Activation-Keyed Momentum: An Anisotropic Momentum Update via the Delta Rule

Euijin Hong **Guannan Qu**
Electrical and Computer Engineering, Carnegie Mellon University
{ehong, gqu}@andrew.cmu.edu

Abstract

Most modern optimizers form their momentum as an exponential moving average (EMA) of past gradients, forgetting every direction at one fixed rate. However, the inputs a deep network sees during training can be highly anisotropic, with a few directions queried frequently while most are seen rarely. Preconditioning methods address this anisotropy by wrapping extra processing around this buffer and leave the momentum update itself unchanged. We propose Activation-Keyed Momentum (AK-MOMENTUM), which builds direction-awareness into the momentum update rule. The gradient of a linear layer splits into an input activation that acts as a *key* and an output-side error that acts as a *value*. Keying on that activation, AK-MOMENTUM updates the momentum buffer by the canonical delta rule, so each direction is forgotten at a rate set by how often it appears. We prove that it is a valid momentum, that it applies the input-side curvature correction without matrix inversion, and that it clears stale directions faster than EMA under both a fixed and a drifting optimum. It is a drop-in replacement for the momentum buffer of any optimizer, its coefficient transfers across widths under μP, and its extra compute stays between 22.2% and 25.0% of a gated-MLP block's linear cost with no persistent memory. In FineWeb-Edu pretraining, AdamW with AK-MOMENTUM (AK-AdamW) reaches AdamW's validation loss in up to $46.64 \pm 4.52\%$ fewer steps at 67M and $22.14 \pm 0.81\%$ at 370M over three seeds, and the gain persists at 1B on a Chinchilla-optimal budget. A Muon baseline tuned under the same protocol sits above AK-AdamW at both language-model scales, and the gain holds for SGD, ResNet-18, and ViT-Tiny on CIFAR-10. Training-time diagnostics confirm the predicted mechanism, better gradient tracking and healthier input directions. Code is available at `https://github.com/euijinh/AK-Momentum`.

1 Introduction

Training large neural networks is expensive, and much of the recent progress in large-scale AI has come from optimizers that reach better solutions in fewer steps, from Adam and AdamW [14, 16] to more recent methods such as Shampoo, SOAP, and Muon [11, 31, 13]. However, underneath all of these optimizers, the momentum remains unchanged. It keeps a running average of past gradients, integrating every new gradient and forgetting older ones at a single fixed rate [23]. This exponential moving average (EMA) update rule of momentum is the part of the optimizer that this paper rethinks.

In a linear layer, the gradient of a single data point cleanly separates into two pieces: the input the layer just received and the backpropagated error from the loss to its output. The first piece indicates where the layer received the signal within the input space, and the second says what corresponding correction to apply at the output. In the literature, it is known that the two pieces can be viewed as a key and a value [1, 12, 27, 3, 26, 5, 4], an address and the content to store at that address, respectively. EMA momentum discards this structure. By averaging each gradient as a flat matrix and fading all of

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

them at uniform rate, it collapses the key-value pairing the gradient provides for free, throwing away which directions are being queried and how often, before the optimizer ever leverages it.

This structure matters because training is largely anisotropic. The input statistics of a layer and the curvature of the loss are dominated by a few directions, followed by a long tail of directions that are nearly flat and rarely excited [25, 10, 20, 18, 9]. A single fixed forgetting rate is problematic at both ends. Along directions that appear often, the newest gradients carry the freshest information, yet the stale gradients are still more retained. Along directions that appear rarely, more averaging would suppress noise, yet the existing information fades shortly even before new gradients arrive. An ideal momentum update would set the forgetting rate by how often a direction is actually seen, which a single rate cannot deliver. Strong optimizers such as K-FAC, Shampoo, and SOAP address this anisotropy by preconditioning, but leave the momentum update itself unchanged, and the goal of this paper is to address it at the momentum level.

Contribution. We propose AK-MOMENTUM (4), which builds this direction-awareness into the momentum update itself. The method treats the incoming input activation as a key, erases the stored content along the direction the key points to, and writes the new content there before moving on. Directions queried often are refreshed often and forgotten quickly, while directions queried rarely are kept over long horizons, so the forgetting along each direction now follows how often that direction is seen. The rule behind this is the classical delta rule [33], the canonical method for storing key-value associations and the mechanism of recent associative-memory sequence models [26, 36], applied here to the momentum buffer with the input activation as the key.

