendix A.6.4 they are diagnostics rather than proposed optimizers.

These cross-pairings include baseline-rate checks and targeted learning-rate searches, whose candidates and selection rule are specified in Appendix B.6. Changing the source of $v_{\ell,t}$ changes per-coordinate effective step sizes, so a shared nominal learning rate does not hold update magnitudes fixed.

Table 24: Numerator/second-moment cross-pairings on RoBERTa-base. Entries are epoch-5 means over all five evaluation seeds, including degenerate runs. Cell B combines BOM's numerator with the CE-gradient second moment; cell C combines AdamW's numerator with the BOM-numerator second moment. Search rules and final rates are in Appendix B.6 and Table 43.

			Cell B: BOM numerator CE second moment		Cell C
Dataset	AdamW	BOM	at BOM's rate	at 3×10^{-5}	AdamW numerator BOM v
CoLA	0.5743	**0.5970**	0.5501	0.5565	0.1146
MRPC	0.9077	**0.9191**	0.9134	0.9134	0.8551
QQP	0.8463	**0.8626**	0.8427	0.8427	0.5175
RTE	0.7155	**0.7350**	0.6578	0.6968	0.5747
SST-2	0.9372	**0.9385**	0.9365	0.9365	0.6716
Macro mean	0.7962	**0.8104**	0.7801	0.7892	0.5467
Δ vs. BOM	-0.0142	–	-0.0304	-0.0212	-0.2637

Targeted searches. Cell B's targeted-search column changes only CoLA and RTE and reuses MRPC, QQP and SST-2; cell C includes an additional CoLA stability screen. These targeted searches are separate from the common optimizer-comparison budget.
Interpretation. All 25 cell B runs finish without degenerate endpoints. At 3×10^{-5}, cell B ends 2.12 macro points below BOM and 0.70 below AdamW. Cell C has 15 majority-class endpoints among its 25 runs (CoLA 4/5, MRPC 3/5, QQP 2/5, RTE 3/5, SST-2 3/5), leaving its all-seed macro 26.4 points below BOM. On CoLA, the additional two-seed screening returns Matthews correlation 0.0000 at all eight rates from 1×10^{-6} to 5×10^{-5}. The complete BOM construction outperforms these tested cross-pairings, supporting the joint use of its numerator and adaptive scale statistic.

A.6.8 Strict Output-Only Second-Moment Diagnostic

The strict output root-mean-square (OutRMS) variant replaces parameter-space $v_{\ell,t-1}$ with a task-space statistic. For per-example residuals $r_{t,i}$ it stores $R_t = \beta_2 R_{t-1} + (1-\beta_2) B^{-1} \sum_i r_{t,i}^{\odot 2}$ and uses $\hat{R}_t = R_t/(1-\beta_2^t)$. Its detached per-example output signal is

$$\tilde{r}_{t,i} = \frac{(1-\lambda) r_{t,i} + \lambda \hat{q}_t}{\sqrt{\hat{R}_t} + \epsilon}, \tag{15}$$

whose batch-mean logit-linear surrogate produces one vector–Jacobian product through the current graph. Thus the implementation averages squared per-example residuals rather than squaring their batch mean, and it keeps no parameter-shaped optimizer state. This two-dataset, two-seed diagnostic produces degenerate endpoints under the tested configuration (Table 25); it supports retaining parameter-local scale information in the reported construction but does not rule out other compressed second moments.

Table 25: Strict output-only second-moment diagnostic on RoBERTa-base at the fixed epoch-5 endpoint. Means use two paired seeds under the primary fine-tuning schedule. BOM-FullV retains parameter-space $v_{\ell,t-1}$; BOM-OutRMS replaces it with a task-space second-moment EMA and backpropagates the resulting RMS-normalized task signal. The output-only variant removes all parameter-shaped optimizer state while retaining task-space EMA state, but stays near a one-class/chance solution.

Task	FullV Train	OutRMS Train	FullV Val	OutRMS Val	ΔVal	OutRMS parameter-shaped state (MiB)
RTE	95.46	49.44	75.45 Acc	49.28 Acc	-26.17 pt	0
MRPC	98.28	50.00	92.15 F1	40.61 F1	-51.54 pt	0

A.6.9 ANCHORED-VARIANT ABLATION RESULTS

The anchored per-tensor momentum variant (APM) tests whether a minimal local reference improves the transported output-momentum direction. For tensor θ_ℓ, it fixes one anchor coordinate i_ℓ and maintains

$$a_{\ell,t} = \beta_1 a_{\ell,t-1} + (1-\beta_1) u_{\ell,t}[i_\ell]. \tag{16}$$

With $a_{\ell,0} = 0$, it uses the bias-corrected anchor $\hat{a}_{\ell,t} = a_{\ell,t}/(1-\beta_1^t)$. The reported $K = 1$ first-coordinate variant forms the exact clipped least-squares scalar and applies it to the BOM numerator,

$$r_{\ell,t} = \mathrm{clip}\left(\frac{\hat{a}_{\ell,t} u_{\ell,t}[i_\ell]}{\max\{u_{\ell,t}[i_\ell]^2, 10^{-12}\}}, 0.1, 3.0\right), \qquad u_{\ell,t}^{\mathrm{APM}} = r_{\ell,t} u_{\ell,t}. \tag{17}$$

The ablation retains BOM's current reprojection and adds $O(L)$ scalar state for L trainable tensors; it does not store an own-step parameter-space direction. Table 26 expands its results by dataset; the BOM–APM aggregate appears in Table 13.

Table 26: RoBERTa-base anchored APM variant by dataset. The corresponding BOM aggregate is reported in Table 13. Entries are epoch-5 endpoints over the five evaluation seeds. Score denotes Matthews correlation for CoLA, F1 for MRPC and QQP, and accuracy for RTE and SST-2; loss is validation cross-entropy.

Dataset	Metric	Val score	Val loss
CoLA	MCC	0.5508	0.5313
MRPC	F1	0.9140	0.3737
QQP	F1	0.8547	0.2607
RTE	Acc	0.7105	0.6167
SST-2	Acc	0.9369	0.1982
Macro mean	–	0.7934	0.3961

A.6.10 FP32-STATE REPLICATION

This replication tests whether the primary comparison persists with FP32 parameter and optimizer-state storage and bf16 compute. Both arms are independently tuned under the shared policy; selected rates and settings are in Appendix B.6.

Table 27 reports the resulting endpoints. BOM has the higher task-appropriate mean score on four of five tasks, increasing the all-seed macro from 0.8055 to 0.8222. QQP is the exception, with means of 0.8809 for AdamW and 0.8777 for BOM, whereas the other four score margins range from $+0.0094$ to $+0.0361$. Validation cross-entropy decreases on four tasks and on the macro, from 0.4558 to 0.3956; AdamW is lower on QQP, 0.2505 against 0.2587. This replication therefore shows that the positive macro comparison is not specific to bf16 parameter and optimizer-state storage.

The five-seed macro margin has a descriptive paired 95% interval of $[+0.0073, +0.0261]$ ($t = 4.935$, $p = 0.0078$). Table 28 reports the task-level and macro paired intervals at this same endpoint.

The cross-entropy ordering differs between the bf16 and FP32-state comparisons. The two comparisons are tuned independently; on QQP, for example, the selected BOM rate is 3×10^{-5} in bf16 and 2×10^{-5} with FP32 state. Cross-entropy measures predicted probabilities as well as classification decisions, so a score gain need not imply a loss reduction. The replication confirms the positive macro result with FP32 parameters and optimizer state.

Table 27: FP32-state replication of the RoBERTa-base AdamW–BOM comparison. Both arms follow the shared fine-tuning protocol and are evaluated once after epoch 5 on the official validation split used as the final test (Appendix B) over the five evaluation seeds. Scores use the task metric shown; losses are validation cross-entropy. Entries are mean ± sample standard deviation over five seeds, and Δ is the paired BOM−AdamW mean. The macro score margin has paired 95% CI $[+0.0073, +0.0261]$ ($p = 0.0078$). Task-level paired intervals in Table 28 use the same endpoint. All 50 endpoints are nondegenerate. BOM has the higher mean score on four of five tasks and a five-task macro of 0.8222 versus AdamW's 0.8055; its validation cross-entropy is lower on four tasks and on the macro, while AdamW is lower on QQP.

Validation score

Dataset	Metric	AdamW	BOM	Δ
CoLA	MCC	0.5789 ± 0.0337	$\mathbf{0.6150} \pm 0.0176$	$+0.0361$
MRPC	F1	0.9048 ± 0.0058	$\mathbf{0.9186} \pm 0.0075$	$+0.0138$
QQP	F1	$\mathbf{0.8809} \pm 0.0074$	0.8777 ± 0.0070	-0.0032
RTE	Acc	0.7350 ± 0.0161	$\mathbf{0.7625} \pm 0.0065$	$+0.0274$
SST-2	Acc	0.9278 ± 0.0047	$\mathbf{0.9372} \pm 0.0073$	$+0.0094$
Macro mean	–	0.8055 ± 0.0047	$\mathbf{0.8222} \pm 0.0053$	$+0.0167$

Validation loss

Dataset	Metric	AdamW	BOM	Δ
CoLA	MCC	0.6300 ± 0.0812	$\mathbf{0.4857} \pm 0.0425$	-0.1443
MRPC	F1	0.4287 ± 0.0589	$\mathbf{0.3898} \pm 0.0309$	-0.0389
QQP	F1	$\mathbf{0.2505} \pm 0.0061$	0.2587 ± 0.0176	$+0.0081$
RTE	Acc	0.7355 ± 0.0779	$\mathbf{0.6239} \pm 0.0441$	-0.1116
SST-2	Acc	0.2341 ± 0.0232	$\mathbf{0.2198} \pm 0.0317$	-0.0143
Macro mean	–	0.4558 ± 0.0345	$\mathbf{0.3956} \pm 0.0185$	-0.0602

Table 28: FP32-state replication: paired validation-score margins from the one-shot final test after epoch 5, matching Table 27 and the protocol in Appendix B. Intervals are descriptive, unadjusted 95% paired-t intervals over five matched evaluation seeds ($df = 4$). Macro uncertainty is computed after averaging the five tasks within each seed. The macro margin excludes zero; QQP's interval includes zero.

Dataset	Δ score	Paired 95% CI	Two-sided p
CoLA	$+0.0361$	$[-0.0176, +0.0898]$	0.1354
MRPC	$+0.0138$	$[+0.0011, +0.0264]$	0.0392
QQP	-0.0032	$[-0.0197, +0.0134]$	0.6250
RTE	$+0.0274$	$[+0.0010, +0.0539]$	0.0450
SST-2	$+0.0094$	$[+0.00032, +0.01849]$	0.0453
Five-task macro	$+0.0167$	$[+0.0073, +0.0261]$	0.0078

A.7 Multiclass Own-Step Projection Control

Table 29 adds the independently tuned multiclass own-step control under the same endpoint convention as Appendix A.4.

Table 29: Independently tuned multiclass own-step projection control on STL10. Entries are accuracy at the 40-epoch endpoint; A–E identify the five matched evaluation runs. Current-reprojection rows reuse Table 11. Both arms use the same selection budget; training and HPO settings are in Appendix B.7.

Backbone	Projection	A	B	C	D	E	Mean
ConvNeXt-Tiny	current	0.9784	0.9688	0.9789	0.9771	0.9814	0.9769
ConvNeXt-Tiny	own-step	0.9541	0.9773	0.9774	0.9808	0.9555	0.9690
ViT-Tiny	current	0.9675	0.9703	0.9626	0.9634	0.9709	0.9669
ViT-Tiny	own-step	0.9559	0.9655	0.9628	0.9611	0.9578	0.9606

Under the main CV reporting convention, the current-minus-own-step mean accuracy differences are $+0.79$ points for ConvNeXt-Tiny (descriptive paired 95% CI $[-1.21, +2.79]$ points) and $+0.63$ points for ViT-Tiny (95% CI $[-0.09, +1.35]$ points). The intervals include zero, and both backbones reproduce the direction of the binary-task control.

A.8 Systems Resource and Timing Results

This section groups fine-tuning memory and step-time measurements, C4 pretraining end-to-end resource measurements, and the H800 single-step benchmark. Each study retains its stated precision, hardware and timing scope; Appendix B.8 specifies the measurement protocols.

A.8.1 Fine-Tuning Memory and Step Time

Memory and training-step time are measured in paired runs over the five tasks and five evaluation seeds. On RoBERTa-base, they cover the three base optimizers and their BOM compositions in bf16, and the AdamW pair is also measured with FP32 master weights and optimizer state; on DeBERTa-v3-base and Qwen3-1.7B, the AdamW pair is timed in bf16 (Appendix B.8). Table 30 reports optimizer state, peak allocated memory, and step time per method, and Figure 6 shows the paired quality, optimizer-state and step-time changes after BOM is composed with each base. In bf16, peak allocated memory falls by 232–249 MiB in the two compositions whose base keeps a full dense first moment, close to the size of the removed bf16 buffer, and by 78 MiB on GaLore, which removes both the first moment in projected coordinates and the dense first moment of the unprojected branch. With FP32 master weights, the AdamW composition removes a 475.5 MiB buffer and lowers peak allocated memory by 329 MiB.

GaLore's official projection scope leaves embeddings, LayerNorms, biases and the classification head in its dense branch. Accordingly, GaLore's state is 155.7 MiB and the composition removes 77.4 MiB. Adam-mini's block partition is preserved by the RoBERTa name adaptation; its composition removes the dense first moment. The paired step-time ratios are 0.957 for AdamW with FP32 master weights, 0.993 for AdamW in bf16, 0.980 for Adam-mini, and 0.969 for GaLore (Table 30).

For the AdamW pair, the five-task mean paired saving is 2.19 ms per step with FP32 master weights and 0.34 ms in bf16 (Table 30); BOM is faster in 21 of the 25 FP32 task–seed pairs. The larger FP32 saving is consistent with the removed first-moment buffer occupying twice as many bytes. On DeBERTa-v3-base and Qwen3-1.7B, the bf16 AdamW pairs average 73.5 versus 69.9 ms and 383.0 versus 364.0 ms per step, paired ratios of 0.952 and 0.936. The Adam-mini and GaLore pairs show reductions of 2.0% and 3.1%. Adam-mini removes a dense first moment; GaLore removes its projected first moment together with the dense first moment of its unprojected branch, 77.4 MiB in total. Their state structures, precision and update implementations require separate calibration before Eq. (19) can predict their step times quantitatively.

Table 30: RoBERTa-base memory and step time for three base optimizers and their BOM compositions over five tasks and five evaluation seeds. FP32 rows keep FP32 master weights and optimizer state with bf16 compute; the other rows use bf16 storage. State counts parameter-shaped optimizer buffers in MiB, excluding the task-space vector; peak denotes allocated GPU memory. Ratios and differences are paired within task and seed. Measurement settings and aggregation are specified in Appendix B.8.

	Optimizer state	Peak allocated (MiB)			Step time	
Method	(MiB)	Mean	Paired ratio	Paired Δ	Mean (ms)	Paired ratio
AdamW (FP32)	951.0	3194.0	–	–	49.35	–
BOM-AdamW (FP32)	475.5	2865.1	0.905	−328.9	47.15	0.957
AdamW	475.5	2276.9	–	–	39.84	–
BOM-AdamW	237.7	2045.2	0.894	−231.7	39.50	0.993
Adam-mini	238.2	1724.5	–	–	35.08	–
BOM-Adam-mini	0.42	1475.3	0.845	−249.2	34.35	0.980
GaLore	155.7	1631.7	–	–	45.58	–
BOM-GaLore	78.3	1553.1	0.948	−78.5	44.07	0.969

Adam-mini and GaLore use the authors' released implementations, with adaptation details in Appendix B.3.1; absolute timings and peak memory are compared within each pair. All three compositions reduce mean state, peak memory and paired step time; state accounting is common across rows.

A.8.2 C4 Resource Measurements

Table 31 and Figure 9 report the C4 resource comparison at 55M, 110M, 440M and 1.1B. Panel (a) pools processed tokens and process-wall seconds across the selected intervals, retaining in-window data I/O, evaluation, checkpoint writes and other overhead. Panel (b) reports the BOM peak-memory difference; panel (c) shows checkpoint sizes. The shorter 55M runs used no intermediate checkpoint. Panel (d) reports mean paired duration ratios with sample-standard-deviation bars. The 55M/110M timings cover complete runs, while all three 440M pairs use the common interval, steps 96,436–207,519 (4.55B tokens). At 1.1B, one pair uses steps 388,184–732,421 and the other two pairs use their full budgets. AdamW uses identical token endpoints within each pair; memory follows the definition in Appendix B.8. The 1.1B readings are approximately 19.252k versus 19.320k tokens/s, with a mean paired duration ratio of 0.996 and a 5.344 GiB reduction in peak allocation. Timing intervals and aggregation are defined in Appendix B.8.

Table 31: C4 resource measurements used in Figure 9. The study uses five consecutive-seed reporting runs at 55M/110M and three at 440M/1.1B. Times are arithmetic means of the selected end-to-end intervals, including data I/O, evaluation, checkpoint writes and other process-wall overhead. Peak denotes allocated GPU memory, measured as defined in Appendix B.8. Timing intervals and aggregation are defined in Appendix B.8.

			Peak allocated (GiB)			E2E time (h)		Mean paired
Model / tokens	$B \times T$	n	AdamW	BOM	Δ (%)	AdamW	BOM	time ratio
55M / 3.0B	128×256	5	27.116	26.901	-0.215 (-0.8%)	8.175	8.714	1.066
110M / 5.0B	24×256	5	7.557	7.151	-0.406 (-5.4%)	19.355	20.063	1.037
440M / 8.5B	160×256	3	87.239	83.118	-4.121 (-4.7%)	31.959	32.975	1.032
1.1B / 15.0B	80×256	3	83.503	78.159	-5.344 (-6.4%)	178.193	177.561	0.996

The final column averages the paired ratios $T_{\mathrm{BOM},i}/T_{\mathrm{AdamW},i}$; Figure 9(d) shows their sample standard deviation. The 55M/110M times cover complete runs. At 440M, all three pairs use steps 96,436–207,519, a common 4,549,959,680-token interval. At 1.1B, one pair uses steps 388,184–732,421 (7,049,973,760 tokens); the other two pairs each cover all 14,999,982,080 tokens. Each AdamW interval matches its BOM interval. The 1.1B mean durations average these unequal horizons and are not full-budget completion times; throughput pools tokens and seconds across intervals. Windowed timings exclude earlier initialization and resume loading.

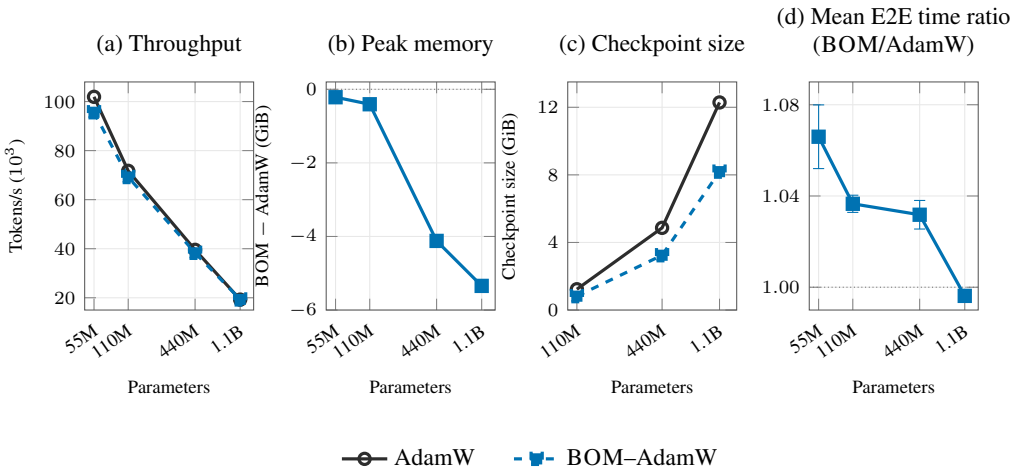

Figure 9: C4 resource comparison at four scales. (a) End-to-end throughput, $\sum_i N_i / \sum_i T_i$, includes in-window data I/O, evaluation, checkpoint writes and other process-wall overhead. (b) Peak allocated-memory difference. (c) Checkpoint sizes; 55M used no intermediate checkpoint. (d) Mean paired duration ratio with sample-standard-deviation bars over $n_t = 5, 5, 3, 3$ pairs; the dotted line marks parity. The 440M pairs use the common interval, steps 96,436–207,519. At 1.1B, one pair uses steps 388,184–732,421 and two pairs use complete runs; each pair has identical token endpoints. See Table 31 and Appendix B.8 for timing intervals and aggregation.

A.8.3 H800 Single-Step Benchmark

Table 44 gives the H800 benchmark architectures.

Implementation-specific measurements. H800 peak allocated-memory savings (AdamW minus BOM) change from -1.031 and -0.155 GiB at 250M and 500M to 2.665, 7.432 and 14.731 GiB at 1B, 2B and 4B (Figure 2). At 4B, AdamW peaks at 60.063 GiB and BOM at 45.332 GiB; the saving matches the 14.731 GiB FP32 first-moment buffer.