The design is not only intuitive but provably well-behaved, and we establish three guarantees. First, AK-MOMENTUM remains a valid momentum, an estimator of the gradient with a closed-form fixed point (Theorem 3.4). Second, that fixed point applies the input-side curvature correction that curvature-based methods obtain by inverting a matrix, without inverting one (Remark 3.6). Third, it clears stale directions strictly faster than EMA, both when the target is fixed (Theorem 3.8) and when it drifts as the weights move (Theorem 3.10), which is what happens in practical training.

Two further properties make AK-MOMENTUM practical. First, it is a drop-in replacement for the momentum in any optimizer, leaving the second moment, weight decay, and any preconditioning exactly as they were. Therefore, it composes with Shampoo, SOAP, and Muon rather than competing with them. Second, using normalized input as the key further stabilizes the training and establishes a scaling law under maximal-update parameterization (μP) [35] whose single new coefficient transfers across widths without re-tuning and whose extra compute is small and well-characterized.

We test these advantages in language-model pretraining at 67M, 370M, and 1B parameters on FineWeb-Edu [22], against AdamW, the baseline that differs from our method only in the momentum update. AdamW with AK-MOMENTUM (AK-AdamW) reaches AdamW's validation loss in up to $46.64 \pm 4.52\%$ fewer steps at 67M and $22.14 \pm 0.81\%$ fewer at 370M and continues to a lower final loss, a gain that comfortably exceeds its compute overhead. The gap remains positive in every run and holds at 1B on a Chinchilla-optimal budget with every setting carried over from the 67M sweep, and AK-AdamW also sits below a tuned Muon baseline. The same pattern holds for SGD, ResNet-18, and ViT-Tiny on CIFAR-10, so the effect is not tied to Adam or to language, and training-time diagnostics confirm the predicted mechanism, better gradient tracking and healthier input directions.

Related Work. Three lines of work are closely related to ours.

Preconditioning-based optimizers. K-FAC, Shampoo, and SOAP precondition the gradient using curvature information, and Muon post-processes the momentum to balance its directions [17, 11, 31, 13]. Each keeps the running-average momentum and wraps extra processing around it, so AK-MOMENTUM, which changes the averaging rule itself, can be combined with all of them.

Input-side preconditioning. FOOF [6] preconditions the gradient with the inverse second moment of the input activations alone, and DoPr and Newton-Muon [37, 8] apply such a preconditioner to the momentum. That is the same input-side factor AK-MOMENTUM reaches implicitly, but AK-MOMENTUM does not need matrix inversion or any additional persistent memory beyond the momentum that the base optimizer already keeps.

Optimizers as associative memory. Viewing a layer trained by gradient descent as an associative memory has a long history. Its weight is the initialization plus a sum of outer products of the inputs it received and the errors it was trained on, so its output is an unnormalized attention read over its own

training history [1, 12], an identity that fast weight programmers also build on [27, 3, 26]. Behrouz et al. [5, 4] carry this view to the optimizer and its momentum, with the input activation as the key and the local error as the value. However, their motivation is memory capacity over long horizons rather than feature anisotropy. Their buffer-level rule keys on the gradient object rather than on its two factors, so a direction is forgotten at a rate set by the size of the error rather than how often that direction is seen. AK-MOMENTUM keys on the input activation, which is what binds the forgetting rate to input density. Wang et al. [32] act at the same level but compress the buffer by low-rank approximation. Appendix B compares the update rules side by side.

2 Background

In step t, the optimizer holds parameters θ_t and momentum M_{t-1}, computes the gradient $g_t = \nabla_\theta \mathcal{L}_t(\theta_t)$, and updates the momentum to M_t and the parameters to θ_{t+1}. EMA momentum updates

$$M_t^{\mathrm{EMA}} = \beta M_{t-1}^{\mathrm{EMA}} + (1-\beta)\, g_t, \tag{1}$$

with momentum decay $\beta \in [0, 1)$. For a linear layer $y = Wx$ with weight $W \in \mathbb{R}^{m\times n}$, input $x \in \mathbb{R}^n$, and output-side error $\delta = \partial L/\partial y \in \mathbb{R}^m$, the per-sample gradient *factorizes* as rank-1 outer product

$$g_t = \delta_t x_t^\top \in \mathbb{R}^{m\times n}. \tag{2}$$

This factored form is the structural fact that the rest of the paper exploits. It identifies x_t as a natural *key* and δ_t as a natural *value*, and makes the momentum buffer an associative memory of past (key, value) pairs.