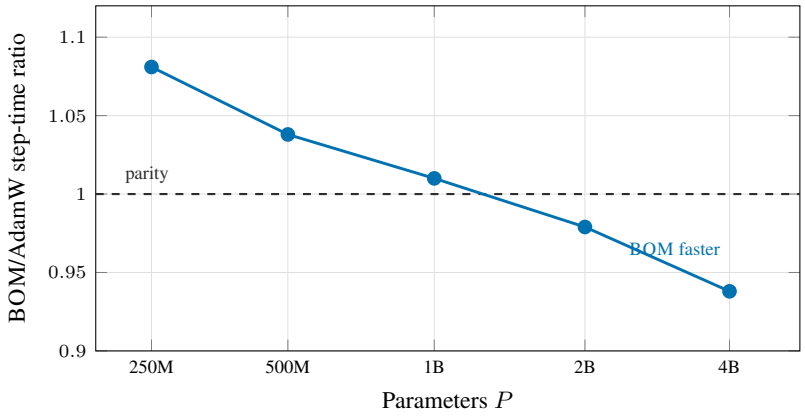

Figure 10: Paired BOM/AdamW step-time ratio against parameter count on the 250M–4B ladder (single H800, 32k vocabulary, matched FP32 protocol, median of 200 timed steps). The ratio falls monotonically, crosses parity between the 1B and 2B rungs, and reaches 0.938 at 4B. Per-device tokens per step halve from the 1B rung onward (Table 32), so r grows faster than P along the ladder. The observed crossing describes these implementations and workloads.

Table 32: H800 (Hopper, 80 GB) single-GPU ladder. Median step time over 200 timed steps; Δ is BOM $-$ AdamW. Micro-batch (mb) is selected by the stated out-of-memory halving procedure on the BOM arm; both arms share it.

Rung	P	V	mb	Tokens /step	r	AdamW (ms)	BOM (ms)	Δ (ms)	Ratio
250M	2.60×10^8	32,768	128	32,768	0.24	504.1	544.9	$+40.9$	1.081
500M	4.94×10^8	32,768	128	32,768	0.46	1041.8	1081.5	$+39.7$	1.038
1B	9.84×10^8	32,768	64	16,384	1.83	889.0	897.9	$+9.0$	1.010
2B	2.00×10^9	32,768	32	8,192	7.4	864.0	845.9	-18.1	0.979
4B	3.95×10^9	32,768	16	4,096	29.5	914.5	858.2	-56.3	0.938

The observed parity crossing lies between the 1B and 2B workloads listed in Table 32. This synthetic-token benchmark measures systems cost under the stated implementations; quality results are reported separately in Section 4.6.

B Reproducibility and Hyperparameters

B.1 Shared Selection, Reporting, and Environment

Table 33 maps the reported studies to their selection and evaluation policies. This appendix records the selected learning rates, batch size, precision, and training schedules used in the reported runs. We use separate HPO search spaces and screening horizons for fine-tuning and from-scratch pretraining because their stable step scales and affordable budgets differ. Across both domains, the default decays are $\beta_1 = 0.9$ and $\beta_2 = 0.999$: AdamW applies β_1 to its parameter-space first moment, whereas BOM applies the same value to its task-space output EMA and maintains no parameter-space first-moment tensor. Other β_1 values appear only in explicitly labeled ablations.

Unless stated otherwise, every independently tuned supervised fine-tuning method receives a learning-rate search over the same admissible envelope, $[10^{-6}, 3 \times 10^{-4}]$, with a fixed budget of 15 configurations per method–dataset pair. Learning rate is the only coordinate varied during this stage. For each task and HPO seed, the official training split is partitioned deterministically into 90% HPO training data and a 10% internal validation holdout. Every method and candidate with the same task and seed uses the identical partition, and the official validation split is not accessed during HPO. A trial denotes one hyperparameter configuration rather than one seed-level run: each configuration is trained for two epochs on the same two HPO seeds, and its selection score is the equally weighted mean of their task-appropriate internal-holdout scores at the two-epoch endpoint. Each NLP fine-tuning HPO configuration trains for two epochs but follows the first two epochs of a five-epoch learning-rate trajectory, with linear warmup over the first 10% of that five-epoch horizon and linear decay to zero only at the end of the full five-epoch trajectory. The learning-rate stage therefore comprises 15 configurations and 30 seed-level training runs; Muon's additional update-scale stage is specified below. The classifier-only historical-contribution diagnostic uses its protocol-selected two-HPO-seed configurations in the primary score. Rescue searches on MRPC, QQP and RTE are described in Appendix B.6 and excluded from that score; all other methods use their protocol-selected HPO results.

For each independently tuned method–dataset pair, the 15-configuration learning-rate budget is allocated to ten shared coarse candidates and five additional local refinement candidates around that pair's coarse incumbent. The coarse ladder is $\{1, 1.5, 2, 3, 5, 7\} \times 10^{-5}$ and $\{1, 1.5, 2, 3\} \times 10^{-4}$. The coarse incumbent is the candidate with the highest mean two-epoch endpoint score on the internal holdout over the two HPO seeds. The five refinement candidates remain within the common admissible envelope: candidates lie between adjacent coarse rates, or, when the smallest coarse rate is the incumbent, below the ladder down to 10^{-6}. This downward extension accommodates methods whose stable learning rates fall below the shared coarse ladder, including Adam-mini and its BOM composition. The final rate is selected from all fifteen evaluated configurations, retaining the coarse incumbent unless a refinement candidate achieves a strictly higher selection score. Refinement is included in the fifteen-configuration budget and cannot increase it. Method-specific coefficients remain fixed throughout the learning-rate stage. Optimizer-state memory, timing, and routing diagnostics do not enter selection. After selection, every NLP run starts afresh, trains for exactly five epochs on the complete official training split, and reports the official-validation score at the fixed epoch-5 endpoint. The RoBERTa runs also log intermediate training-probe and validation evaluations, shown descriptively in Figure 5; these curves do not replace the fixed reporting endpoint. The primary STL10 AdamW–BOM pair and the multiclass own-step control follow the same train-only HPO split, 15-configuration budget, and final-test separation, with a 40-epoch final budget. Each pair uses the same data, precision and endpoint convention, with five common consecutive evaluation seeds disjoint from the two HPO seeds. Throughout the paper, A–E denote matched evaluation runs rather than numerical seed identifiers. The paper retains the conventional GLUE term "validation score" in result tables to identify the released split, although its protocol role is final test; STL10 tables use the same term for the held-out split of their one-shot final evaluation.

Muon first completes this common learning-rate search with its update scale fixed at 0.2, then holds its selected rate fixed for an additional update-scale search over $\{0.2, 0.5, 1.0, 1.5, 2.0\}$ using the same two-epoch, two-HPO-seed internal-holdout criterion. This second stage is additional to the shared learning-rate budget; the selected scales are given in Table 39. The historical-projection control receives the common learning-rate search. The objective–momentum controls also receive the same 15-configuration, two-HPO-seed learning-rate search, with rates selected independently

Table 33: Protocol overview. LR denotes peak learning rate. HPO uses deterministic 90/10 partitions of the applicable training pool and does not access the designated validation split. After HPO selection, or after the peak rate is specified for the no-HPO 1.1B study, final runs train afresh on the complete training pool. Supervised fine-tuning reports the fixed official-validation endpoint, with logged RoBERTa intermediate evaluations shown descriptively; pretraining uses its designated validation split and reports it as validation. Each candidate uses two HPO seeds unless stated otherwise; reported quality uses five consecutive evaluation seeds disjoint from the HPO seeds, except the three-run C4-440M and C4-1.1B groups. Timing is excluded from learning-rate selection.

Study	Selection policy	Reporting and qualifications
NLP primary pairs; RoBERTa optimizer controls	Unified LR envelope $[10^{-6}, 3 \times 10^{-4}]$; 15 configurations, two-epoch screening on the internal 10% holdout	Fresh five-epoch training on full official train; fixed epoch-5 official-validation endpoint; bf16 compute and stated storage. Muon additionally searches five update scales after LR selection.
FP32-state replication	Repeat the same train-only LR policy independently for each arm and task	FP32 parameters/state, bf16 compute; same five evaluation seeds and one final evaluation on the official validation split.
NLP own-step; full temporal kernel	Unified fine-tuning LR policy	Per-task rates in Table 36; constructions in Appendix A.6.
Classifier-only past history	Unified 15-configuration, two-HPO-seed policy; primary score uses the protocol-selected configurations	Rescue searches on MRPC (one), QQP (one) and RTE (three) are all reported, and none replaces the primary configurations (Appendix B.6).
Objective–momentum controls	Unified fine-tuning LR policy; same 15-configuration, two-HPO-seed budget	Independently selected taskwise rates; five evaluation seeds at the epoch-5 endpoint (Table 43).
APM; coefficient sensitivity	Task-specific diagnostic configurations	Table 36; coefficient sweeps are post-selection diagnostics.
Numerator/second-moment diagnostics	Baseline rates plus targeted LR searches	Appendix B.6 specifies cell B's CoLA/RTE searches and cell C's CoLA stability screen; Table 43 lists endpoint rates.
STL10 primary pair; multiclass own-step	Unified 15-configuration train-only LR policy for each backbone–method pair; two HPO seeds and two-epoch screening. Own-step independently selects 3×10^{-5} on both backbones	Fresh full-train runs for 40 epochs and one final test over the five consecutive evaluation seeds (Tables 40 and 29).
Language pretraining	Matched admissible LR range $[5 \times 10^{-5}, 10^{-2}]$ and 10-evaluation budget for both arms; internal-holdout screening at 500M/1.0B/1.5B tokens for the 55M/110M/440M scales. The first two average two HPO seeds; 440M uses one. The 1.1B study performs no HPO and fixes a shared 10^{-3} rate a priori	Fresh full-budget runs (3.0B/5.0B/8.5B/15.0B tokens as specified in Table 42) on the complete prepared training pool. The designated validation pool is first accessed after HPO selection, or after the peak rate is specified for 1.1B, for trajectories and endpoint loss over five consecutive-seed reporting runs except C4 440M and 1.1B ($n = 3$ each).
ImageNet-1k pretraining	Matched pretraining LR range and 10-evaluation budget; epoch-15 top-1 on a seed-specific 10% holdout from the official training split using two HPO seeds	Fresh 90-epoch runs on the complete training split; the official validation split is first accessed after selection for epochwise and epoch-90 validation top-1 over five consecutive-seed reporting runs disjoint from HPO.

for each control and task. The APM diagnostic is reported at its task-specific configurations; the decay sweep holds each task's learning rate and all other coefficients fixed while varying β_1.

If coarse candidates tie in their mean selection score, the candidate with the smaller variance across the two HPO-seed endpoint scores becomes the coarse incumbent; if that variance also ties, the first evaluated candidate wins. This tie rule is applied before generating local refinement candidates.

Local refinement rule. The five-candidate local stage is specified as follows. For the smallest coarse incumbent, 10^{-5}, it uses $\{1.5, 2, 3, 5, 7\} \times 10^{-6}$. Otherwise, let a and b be the preceding and following coarse rates; for the largest incumbent, set $b = 3 \times 10^{-4}$. Construct $a(b/a)^{k/8}$ for $k = 1, \ldots, 7$, round each to two significant digits, remove duplicates and coarse rates, and retain the five candidates closest to the incumbent in absolute learning-rate distance (smaller rates break ties). The incumbent is fixed during candidate generation. All five candidates are evaluated; selection retains the coarse incumbent on a score tie, and otherwise breaks ties between refinement candidates by smaller learning rate.