The Delta Rule The classical delta rule [33] updates a matrix $M \in \mathbb{R}^{m\times n}$ that associates keys $k \in \mathbb{R}^n$ with values $v \in \mathbb{R}^m$, with step size η, by

$$M \leftarrow M + \eta\,(v - Mk)\, k^\top, \tag{3}$$

where $v - Mk$ is the *retrieval error*. Each step is one online stochastic gradient step on the prediction loss $\frac{1}{2}\|v - Mk\|_2^2$, so M solves the least-squares problem $\min_M \sum_\ell \frac{1}{2}\|v_\ell - Mk_\ell\|_2^2$ over the stored pairs online. A purely Hebbian accumulator $M_t = M_{t-1} + v_t k_t^\top$ accumulates interference. Fast weight programmers for sequence modeling instead use the delta rule, $M_t = M_{t-1}(I_n - \eta k_t k_t^\top) + \eta v_t k_t^\top$, which removes the old association at k_t before writing the new one and provably preserves the orthogonality structure of the stored memory [26, 36].

3 AK-MOMENTUM: Methodology

Rationale: online key–value gradient memory. AK-MOMENTUM applies the delta rule (3) to the momentum buffer, with the input activation as the key, $k_t = x_t$, and the output-side error as the value, $v_t = \delta_t$. A uniform decay β keeps an EMA-style memory along directions no key touches.

AK-MOMENTUM UPDATE RULE.

$$M_t = \beta M_{t-1} + \eta\left(\delta_t - M_{t-1}x_t\right) x_t^\top, \tag{4}$$

with EMA decay $\beta \in [0, 1)$ and delta-rule correction coefficient $\eta \in (0, 1]$. Setting $\beta = 1$ recovers the pure delta rule. Factoring out M_{t-1} gives the form used throughout the analysis,

$$M_t = M_{t-1}\left(\beta I_n - \eta\, x_t x_t^\top\right) + \eta\, \delta_t x_t^\top. \tag{5}$$

The right-multiplying operator $\beta I - \eta x_t x_t^\top$ is a data-dependent forgetting factor that contracts directions aligned with the current key and leaves orthogonal directions untouched, while $\eta\delta_t x_t^\top$ writes the new association. Rather than averaging gradients as flat matrices, AK-MOMENTUM stores the associations (x_t, δ_t) in M and updates them by the delta rule. Equation (5) is the conceptual centerpiece, since every property that follows is a property of this operator: the fixed point, the memory horizons, and the tracking dynamics.

Modular instantiations. AK-MOMENTUM replaces the first-moment buffer of any base optimizer. SGD applied with AK-MOMENTUM (AK-SGD) uses (4) as the momentum, followed by $\theta_{t+1} =$

$\theta_t - \alpha M_t$ with learning rate α. AdamW applied with AK-MOMENTUM (AK-AdamW) applies (4) to the first moment and leaves the second moment $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^{\odot 2}$, bias correction, and decoupled weight decay unchanged, with preconditioned update $U_t = \hat{M}_t \oslash (\sqrt{\hat{v}_t} + \varepsilon)$.

μP-clean width transfer. AK-MOMENTUM is μP-compatible. With the normalized key, the spectral norm of the rank-1 forgetting operator is $\|\hat{x}_t \hat{x}_t^\top\|_{\mathrm{op}} = 1$ at every width, so η is width-invariant. Under μP, (32) has a fixed-point per-coordinate scale of $\Theta(n^{-1/2})$, the scale EMA buffer attains and the base learning-rate rule expects, if and only if $\eta = \Theta(1)$ (Appendix Q). That rule is therefore inherited unchanged, with $\alpha = \Theta(1/n)$ for AK-AdamW and $\Theta(1)$ for AK-SGD on hidden layers. The result is zero-shot hyperparameter transfer across widths, which a coordinate check at widths 128 to 2048 confirms (Appendix S), and Appendix T gives the recommended recipe for the one new coefficient.

FLOPs and memory. Against EMA (1), the only extra work of the deployed rule is the correction $\eta M \hat{\Sigma}$, where $\hat{\Sigma}$ is the second moment of the normalized keys, estimated on each batch. Its gradient term is the weight gradient that the backward pass already computes (Appendix L). Per transformer block, with the cheapest contraction order for each layer, the correction costs $(6+4\gamma)/(24+18\gamma)$ of the linear forward and backward, where γ is the gated-MLP expansion ratio. This share decreases in γ, so it stays between 22.2% and 25.0% for $\gamma \geq 2$ and is 23.1% at the Llama-2 ratio we use. Counted by our training suite, the realized per-step overhead is 10.0% at 67M, 15.5% at 370M, and 18.1% at 1B. These sit below the per-block band, since attention carries no delta-rule work and the output head's correction is small next to its own cost, and they rise with width toward it. Appendix L gives the batch form of the deployed rule and Appendix M the full count and measurement setup. Persistent optimizer-state memory overhead is zero, and transient memory is bounded by $4Nn_{\max}$ bytes per block for a microbatch of N tokens and largest layer input width $n_{\max}$.