The supervised fine-tuning and ImageNet endpoint runs use bf16 compute under their stated runners, and the fine-tuning memory and step-time figures (Tables 1, 2 and 30) use the same bf16 compute; the FP32 rows of Table 30 use the FP32-state configuration and learning rates described next. The FP32-state replication of Appendix A.6.10 repeats the RoBERTa-base AdamW–BOM learning-rate searches with the same 15-configuration, two-HPO-seed learning-rate budget and selection rule, and the five evaluation seeds, while keeping FP32 master weights and optimizer state with bf16 autocast compute. All language-pretraining endpoint studies use the same FP32-storage/bf16-autocast combination; the separate H800 step-time ladder of Appendix A.8 uses the FP32 storage protocol stated in Appendix B.8.

Fixed optimizer coefficients. Within supervised NLP fine-tuning, every coefficient of the primary AdamW–BOM pair other than learning rate is fixed before HPO. Table 34 gives these values; Table 39 gives the method-specific settings of additional controls. Muon's update scale is the sole additional searched coefficient. Both primary methods apply decoupled weight decay, disabled for bias and normalization parameters, and use linear learning-rate warmup over the first 10% of updates followed by linear decay to zero, matching the schedule shape used in language pretraining. Reported learning-rate values denote peak rates. Fine-tuning uses bf16 compute and the stated storage precision; the independently tuned FP32-state replication is detailed in Appendix B.6. The buffer-count ratio in Section 3.4 is independent of storage precision.

Table 34: Fixed optimizer coefficients for the primary supervised AdamW–BOM comparisons. The BOM objective uses a fixed $0.5/0.5$ current/history mixture, and its task-space EMA decay is set to the same value as AdamW's first-moment decay β_1, so the two *stored* histories share the EMA weights $w_{t,\tau}$. After EMA bias correction becomes negligible, BOM's batch-mean projected residual has current/past total weights $0.55/0.45$, with the current covariance term weighted 0.5 (Eq. 5); AdamW's first moment weights current/past full gradients $0.10/0.90$.

Coefficient	AdamW	BOM
First-order decay β_1 (parameter EMA / task-space EMA)	0.9	0.9
Second-moment decay β_2	0.999	0.999
Denominator constant ϵ	10^{-8}	10^{-8}
Decoupled weight decay ω	0.01	0.01
Current/history mixture $(1-\lambda)/\lambda$	–	$0.5/0.5$

Metrics and aggregation. For NLP, *score* denotes Matthews correlation for CoLA, F1 for MRPC and QQP, and accuracy for RTE and SST-2. For STL10, score denotes accuracy. Reported multi-task scores are macro averages of the applicable dataset scores, computed only after averaging the five evaluation seeds within each dataset. Loss is cross-entropy. The training-side endpoint-equivalent diagnostic uses the fixed epoch-5 AdamW training-probe macro score as its target; the first BOM hit is linearly interpolated between adjacent half-epoch training checkpoints. The official validation split does not enter this trajectory calculation. Optimizer-state memory, batch time, and wall-clock time are reported separately and are never combined with task scores; every timing pair covers the

five tasks and five evaluation seeds, and the RoBERTa-base AdamW pair is timed both in bf16 and with FP32 master weights. Each comparison shares task, seed and GPU model (Table 30); timing does not enter HPO.

B.1.1 Environment and Data

Table 35 records the models, data, software, and hardware behind the reported runs; the protocol described above applies unchanged.

Table 35: Environment for the reported runs. Model identifiers are Hugging Face model ids; every backbone is loaded through its standard sequence-classification interface, so no custom head or pooling is introduced. Precision, batch sizes, schedules, and the seed protocol are specified in the surrounding text of this appendix.

Item	Value
NLP backbones	`roberta-base`; `microsoft/deberta-v3-base`; `Qwen/Qwen3-1.7B`
Classification head	each backbone's `AutoModelForSequenceClassification` head, `num_labels`$= 2$
Tokenization	each backbone's fast `AutoTokenizer`; the decoder-only backbone pads with its end-of-sequence (EOS) token
NLP data	GLUE (`nyu-mll/glue` via Hugging Face `datasets`); CoLA, MRPC, QQP, RTE, SST-2; HPO uses a deterministic 10% holdout from official train, and the official validation split supplies the fixed epoch-5 reporting scores; logged RoBERTa intermediate evaluations are descriptive
Software	PyTorch $\geq$2.2, Transformers $\geq$4.53, `datasets` $\geq$2.18, `torchvision` $\geq$0.17, `timm` 1.0.15 (pinned), `adam-mini` 1.1.1 and `galore-torch` 1.0 (pinned)
Control implementations	Adam-mini and GaLore use the authors' released packages, `adam-mini` 1.1.1 and `galore-torch` 1.0, using the parameter grouping and projection scope described in Appendix B.3.1; all other controls use the sources cited with them
Hardware, NLP fine-tuning endpoints	one NVIDIA RTX 4090 per run for the primary pairs and the RoBERTa-base controls; the FP32-state replication (Appendix A.6.10), the AdamW $\beta_1 = 0.45$ control (Table 18) and the fixed-batch drift probe (Table 22) on one NVIDIA RTX 4090 per run; the own-step control and its drift telemetry (Tables 20 and 21) on one NVIDIA RTX PRO 6000 per run
Hardware, vision endpoints	one NVIDIA RTX 6000 Ada per run (STL10 and ImageNet-1k)
Hardware, language pretraining	one NVIDIA RTX PRO 6000 per run (Qwen3-55M on Python code and FineWeb-Edu; C4 at 55M, 110M, 440M and 1.1B); C4-1.1B uses the Blackwell Server Edition, PyTorch 2.8.0 with CUDA 12.8 and Transformers 5.8.0
Hardware, step-time measurements	RoBERTa-base, DeBERTa-v3-base and Qwen3-1.7B pairs on NVIDIA RTX 4090 cards, one job per card (Table 30); pretraining ladder on a single H800 (Appendix A.8)

B.2 NLP Fine-Tuning Parameters

Unless stated otherwise (the FP32-state replication and the fixed-batch drift probe), all supervised NLP fine-tuning runs, for every backbone and method, use batch size 32 and bf16 parameter and optimizer-state storage and compute, with the task-space EMA kept in FP32; the maximum sequence length is 128 for RoBERTa-base and DeBERTa-v3-base and 256 for Qwen3-1.7B. The AdamW–BOM pairs use the fixed coefficients in Table 34, and the other methods the settings in Table 39. The FP32-state replication keeps the batch size, sequence length and fixed coefficients

while independently tuning learning rates (Appendix B.6). Table 36 lists the selected learning rates for the independently tuned RoBERTa-base comparisons and the rates used for the APM diagnostic; Tables 37 and 38 cover the other two backbones, Table 40 the cross-domain vision setting, and Table 39 the fixed method-specific coefficients and selected Muon update scales. Every reported fine-tuning rate lies inside the common $[10^{-6}, 3 \times 10^{-4}]$ envelope. On RoBERTa-base, DeBERTa-v3-base and Qwen3-1.7B, AdamW and BOM independently select each task's learning rate under the same 15-configuration, two-HPO-seed protocol; the FP32-state replication repeats this protocol.

Table 36: Task-specific learning rates for the RoBERTa-base optimizer comparisons and the APM diagnostic, by dataset. Values are peak learning rates. All cells use max length 128, 10% linear warmup, and subsequent linear decay to zero.

Method	CoLA	MRPC	QQP	RTE	SST-2
AdamW	3×10^{-5}	3×10^{-5}	3×10^{-5}	2.5×10^{-5}	3×10^{-5}
BOM-AdamW	2.5×10^{-5}	3×10^{-5}	3×10^{-5}	2×10^{-5}	3×10^{-5}
APM	3×10^{-5}	3×10^{-5}	3×10^{-5}	2×10^{-5}	3×10^{-5}
Lion	3×10^{-5}	3×10^{-5}	2×10^{-5}	3×10^{-5}	1×10^{-5}
Adam-mini	7×10^{-6}	1×10^{-5}	3×10^{-6}	3×10^{-6}	1×10^{-5}
BOM-Adam-mini	5×10^{-6}	5×10^{-6}	1×10^{-5}	3×10^{-6}	5×10^{-6}
GaLore	1.5×10^{-5}	1×10^{-4}	3×10^{-5}	1.5×10^{-5}	1×10^{-4}
BOM-GaLore	1.5×10^{-5}	5×10^{-5}	3×10^{-5}	1.5×10^{-5}	3×10^{-5}
Muon	3×10^{-5}	3×10^{-5}	3×10^{-5}	1.2×10^{-4}	3×10^{-5}
SCALE	1×10^{-5}	3×10^{-5}	3×10^{-5}	1×10^{-5}	3×10^{-5}
Own-step control	2×10^{-5}	3×10^{-5}	2×10^{-5}	1×10^{-5}	3×10^{-5}

Table 37: DeBERTa-v3-base reproducibility parameters by dataset, for the methods reported on this backbone (AdamW and BOM). Each arm independently selects its learning rate using the common 15-configuration, two-HPO-seed protocol; the selected rates coincide. Values are peak learning rates; all cells use 10% linear warmup followed by linear decay to zero and train for five epochs.

Dataset	AdamW LR	BOM LR	Epochs	Max length
CoLA	3×10^{-5}	3×10^{-5}	5	128
MRPC	3×10^{-5}	3×10^{-5}	5	128
QQP	3×10^{-5}	3×10^{-5}	5	128
RTE	3×10^{-5}	3×10^{-5}	5	128
SST-2	3×10^{-5}	3×10^{-5}	5	128

Table 38: Qwen3-1.7B reproducibility parameters by dataset, for the methods reported on this backbone (AdamW and BOM). Each arm independently selects its learning rate using the common 15-configuration, two-HPO-seed protocol; the selected rates coincide. Values are peak learning rates; all cells use 10% linear warmup followed by linear decay to zero and train for five epochs.

Dataset	AdamW LR	BOM LR	Epochs	Max length
CoLA	2×10^{-5}	2×10^{-5}	5	256
MRPC	2×10^{-5}	2×10^{-5}	5	256
QQP	2×10^{-5}	2×10^{-5}	5	256
RTE	1×10^{-5}	1×10^{-5}	5	256
SST-2	2×10^{-5}	2×10^{-5}	5	256

B.3 Optimizer-Control and Composition Parameters

SCALE's selected rates are frozen before the five evaluation runs; its HPO and reporting outcomes are discussed alongside the control results in Appendix A.2.

Table 39: Settings specific to individual optimizers, beyond the learning rates in Table 36. Every value is fixed before formal HPO and is not varied during it, except Muon's update scale, fixed at 0.2 during learning-rate screening and then given an additional five-point search after the common learning-rate search, with the selected learning rate held fixed (Appendix B). GaLore's projected modules and rank follow the authors' GLUE fine-tuning script, and its projection scale and refresh interval are the `galore-torch` 1.0 defaults; all are fixed for every GaLore and BOM-GaLore cell. Parameters outside the projected module list—embeddings, LayerNorms, biases and the classification head—remain unprojected. The GaLore baseline applies the package's AdamW update to these parameters. In BOM-GaLore, both branches omit the local first moment and retain their second-moment rules and decoupled weight decay; the second moments are updated from the corresponding BOM numerator. SCALE applies its AdamW branch to one-dimensional parameters at the selected matrix rate, so only that rate is searched. SCALE uses the authors' official implementation. Its backbone matrices carry no first moment; the 0.9 decay is the output-layer momentum. The backbone-matrix update normalizes its gradient before applying the learning rate.