We organize the remaining parts of this section as follows. Section 3.1 shows why AK-MOMENTUM is a valid momentum. Section 3.2 proves the lemma that explains why it is better than EMA. Sections 3.3 and 3.4 derive the two consequences from that lemma: faster per-direction convergence and direction-selective tracking. Appendix A lists all notation.

3.1 AK-MOMENTUM is a Valid Gradient Estimator

Any momentum buffer is, to first order in the learning rate, an estimator of the population gradient $\bar{G}(\theta_t) = \nabla_\theta \mathcal{L}(\theta_t)$, the gradient of the expected loss over the data distribution rather than that of any single minibatch. We identify the loss whose minimization defines AK-MOMENTUM and characterize its fixed point.

Remark 3.1 (Normalized-key substitution)**.** The results from now on are stated for the unnormalized rule (5), whose key has second moment $\Sigma_x = \mathbb{E}[xx^\top]$ with eigenvalues λ_i, and use the key only through that second moment. So, they hold for the normalized-key rule (16) under $\Sigma_x \to \hat{\Sigma}$, $\bar{G} \to \hat{G}$, and $\lambda_i \to \hat{\lambda}_i$. Since $\hat{\Sigma}$ has unit trace, $\hat{\lambda}_i \in [0,1]$ measures the directional density of the input rather than its energy (Proposition C.1). The convergence point of (8) and the trace of (12) need more than substitution.

Momentum as a gradient estimator. For any realization of M_t, a first-order Taylor expansion gives $\mathcal{L}(\theta_t) - \mathcal{L}(\theta_t - \alpha M_t) = \alpha \langle M_t, \bar{G}(\theta_t) \rangle_F + O(\alpha^2)$, so a buffer that deviates from $\bar{G}_t$ pays a first-order penalty $\alpha \langle \bar{G}_t - M_t, \bar{G}_t \rangle_F$, and the quality of M_t as an estimator of $\bar{G}_t$ governs optimization efficiency (Appendix D). What distinguishes EMA from AK-MOMENTUM is therefore not whether they estimate the gradient, but *which loss the buffer minimizes online* to perform that estimate.

Two natural objectives, two distinct fixed points. The factored gradient $g_t = \delta_t x_t^\top$ admits two natural mean-square estimation objectives.

Definition 3.2 (Weight-space MSE)**.** $J_w(M) := \frac{1}{2}\mathbb{E}\big[\|M - \delta x^\top\|_F^2\big]$.

Definition 3.3 (Function-space prediction)**.** $J_f(M) := \frac{1}{2}\mathbb{E}\big[\|\delta - Mx\|_2^2\big]$.

Under stationarity, EMA (1) is online stochastic gradient descent on J_w and converges in expectation to $\bar{G}$, an unbiased estimator of the population gradient. The pure delta rule is online stochastic gradient descent on J_f and, when $\Sigma_x \succ 0$, converges to $\bar{G}\Sigma_x^{-1}$, the Wiener-optimal linear predictor of δ given x. The two fixed points agree up to a scalar for every $\bar{G}$ if and only if $\Sigma_x = cI$ (Appendices E.1 and E.2).

The Tikhonov-regularized Wiener fixed point of AK-MOMENTUM. Adding the uniform decay $\beta < 1$ regularizes this Wiener predictor.

Theorem 3.4 (Fixed point of AK-MOMENTUM)**.** *Under the quasi-static approximation (parameters change slowly relative to the momentum dynamics, so $\bar{G}$ and Σ_x are treated as constant) and $\rho(\beta I - \eta\Sigma_x) < 1$, the* AK-MOMENTUM *update* (4) *converges in expectation to*

$$M^\star = \eta\,\bar{G}\left((1-\beta)I + \eta\Sigma_x\right)^{-1}. \tag{6}$$

In the eigenbasis of Σ_x,

$$M^\star u_i = \phi_i\,\bar{G}u_i, \qquad \phi_i = \frac{\eta}{(1-\beta)+\eta\lambda_i} = \frac{1}{\mu+\lambda_i}, \qquad \mu := \frac{1-\beta}{\eta}. \tag{7}$$