Setting	Value
GaLore, BOM-GaLore	
Implementation	`galore-torch` 1.0 (authors' release)
Projected modules	the authors' GLUE target list: attention and feed-forward `Linear` weights
Projection rank	8
Projection scale	1.0
Projection refresh interval	200 steps
Decay coefficients (β_1, β_2), ϵ	$(0.9,\ 0.999)$, 10^{-8}
Decoupled weight decay	0.01, disabled on bias and LayerNorm parameters
Lion	
Implementation	authors' official release
Decay coefficients (β_1, β_2)	$(0.9,\ 0.999)$
Decoupled weight decay	0.01
Adam-mini, BOM-Adam-mini	
Implementation	`adam-mini` 1.1.1 (authors' release), name lists re-pointed at RoBERTa
Second-moment sharing	the package's own partition: per head, per output neuron, whole-tensor, or elementwise
Decay coefficients (β_1, β_2), ϵ	$(0.9,\ 0.999)$, 10^{-8}
Decoupled weight decay	0.01, disabled on bias and LayerNorm groups
BOM-Adam-mini, BOM-GaLore	
Current/history mixture $(1-\lambda, \lambda)$	$(0.5,\ 0.5)$, fixed
Task-space EMA decay β_1	0.9, fixed
SCALE	
Output-layer first-moment decay	0.9
One-dimensional parameters	AdamW branch, $(\beta_1, \beta_2) = (0.9,\ 0.999)$, $\epsilon = 10^{-8}$, at the selected matrix rate
Weight decay	0
APM	
Anchor	fixed scalar per tensor
Ratio clip	$[0.1, 3.0]$
Muon	
Implementation	authors' official release
Momentum	0.95
Newton–Schulz steps	5
Update scale during LR search	0.2
Subsequent scale candidates	$\{0.2,\ 0.5,\ 1.0,\ 1.5,\ 2.0\}$
Selected update scale	0.2 (2.0 on RTE)

B.3.1 Optimizer Implementations

GaLore: the authors' implementation. Both GaLore rows use the authors' released `galore-torch` 1.0 package. As with Adam-mini, the only change to the optimizer is a `use_momentum` flag, so the composition can drop the parameter-local first moment and its associated bias correction. The projector, reconstruction, and second-moment update and bias correction retain the package's rules; the baseline keeps the package's first-moment update and correction. The projection scope is the one the authors' GLUE driver uses for a BERT-family backbone: the weights of the attention and feed-forward `Linear` modules are projected, and the embeddings, LayerNorms, biases and classification head remain in the unprojected branch. In the GaLore baseline, this branch uses the package's AdamW update. In BOM-GaLore, both the projected and unprojected branches replace the local first-moment numerator with the corresponding BOM numerator and allocate no local first-moment buffer. Each branch retains its second-moment construction and bias correction, now driven by the new numerator, together with decoupled weight decay. The projected module list and rank follow the authors' GLUE fine-tuning script and the projection scale and refresh interval are the `galore-torch` 1.0 defaults; all four are recorded in Table 39, fixed in advance and never searched.

Adam-mini: the authors' implementation. Both Adam-mini rows—the fixed-backbone control and the BOM composition of Table 2—use the released `adam-mini` 1.1.1 package. The only change to the optimizer is a `use_momentum` flag, so that the composition can drop the parameter-local first moment exactly as it does on the other base optimizers, disabling the associated first-moment bias correction while preserving the block partition and second-moment update and correction. Two checks were run before training. With the flag on, the modified optimizer reproduces the released package bit for bit—identical parameters after six steps and identical state size—so the control arm is the authors' optimizer; with it off, no first moment is allocated. One adaptation is required for RoBERTa: the shipped parameter-name lists are written for LLaMA-style module names, and on RoBERTa the substring `output` also matches `attention.output.*`, which routes 48 encoder tensors to the per-neuron branch and then indexes past the end of the one-dimensional LayerNorm weights. The lists are re-pointed at RoBERTa's names, leaving the block scheme itself unchanged; LayerNorms then fall through to the whole-tensor branch they were intended for. The routing printed at the start of every run then places 24 tensors on the per-head branch (query and key of the twelve layers), 53 on the per-neuron branch (embeddings, value, attention output, multilayer perceptron (MLP) and classifier), 25 on the whole-tensor branch (LayerNorm weights), and 99 on the plain elementwise branch (biases). Weight decay is disabled on the 124 bias and LayerNorm groups, matching the fine-tuning protocol.

B.4 Vision Fine-Tuning Parameters

In the primary STL10 comparison, the two arms select the same rate under the shared protocol (Table 40); the own-step control independently applies that same budget and selects 3×10^{-5} for each backbone (Appendix B.7).

Table 40: STL10 transfer parameters for AdamW and BOM. Both backbones fine-tune from ImageNet-pretrained weights with batch size 32, bf16 compute, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, weight decay 0.01, and $\lambda = 0.5$ for BOM. Both arms use linear warmup over the first 10% of updates followed by linear decay to zero. Independently selected learning rates coincide. Both arms report the final checkpoint under the shared five-seed convention. Own-step settings are in Appendix B.7.

Backbone	Dataset	LR	Epochs	Image size
ConvNeXt-Tiny	STL10	3×10^{-5}	40	224
ViT-Tiny	STL10	3×10^{-5}	40	224

B.5 Pretraining Parameters

Tables 42 and 12 list the configurations and aggregate endpoints of the from-scratch studies in Section 4.6. All pretraining runs use the shared default decays $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for both

arms: AdamW applies β_1 to its parameter-space first moment, and BOM applies it to its task-space output EMA without maintaining a parameter-space first-moment tensor. The BOM current/history mixture remains fixed at $0.5/0.5$ throughout training. The remaining language-modeling settings are decoupled weight decay 0.1, gradient clipping at 1.0, a linear learning-rate schedule with 10% warmup decaying to zero, and $\epsilon = 10^{-8}$ for both optimizers. The ImageNet runs use weight decay 0.05 and a linear schedule with five warmup epochs. ImageNet uses bf16 compute under its runner; all language-modeling pairs use FP32 parameters and optimizer states with bf16 autocast compute.

Pretraining HPO and reporting. Pretraining uses the same train-only HPO separation as fine-tuning. For each HPO seed, the applicable training pool is partitioned deterministically into 90% HPO training data and a 10% internal validation holdout. AdamW and BOM use identical seed-specific partitions, matched search spaces and matched budgets; the designated validation source is not accessed during configuration selection, and no reporting seed enters HPO. After selection, reporting runs start afresh on the complete training pool. The designated validation source is then used for reporting trajectories and endpoints, which are reported as validation measurements. Pretraining does not reuse the fine-tuning ladder, since its stable step scales are an order of magnitude larger. For each independently tuned ImageNet-1k, Python-code, FineWeb-Edu, or C4 setting, AdamW and BOM select their learning rates separately using 10 screening evaluations per setting–method study over the admissible learning-rate range $[5 \times 10^{-5}, 10^{-2}]$, organized as seven coarse evaluations on a fixed coarse ladder, $\{1, 2, 5\} \times 10^{-4}$ and $\{1, 2, 4, 6\} \times 10^{-3}$, followed by three local evaluations of the incumbent and its two neighboring rates, with all other optimizer, batch, schedule, and model settings fixed during screening. Except for the C4-440M study described below, one screening evaluation denotes one learning-rate setting run on the same two HPO seeds and scored by their mean, so evaluation count, HPO-seed count, screening budget, and selection criterion are matched between AdamW and BOM within each setting. Language-model configurations use internal-holdout next-token loss at a scale-specific screening endpoint: the 55M settings with 3.0B-token final budgets stop at 500M tokens, C4-110M with a 5.0B-token final budget stops at 1.0B tokens, and C4-440M with an 8.5B-token final budget stops at 1.5B tokens. Each screening run retains the full horizon and 10% warmup of its corresponding final schedule. The output-EMA decay and current/history mixture remain fixed throughout screening and final training. ImageNet-1k configurations follow the fixed 90-epoch learning-rate schedule and stop after 15 epochs for screening; selection maximizes the mean internal-holdout top-1 accuracy at the epoch-15 endpoint over the two HPO seeds. After selection, reporting runs start afresh from random initialization on the complete prepared training pool. The standard settings use five consecutive-seed reporting runs disjoint from the two HPO seeds.

C4-440M retains the same coarse grid, admissible range, 10-evaluation allocation, internal-holdout loss objective, and fixed training settings, but uses its scale-specific 1.5B-token screening endpoint and uses one HPO seed per screening evaluation instead of averaging two. AdamW and BOM are screened separately and both select 4×10^{-3}. Its final 8.5B-token comparison then uses three consecutive-seed reporting runs rather than the standard five. Scheduled measurements and the reported endpoint use the designated validation source only after the rates are frozen.

The C4-1.1B study performs no HPO. It fixes 10^{-3} for both arms over a 15.0B-token schedule and uses three consecutive-seed reporting runs. The shared rate was fixed before the reporting runs. The comparison otherwise inherits the shared optimizer coefficients, precision, and schedule conventions.

Three-point pretraining refinement. For an interior coarse incumbent g_i in the ordered seven-point ladder, the local stage evaluates three rates: $g_i - 0.5(g_i - g_{i-1})$, g_i itself, and $g_i + 0.5(g_{i+1} - g_i)$. At the lower boundary incumbent 10^{-4}, the local triplet is $\{5\times10^{-5}, 10^{-4}, 1.5\times10^{-4}\}$; at the upper boundary incumbent 6×10^{-3}, it is $\{5 \times 10^{-3}, 6 \times 10^{-3}, 10^{-2}\}$. The incumbent is evaluated again on the same HPO seed or seeds rather than merely reusing its coarse-stage score. The coarse-stage scores determine the local triplet; final selection compares only the three new local-stage scores. Thus the seven coarse and three local evaluations use ten screening slots but cover nine distinct learning rates. The factor 0.5 in the interior formula means half of the adjacent coarse-grid gap, not an absolute learning-rate increment; the boundary triplets are specified separately above. For example, an incumbent of 4×10^{-3} gives the local triplet $\{3, 4, 5\} \times 10^{-3}$. For pretraining screens with two HPO seeds, candidates tied on their mean endpoint score are ordered by the variance of

the two seed-level endpoint scores, with smaller variance preferred; if the variance also ties, the candidate first evaluated wins. For the single-HPO-seed C4-440M screen, cross-seed variance is undefined, so an exact endpoint-score tie is resolved in favor of the first evaluated candidate.

Model and token-level update. Qwen3-55M denotes our randomly initialized, reduced-size implementation of the Qwen3 architecture, instantiated as `Qwen3ForCausalLM`. It has $54{,}540{,}288$ trainable parameters: 12 layers, hidden size 512, feed-forward size 1536, eight query heads and four key/value heads of dimension 64. The C4 scale study adds a 110M model with $110{,}121{,}216$ parameters (12 layers, hidden size 768, feed-forward size 2304, 12 query and six key/value heads of dimension 64) and a 440M model with $435{,}216{,}640$ parameters (20 layers, hidden size 1280, feed-forward size 3840, ten query and five key/value heads of dimension 128). The 1.1B model has $1{,}099{,}275{,}264$ parameters: 27 layers, hidden size 1792, feed-forward size 5376, 14 query heads and seven key/value heads of dimension 128. All configurations use tied input/output embeddings over a $32{,}768$-token vocabulary, SiLU, RMSNorm with $\epsilon = 10^{-6}$, RoPE base 10^6, no attention bias or dropout, and initialization standard deviation 0.02. Python code and FineWeb-Edu use the 55M configuration; all models are trained from random initialization.

For language modeling, each supervised next-token position plays the role of an example in Section 3.2. If $\mathcal{I}_t$ is the set of valid target positions and $N_t = |\mathcal{I}_t|$, the vocabulary-space signal is

$$\begin{aligned} s_t &= \frac{1}{N_t} \sum_{(i,j)\in\mathcal{I}_t} \left[\mathrm{softmax}(z_{t,i,j}) - \mathrm{onehot}(y_{t,i,j})\right], \\ \mathcal{L}_t^{\mathrm{mix}} &= (1-\lambda)\mathcal{L}_t^{\mathrm{CE}} + \frac{\lambda}{N_t} \sum_{(i,j)\in\mathcal{I}_t} \langle z_{t,i,j}, \hat{q}_t \rangle. \end{aligned} \tag{18}$$

Ignored targets are excluded from both averages; $\hat{q}_t$ is detached. One vocabulary-sized EMA is updated per optimizer step, and no language run uses gradient accumulation. All 55M runs (Python code, FineWeb-Edu and C4) and the 110M, 440M and 1.1B C4 runs use context length 256 and respective micro-batches of 128, 24, 160 and 80, for 91,552, 813,802, 207,519 and 732,421 optimizer steps. Evaluation uses 64 held-out batches: every 4,096 steps at 55M and every 50M processed tokens at the larger scales. Final evaluation loss is the endpoint at every language scale; for 1.1B, this is the evaluation after 14,999,982,080 processed tokens. Perplexity is computed by exponentiating the final loss of each run before aggregation. The 1.1B configuration follows the shared optimizer and precision settings above, with both peak learning rates set to 10^{-3}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and task-space EMA decay 0.9. ImageNet instead reports the epoch-90 validation top-1, evaluated every epoch on one RTX 6000 Ada per run.

C4-110M micro-batch selection. The common micro-batch was chosen by a throughput sweep over 12 candidate sizes from 16 to 128. For each size, the qualification ran two AdamW learning rates (3×10^{-3} and 4×10^{-3}) and two BOM rates (10^{-3} and 1.5×10^{-3}), and selected the size maximizing the minimum tokens/s across these four configurations. Table 41 reports three candidates from this sweep. Batch 24 achieved the highest observed minimum throughput; its margin over 32 was only about 0.16%. This was a throughput-based choice, not a maximum-memory batch limit. Both reporting arms retained batch 24. With context length 256 and no gradient accumulation, C4-110M therefore processes $6{,}144$ tokens per update, compared with $32{,}768$ for C4-55M (batch 128). This difference changes the optimization setting, including the number of updates per token budget, and may affect the training trajectory and endpoint loss.

Table 41: Selected C4-110M batch-qualification measurements. Throughput is the minimum over the four pre-run configurations described above, rounded to the nearest token/s. These measurements select a common micro-batch; they do not compare validation quality.

Micro-batch	Minimum throughput (tokens/s)
24 (selected)	64,698
32	64,598
128	59,121

Language corpora. The Python corpus is prepared from `codeparrot/codeparrot-clean`, the cleaned Python-code dataset released by CodeParrot[1]. Its prepared training and held-out evaluation pools contain 2.10B and 8,388,608 tokens, respectively; the 3.0B-token training budget counts tokens processed from the training pool. Tokenization uses Qwen3-1.7B-Base, with a corpus-frequency mapping retaining the 32,767 most frequent tokens plus an unknown token (0.981637 of token mass retained). FineWeb-Edu uses the `sample-10BT` configuration (shards 000–005); its documents are tokenized in sorted shard order with an appended EOS, the first 3.10B tokens form the training pool and the next 8,388,608 tokens the held-out evaluation pool, so at most one document spans the two pools. It uses the same tokenizer and vocabulary construction, with frequencies computed from training tokens only, retaining 0.98129 of token mass. C4 uses the English configuration of `allenai/c4` (Raffel et al., 2020), pinned to dataset revision `1588ec454efa1a09f29cd18ddd04fe05fc8653a2`. At 55M its prepared pools contain 2.10B training tokens and 8,388,608 held-out tokens, and the 3.0B budget again counts tokens processed; the 110M and 440M runs draw from an 8.50B-token training pool with the same vocabulary construction. It uses the Qwen3-1.7B-Base tokenizer with the same top-32,767-plus-unknown construction, retaining 0.976657 of token mass.

For the 1.1B configuration, the C4 training pool contains 15.0B tokens and uses the same frozen top-32,767 vocabulary map described above. Its batch size is 80, so the 732,421 updates process 14,999,982,080 tokens. Evaluation and checkpointing occur at each 50M-token boundary and at the final update.

Table 42: From-scratch pretraining parameters. Reporting uses five consecutive-seed runs except for the C4 440M and 1.1B studies, which each use three. Budgets refer to final schedules. Each ImageNet HPO evaluation screens its learning rate for 15 epochs over two HPO seeds. Language HPO uses scale-specific screening endpoints: 500M tokens for the 55M/3.0B settings, 1.0B for C4-110M/5.0B, and 1.5B for C4-440M/8.5B. The 55M and 110M screens average two HPO seeds. C4-440M retains the same coarse grid, admissible range, 10-evaluation allocation, loss criterion, and selection rule but uses one HPO seed; both separate screens select 4×10^{-3}. C4-1.1B performs no HPO and fixes 10^{-3} for both arms a priori. Both arms use $\beta_1 = 0.9$ and $\beta_2 = 0.999$: AdamW applies β_1 to its parameter-space first moment and BOM applies it to its task-space output EMA. The BOM arm has no parameter-space first-moment tensor and uses $\lambda = 0.5$.

Setting	Model	Budget	Schedule (warmup)	AdamW LR	BOM LR
ImageNet-1k	ResNet-50 (random init)	90 epochs	linear (5 epochs)	1×10^{-3}	1.5×10^{-3}
Python code	Qwen3-55M, 32k vocab	3.0B tokens	linear (10%)	5×10^{-3}	1.5×10^{-3}
FineWeb-Edu	Qwen3-55M, 32k vocab	3.0B tokens	linear (10%)	4×10^{-3}	1×10^{-3}
C4	Qwen3-55M, 32k vocab	3.0B tokens	linear (10%)	4×10^{-3}	2×10^{-3}
C4	Qwen3-110M, 32k vocab	5.0B tokens	linear (10%)	3×10^{-3}	1.5×10^{-3}
C4	Qwen3-440M, 32k vocab	8.5B tokens	linear (10%)	4×10^{-3}	4×10^{-3}
C4	Qwen3-1.1B, 32k vocab	15.0B tokens	linear (10%)	1×10^{-3}	1×10^{-3}

B.6 RoBERTa Diagnostic Training and Selection Details

The objective–momentum controls of Table 15 use independently selected learning rates under the same 15-configuration, two-HPO-seed HPO budget as the primary pair. The anchored-variant diagnostic in Appendix A.6.9 is reported at its task-specific configurations. The APM learning rates

[1] https://huggingface.co/datasets/codeparrot/codeparrot-clean

are listed in Table 36, and its anchor and ratio-clip settings in Table 39. Table 43 gives the learning rates for the two objective–momentum controls, the numerator/second-moment cross-pairings, and the fixed-batch drift probe.

Table 43: Learning rates used for the RoBERTa-base mechanism diagnostics. The objective–momentum rows report independently selected learning rates under the same 15-configuration, two-HPO-seed HPO budget as AdamW and BOM (Table 15); the cell B/C rows correspond to Table 24; the fixed-batch drift probe corresponds to Table 22. Cell B's first row uses the primary BOM rates; its second row uses the rates selected by targeted CoLA/RTE searches and retains the anchor rates on the other tasks. Cell C uses the primary AdamW rates, with a separate CoLA stability screen described in Appendix B.6. The two cell B rows reuse the same MRPC, QQP and SST-2 runs, where their rates coincide.

Diagnostic	CoLA	MRPC	QQP	RTE	SST-2
Mixed + local m	3×10^{-5}	3×10^{-5}	3×10^{-5}	2×10^{-5}	3×10^{-5}
CE + no local m	3×10^{-5}	3×10^{-5}	3×10^{-5}	2×10^{-5}	3×10^{-5}
Cell B, at BOM's rate	2.5×10^{-5}	3×10^{-5}	3×10^{-5}	2×10^{-5}	3×10^{-5}
Cell B, at 3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}
Cell C, at AdamW's rate	3×10^{-5}	3×10^{-5}	3×10^{-5}	2.5×10^{-5}	3×10^{-5}
Fixed-batch drift probe	2.5×10^{-5}	3×10^{-5}	3×10^{-5}	2×10^{-5}	3×10^{-5}

The cell B/C runs and the fixed-batch drift probe use RoBERTa-base, training and evaluation batch size 32, maximum sequence length 128, $\lambda = 0.5$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and weight decay 0.01. Their HPO runs evaluate the internal 10% holdout every half epoch. Their final reporting runs train for exactly five epochs on the complete official training split and then evaluate the official validation split once as the final test. Cell B/C use bf16 weights; the fixed-batch probe uses FP32 weights with bf16 autocast. The probe follows a BOM trajectory without parameter-local first-moment storage and runs in evaluation mode every 20 steps on two fixed training-side batches, using $s^* = (1, -1)/\sqrt{2}$ at lags 20, 100 and 300. It is separate from both the objective–momentum controls and the own-step projection control.

Post-selection coefficient diagnostics. The main configuration and HPO policy are frozen before these diagnostics, and none of their outcomes feeds back into selection. The local mixture sweep changes only λ to 0.25 or 0.75 on MRPC and SST-2, holding $\beta_1 = 0.9$. The decay sweep changes only β_1 over $\{0, 0.8, 0.9, 0.99\}$ on all five RoBERTa tasks, holding the $0.5/0.5$ mixture and BOM's selected task-specific learning rates fixed. Both use the five evaluation seeds and the one-shot epoch-5 final test. The AdamW current/history-ratio control changes only β_1 from 0.9 to 0.45 at AdamW's main-experiment settings. Results and their mechanistic interpretation are in Appendix A.6.2.

Full temporal-kernel control. This control is reported under the unified fine-tuning HPO policy of Appendix B: the common learning-rate envelope, configuration budget, and two-seed internal-holdout selection rule, followed by five evaluation seeds. The selected learning rates are 2×10^{-5} (CoLA), 3×10^{-5} (MRPC), 2×10^{-5} (QQP), 1×10^{-5} (RTE), and 3×10^{-5} (SST-2). Runs use the five evaluation seeds, batch size 32, maximum length 128, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ and weight decay 0.01. Parameters and second moments use bf16 storage; the added full-gradient history buffer is FP32. Each final run trains for five epochs on the complete official training split and is then evaluated once on the official validation split used as the final test set.

Classifier-only history and rescue searches. Runs use RoBERTa-base, five epochs, the five evaluation seeds, batch size 32, maximum length 128, $\lambda = 0.5$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, weight decay 0.01, bf16 parameter, gradient and second-moment storage, and an FP32 residual EMA.

Primary selection for this diagnostic follows the unified learning-rate envelope, 15-configuration budget, two-epoch screening endpoint, and two-HPO-seed internal-holdout mean used by the other independently tuned fine-tuning controls; it selects 2×10^{-5} on MRPC, 3×10^{-5} on QQP and 2×10^{-5} on RTE, and these five-seed groups form the primary result. To test whether the control is under-tuned, rescue searches on MRPC, QQP and RTE, triggered by reporting outcomes, use the

same candidate envelope, configuration budget, screening endpoint and internal-holdout criterion with a three-HPO-seed mean as the selection statistic. MRPC and QQP each receive one rescue search, selecting 3×10^{-5} and 2×10^{-5}; the three RTE rescue searches yield two distinct alternative rates, 1.5×10^{-5} and 1×10^{-5}. Accordingly, none enters the primary classifier-only score. Every five-seed group is reported in Table 23.

Numerator/second-moment cross-pairings. The cross-pairings are mechanism diagnostics with baseline-rate checks and targeted learning-rate searches. Cell B is first evaluated at BOM's task-specific rate. Additional searches on CoLA and RTE use two-epoch internal-holdout scores averaged over the two HPO seeds, selecting 3×10^{-5} on both tasks. The CoLA candidates are $\{1, 1.5, 2, 2.5, 3, 5, 7\} \times 10^{-5}$; RTE uses the same set without 2.5×10^{-5}. MRPC, QQP and SST-2 retain their 3×10^{-5} anchor rates, so the second cell B column combines the targeted-search CoLA/RTE endpoints with the unchanged runs on those three tasks. Cell C is reported at AdamW's task-specific rate, supplemented by a CoLA stability screen over $\{1, 2, 3, 5\} \times 10^{-6}$ and $\{1, 2, 3, 5\} \times 10^{-5}$ using the same two-epoch, two-HPO-seed criterion. These targeted searches are separate from the 15-configuration protocol for the independently tuned optimizer comparisons. Changing the source of $v_{\ell,t}$ changes the per-coordinate effective step sizes, so a shared nominal rate does not by itself hold update magnitudes fixed.

Output-only second moment. The OutRMS diagnostic uses two paired seeds on the two reported datasets at the fixed epoch-5 endpoint, under the primary fine-tuning schedule. It is a fixed-configuration diagnostic rather than an independently tuned optimizer comparison (Table 25).

FP32-state replication. Both arms use FP32 master weights and FP32 optimizer state with bf16 autocast compute, batch size 32, maximum length 128, the same fixed optimizer coefficients, and the same five-epoch budget as the primary study. AdamW and BOM are independently tuned for every task with the same 15-configuration, two-HPO-seed learning-rate budget and selection rule used by the bf16 comparison, then evaluated on the five evaluation seeds. The selected learning-rate pairs (AdamW, BOM) are $(3, 1.5) \times 10^{-5}$ on CoLA, $(2, 3) \times 10^{-5}$ on MRPC, $(3, 2) \times 10^{-5}$ on QQP, $(2, 2) \times 10^{-5}$ on RTE, and $(3, 2) \times 10^{-5}$ on SST-2.

B.7 Multiclass Own-Step Training and Selection Details

The STL10 own-step control independently uses the shared 15-configuration, two-HPO-seed, two-epoch screening policy and selects 3×10^{-5} on both backbones. It uses pretrained initialization, batch size 32, image size 224, bf16 compute, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, weight decay 0.01 and $\lambda = 0.5$. Final runs train on the complete training pool for 40 epochs and evaluate once using the five consecutive evaluation seeds disjoint from the two HPO seeds. Current-reprojection references reuse the primary STL10 endpoints.

B.8 Memory and Timing Protocols

For RoBERTa-base, optimizer state counts the parameter-shaped buffers allocated by each run, excluding the task-space vector. Memory is measured in MiB (2^{20} bytes); peak allocated memory is recorded with `max_memory_allocated` at the end of each run and includes parameters, gradients, activations and workspace. Paired ratios and differences are formed within task and seed and then averaged.

All timing pairs run on RTX 4090 cards with one job per card and cover five tasks and five evaluation seeds. The RoBERTa-base AdamW pairs train for five epochs under torch 2.5.1, in bf16 and with FP32 master weights, and the two runs of a pair execute back to back; the DeBERTa-v3-base and Qwen3-1.7B AdamW pairs are timed in bf16. In every AdamW pair, both arms use the same AdamW implementation, and the BOM arm omits the first moment and its bias correction. The Adam-mini and GaLore pairs use torch 2.8.0; in each pair, the base arm runs the update of the authors' released package and the BOM arm retains the corresponding update structure while omitting the local first moment. Every pair shares task, seed and GPU model. Each row's absolute step time is the mean per-step time over its runs, while composition comparisons use the mean paired ratio; absolute timings are compared within each pair.

For the training-probe comparison in Table 10, we compute each dataset's task-appropriate training-probe score and then average over the five tasks and five evaluation seeds at every half-epoch checkpoint. The epoch-5 AdamW macro score of 0.8832 defines the reference. Linear interpolation between adjacent checkpoints gives the epoch at which BOM first reaches this reference. This is a training-score comparison; the reported validation comparison uses the fixed epoch-5 endpoint. Separately, the step-time measurements in Table 30 time the forward pass, backward pass and optimizer update, excluding evaluation and checkpointing.

The separate H800 timing protocol behind Section 4.6 uses FP32 parameters, gradients, and optimizer state; bf16 autocast compute; context length 256; weight decay 0.1; $\beta_2 = 0.999$; gradient clipping 1.0; BOM at the $0.5/0.5$ current/history mixture with output-EMA decay $\beta_1 = 0.9$; gradient accumulation 1. Batches are synthetic tokens pre-generated on device, so no data pipeline contributes. For each rung the BOM arm runs first and determines the feasible micro-batch (halved automatically on out-of-memory), and the AdamW arm then runs at the same micro-batch on the same GPU back to back; each arm takes 50 warmup and 200 CUDA-event-timed steps, and we report the median. The 250M–4B rungs use the benchmark builder's size-specific Qwen3 configurations. Table 44 gives their layer configurations and exact parameter counts. Table 32 lists the workloads and measurements, and Figure 10 plots the resulting ratio curve.

Table 44: Architectures of the single-GPU systems benchmark in Table 32, taken from its model builder; P is the exact parameter count, including query/key normalization weights. All use `Qwen3ForCausalLM`, tied input/output embeddings and disabled KV caching; the timed context length is 256. L, d, and d_{ff} denote decoder layers, hidden width and intermediate width; H_q/H_{kv} are query/key–value head counts, and d_h is head width. Pos. is the configured maximum position count, and GC denotes activation checkpointing. Remaining fields in constructed configurations use the benchmark's `Qwen3Config` defaults.

Rung	P	L	d	d_{ff}	H_q/H_{kv}	d_h	Vocab.	Pos.	GC
250M	260,089,344	18	1024	3072	8/4	128	32,768	4096	No
500M	494,207,488	23	1280	3840	10/5	128	32,768	4096	Yes
1B	983,658,240	24	1792	5376	14/7	128	32,768	4096	Yes
2B	1,995,527,680	31	2304	6912	18/6	128	32,768	4096	Yes
4B	3,954,407,168	35	3072	9216	24/8	128	32,768	4096	Yes

The C4 resource study uses process-wall durations, including data I/O, evaluation, checkpoint writes and other overhead within the measured interval. The 55M and 110M summaries each use five complete paired reporting runs. For resumed runs at those scales, total duration is computed by dividing the total processed tokens by the logged full-budget process-wall throughput. At 440M, two BOM reporting runs initially held a redundant tensor copy that does not enter the update computation; it was dropped at the checkpoints at steps 95,215 and 92,774, and the third run never held it. The first evaluation points without the copy are steps 96,436 and 93,995, so all three pairs use the common window from step 96,436 to 207,519, with identical AdamW endpoints: 3,950,018,560 to 8,499,978,240 tokens, or 4,549,959,680 measured tokens. Duration is the difference of the two recorded process-wall times, with no timer reset inside the window. This retains in-window I/O, evaluation and checkpoint writes, but excludes pre-window initialization, resume loading and all earlier training; it is neither a full-run 440M duration nor an isolated-step timer. For each arm, throughput is measured tokens divided by mean duration. The time-ratio statistic is $\bar{r} = n_t^{-1} \sum_{i=1}^{n_t} T_{\text{BOM},i}/T_{\text{AdamW},i}$, with sample standard deviation across $n_t = 5, 5, 3$ pairs. Peak memory at 440M is also measured without the redundant copy. Because the copy does not affect the update, quality uses the complete 8.5B-token trajectories of the three reporting runs.

For the 1.1B timing comparison, one pair uses steps 388,184–732,421 (7,049,973,760 tokens), the interval after its BOM run dropped the redundant copy; the other two pairs use their full 14,999,982,080-token budgets. Token endpoints are identical within each AdamW–BOM pair. At an interval boundary, let N be cumulative processed tokens and ρ the logged cumulative process-wall throughput. Cumulative elapsed time is $T = N/\rho$, so the partial-interval duration is $N_{\text{end}}/\rho_{\text{end}} - N_{\text{start}}/\rho_{\text{start}}$; full-run durations use the final N/ρ. These process-wall measurements include training, evaluation, checkpointing and other elapsed overhead. Throughput across the unequal intervals is $\sum_i N_i / \sum_i T_i$. The mean duration describes the measured intervals, not

a full 15B-token run. The time ratio is the arithmetic mean of the three matched duration ratios, with sample-standard-deviation bars. Peak allocation and checkpoint sizes are measured without the redundant copy.

B.8.1 First-order cost accounting

Here B is the number of examples or sequences, $C = d_{\text{out}}$ the number of classes, T the language-model context length, and $V = d_{\text{out}}$ the vocabulary size. For a dense first moment on one unsharded device without gradient accumulation, the relocation changes two elementwise workloads: it removes maintenance of $m_{\ell,t}$ over P elements and adds task-space bookkeeping over N_{out} supervised output coordinates per optimizer step—$B\,C$ for classification and $B\,T\,V$ for autoregressive language modeling. Treating the shared model forward and backward work as common gives the first-order approximation

$$\Delta t_{\text{step}} \;\approx\; \alpha\, N_{\text{out}} - \beta\, P, \qquad r \;:=\; \frac{P}{N_{\text{out}}}, \tag{19}$$

where α, β are time costs per element that depend on hardware, precision, and implementation. Within this approximation, shorter steps require $r > r^* = \alpha/\beta$. Table 45 records the element counts; converting them into timing predictions requires calibration for the implementation being measured. Section 4.6 reports paired single-device timings. Permanent optimizer-state savings instead compare the removed P-element buffer with the added d_{out}-element EMA; peak memory additionally depends on temporary output-space allocations.

H800 calibration and checkpoint accounting. For Figure 2(a), we fit Eq. (19) by unweighted least squares without an intercept to the five single-GPU, $V = 32{,}768$ H800 rungs from 250M to 4B, using their exact parameter counts and $N_{\text{out}} = BTV$. With time in milliseconds, the fitted relation is

$$\widehat{\Delta t} = 43.2758\,\frac{N_{\text{out}}}{10^9} - 15.4769\,\frac{P}{10^9}. \tag{20}$$

The plotted ratio at rung i is $1 + \widehat{\Delta t}_i / t_{\text{AdamW},i}$, where the denominator is the measured AdamW step time. This curve is the in-sample cost-model fit to those five workloads. Both arms use activation checkpointing from the 500M rung onward, and tokens per step decrease from 1B onward. Figure 2(c) instead uses unfitted tensor counts: FP32 model weights plus two parameter-shaped moment buffers give $S_{\text{AdamW}} \simeq 12P$ bytes, whereas removing the first moment gives $S_{\text{BOM}} \simeq 8P$ bytes. These leading terms omit small serialization metadata and task-space buffers. The theoretical curves extend to 4B; checkpoint measurements are at 110M, 440M and 1.1B.

Table 45: The dimensionless element-count ratio $r = P/N_{\text{out}}$ of Eq. (19) across the training configurations of the main suite. These are structural element counts. Timing predictions additionally require the implementation-specific coefficients in Eq. (19).

Setting	Output coordinates	P	N_{out}	r
NLP fine-tuning, RoBERTa-base (B=32, C=2)	$B \cdot C$	1.25×10^8	64	$\sim 2 \times 10^6$
STL10 fine-tuning, ViT-Tiny / ConvNeXt-Tiny (B=32, C=10)	$B \cdot C$	5.7×10^6–2.8×10^7	3.2×10^2	$\sim 10^4$–10^5
ImageNet-1k pretraining, ResNet-50 (B=256, C=1000)	$B \cdot C$	2.56×10^7	2.6×10^5	$\sim 10^2$
LM pretraining, Qwen3-55M ($B\,T$=32,768, V=32,768)	$B \cdot T \cdot V$	5.45×10^7	1.1×10^9	~ 0.05

B.8.2 Sharding and Gradient Accumulation

The measured step-time crossover uses one unsharded device without gradient accumulation. Extending the accounting requires $P_{\text{state,local}}$, the first-moment elements maintained per device, and

$N_{\mathrm{out,local}}$, the output coordinates processed on that device across an optimizer step. At fixed local output workload, state sharding reduces $P_{\mathrm{state,local}}$; at fixed micro-batch size, accumulation increases $N_{\mathrm{out,local}}$. Both can reduce the expected saving from first-moment maintenance, while distributed communication requires separate measurement. These extensions have not been benchmarked here.

C Conditional Descent and Stationarity

This appendix gives a sufficient condition for descent of the original loss and a finite-horizon stationarity bound. The detached surrogate in Eq. (3) is a device for constructing an update, not a fixed objective whose minimization implies convergence of the supervised loss. We therefore analyze the actual update as a perturbed, diagonally scaled gradient step. The assumptions below are explicit; they are not established by the empirical comparisons.

Update and assumptions. Stack all parameter tensors into θ_t. Let F be the differentiable objective of interest, bounded below by $F_\star$, with L-Lipschitz gradient on a region containing the iterates and their connecting segments. Write $G_t = \nabla F(\theta_t)$ and let $\widetilde{u}_t$ denote the numerator actually supplied to the adaptive update, including clipping when used. Define

$$D_t = \operatorname{diag}\Big((\sqrt{\hat{v}_t} + \epsilon)^{-1}\Big), \qquad e_t = \widetilde{u}_t - G_t + D_t^{-1} W \theta_t, \tag{21}$$

where W is the diagonal matrix of decoupled weight-decay coefficients. The implemented parameter update is exactly

$$\theta_{t+1} = \theta_t - \eta_t D_t (G_t + e_t). \tag{22}$$

Assume $0 < a \le b < \infty$ with $aI \preceq D_t \preceq bI$ along the trajectory. For the stated second-moment recursion, $\|\widetilde{u}_t\|_\infty \le U_\infty$ and $\epsilon > 0$ suffice: bias correction makes each coordinate of $\hat{v}_t$ a convex combination of squared past numerators, so one can take $a = (U_\infty + \epsilon)^{-1}$ and $b = \epsilon^{-1}$. These bounds may be conservative. No independence between D_t and the current gradient is assumed.

Proposition 1 (finite-horizon bound). If $0 \le \eta_t \le a/(4Lb^2)$ and $S_T = \sum_{t=1}^T \eta_t > 0$, then

$$\frac{\sum_{t=1}^T \eta_t \|G_t\|^2}{S_T} \le \frac{4\big(F(\theta_1) - F_\star\big)}{aS_T} + \frac{3b^2}{a^2} \frac{\sum_{t=1}^T \eta_t \|e_t\|^2}{S_T}. \tag{23}$$

Thus the weighted mean squared gradient tends to zero if $S_T \to \infty$ and the weighted mean squared perturbation tends to zero. This is a stationarity guarantee, not convergence to a global minimum or convergence of the parameter sequence. A persistent perturbation gives a residual bound instead.

Proof. Smoothness and the spectral bounds give

$$\begin{aligned} F(\theta_{t+1}) &\le F(\theta_t) - \eta_t \langle G_t, D_t(G_t + e_t)\rangle + \frac{L\eta_t^2}{2} \|D_t(G_t + e_t)\|^2 \\ &\le F(\theta_t) - a\eta_t \|G_t\|^2 + b\eta_t \|G_t\| \|e_t\| + Lb^2\eta_t^2 (\|G_t\|^2 + \|e_t\|^2) \\ &\le F(\theta_t) - \frac{a\eta_t}{4} \|G_t\|^2 + \frac{3b^2\eta_t}{4a} \|e_t\|^2. \end{aligned} \tag{24}$$

The last line uses $b\|G_t\|\|e_t\| \le (a/2)\|G_t\|^2 + (b^2/(2a))\|e_t\|^2$, the step-size restriction, and $a \le b$. Summing, using $F(\theta_{T+1}) \ge F_\star$, and dividing by $aS_T/4$ proves Eq. (23). In particular, a nonstationary step strictly decreases F whenever $\eta_t > 0$ and $\|e_t\| < a\|G_t\|/(\sqrt{3}b)$. The argument holds for every realized batch sequence; expectations can also be taken when the terms are integrable. It does not treat an adaptive preconditioner as independent of gradient noise. □

The BOM perturbation. Let g_t be the current batch gradient and u_t the unclipped BOM numerator. The exact batch decomposition in Eq. (1) yields

$$e_t = (g_t - G_t) + \lambda \bar{J}_t^\top (\hat{q}_t - s_t) - \lambda c_t + (\widetilde{u}_t - u_t) + D_t^{-1} W \theta_t. \tag{25}$$

This separates sampling error, temporal tracking error, the covariance contribution removed from the history branch, clipping, and decoupled decay. In particular, a small tracking error alone does not imply that u_t is a descent direction: the covariance and other terms also matter. Decoupled decay is included in the perturbation for the chosen F; it is not silently identified with the gradient of an ordinary ℓ_2-regularized objective under a varying D_t.

Lemma 2 (bias-corrected EMA tracking). For $q_0 = 0$ and $0 \le \beta_1 < 1$, define $\alpha_{t,\tau} = (1-\beta_1)\beta_1^{t-\tau}/(1-\beta_1^t)$. Then $\sum_{\tau=1}^{t} \alpha_{t,\tau} = 1$ and

$$\|\hat{q}_t - s_t\| \le \sum_{\tau=1}^{t} \alpha_{t,\tau} \|s_\tau - s_t\|. \tag{26}$$

For a fixed full batch, suppose $s(\theta)$ is L_s-Lipschitz and $\|\theta_{j+1} - \theta_j\| \le \eta_j U$ with $\eta_j \le \bar{\eta}$. Then

$$\|\hat{q}_t - s_t\| \le L_s U \bar{\eta} \, \frac{\beta_1}{1-\beta_1}. \tag{27}$$

Proof. Expand the bias-corrected EMA and apply the triangle inequality. For the second claim, telescope parameter displacements to bound $\|s_\tau - s_t\|$ by $L_s U \bar{\eta}(t-\tau)$. The mean age under the normalized truncated geometric weights is at most $\beta_1/(1-\beta_1)$. For $\beta_1 = 0$ the tracking error is zero. □

Corollary 3 (a sufficient convergence regime). Consider the fixed-full-batch case $g_t = G_t$, without clipping or weight decay, under the assumptions above. Suppose additionally $\|\tilde{J}_t\|_{\mathrm{op}} \le K$ and $\|c_t\| \le C$. Equations (25) and (27) imply

$$\frac{\sum_{t=1}^{T} \eta_t \|G_t\|^2}{S_T} \le \frac{4\big(F(\theta_1) - F_\star\big)}{a S_T} + \frac{3b^2\lambda^2}{a^2} \left(C + K L_s U \bar{\eta} \frac{\beta_1}{1-\beta_1} \right)^2. \tag{28}$$

If $C = 0$, a horizon-dependent schedule with $\bar{\eta} = O(T^{-1/2})$ and $S_T = \Omega(\sqrt{T})$ gives an $O(T^{-1/2})$ stationarity rate, provided the stated constants are uniform in T. A fixed-fraction linear warmup followed by linear decay, with peak rate proportional to $T^{-1/2}$ and satisfying the step-size restriction, meets these schedule conditions. Vanishing covariance holds, for example, when the examples have identical Jacobians; it is not assumed for the reported neural-network runs. For nonzero C, Eq. (28) retains an explicit covariance-dependent residual term.

Scope of the guarantee. The bound applies to the AdamW-style BOM update through its actual diagonal preconditioner. The full-batch corollary demonstrates a sufficient convergence regime; it does not prove that fixed-batch stochastic training with fixed $\lambda = 0.5$, clipping, nonzero decay, and the reported peak learning rates satisfies it. With changing batches, Eq. (26) also contains sample replacement effects, and Eq. (27) cannot be invoked from parameter smoothness alone. Establishing a sharper stochastic guarantee for those settings, or for the Adam-mini and GaLore compositions, remains open. The result neither predicts faster convergence than AdamW nor guarantees a particular validation loss.