Proof. Take expectations of (5). At stationarity $M^\star\big((1-\beta)I+\eta\Sigma_x\big) = \eta\bar{G}$. See Appendix E.3. □

Corollary 3.5 (Fixed point of the deployed algorithm)**.** *For the normalized-key update the same computation gives*

$$M^* = \eta\,\hat{G}\left((1-\beta)I + \eta\hat{\Sigma}\right)^{-1} = \hat{G}\,(\mu I + \hat{\Sigma})^{-1}, \qquad M^* u_i = \frac{\hat{G}u_i}{\mu+\hat{\lambda}_i}. \tag{8}$$

When the key magnitude $\|x\|$ is independent of $(\hat{x}, \delta)$, (8) is the unnormalized fixed point with a rescaled delta step, and the leftover scalar is absorbed by the learning rate. This holds exactly when $\|x\|$ is deterministic, as at every linear layer of a pre-normalization transformer that reads directly from an RMSNorm output [30]. Elsewhere the buffer weights directions of larger input norm more heavily, and none of the results below depends on the condition, since they use the key only through its second moment. For the deployed rule (32), $\hat{G}$ in (8) becomes $\bar{G}$.

Table 1: Three estimator targets and their per-direction scaling of $\bar{G}u_i$ in the eigenbasis of Σ_x. The Tikhonov ridge is $\mu = (1-\beta)/\eta$.

Estimator	$M^\star u_i / \bar{G}u_i$	Description
EMA	1	Unbiased estimator of $\bar{G}$
Wiener ($\bar{G}\Sigma_x^{-1}$)	$1/\lambda_i$	Optimal linear predictor of δ given x
AK-MOMENTUM	$\phi_i = 1/(\mu+\lambda_i)$	Tikhonov-regularized Wiener

The fixed point (6) interpolates monotonically between two limits. As $\mu \to 0$ (i.e., $\beta \to 1$), $\phi_i \to 1/\lambda_i$ recovers the unregularized Wiener predictor, and as $\mu \to \infty$ (i.e., $\eta \to 0$), $\phi_i/\phi_j \to 1$ recovers the direction-uniform scaling of EMA. AK-MOMENTUM thus sits between EMA (direction-uniform) and Wiener (input-whitening), parameterized by the single ratio μ, as shown in Table 1.

Remark 3.6 (K-FAC connection)**.** The Kronecker-factored Fisher approximation of a linear layer is $F \approx \mathbb{E}[\delta\delta^\top] \otimes \mathbb{E}[xx^\top]$, with natural-gradient update $\mathbb{E}[\delta\delta^\top]^{-1}\bar{G}\Sigma_x^{-1}$. The pure delta-rule fixed point $\bar{G}\Sigma_x^{-1}$ is exactly the input-side factor of this Kronecker product, so AK-MOMENTUM implements *implicit input-side natural-gradient preconditioning* without inverting Σ_x, at the cost of a single rank-1 update per step rather than inverting a layer-wise covariance matrix. The deployed algorithm whitens $\hat{\Sigma}$ rather than Σ_x, so this identification holds up to a scalar factor, given in Corollary 3.5.

Regarding (8) as a predictor of δ from $\hat{x}$ or as the gradient $\hat{G}$ right-multiplied by the preconditioner $(\hat{\Sigma}+\mu I)^{-1}$ describes the same object in two coordinate systems (Appendix E.5), and Appendix K discusses how the delta correction interacts with the second moment of Adam within AK-AdamW.

3.2 Anisotropic Memory Transport

Iterating (5) from $M_0 = 0$ expands M_t as a sum of past error-key outer products, each transported forward in time by a product of data-dependent contractions,

$$M_t = \eta \sum_{\ell=1}^{t} \delta_\ell x_\ell^\top \underbrace{\prod_{j=\ell+1}^{t} \left(\beta I_n - \eta x_j x_j^\top\right)}_{P_{\ell\to t}\,\in\,\mathbb{R}^{n\times n}}, \tag{9}$$

with $P_{t\to t} = I_n$. Each term $\eta\delta_\ell x_\ell^\top P_{\ell\to t}$ is the fading memory of the association stored at step ℓ. Along directions orthogonal to every subsequent key, $P_{\ell\to t}$ reduces to $\beta^{t-\ell}I$, a uniform geometric decay. Along directions aligned with subsequent keys, the additional factor $-\eta x_j x_j^\top$ applies a direction-selective attenuation, erasing stale content that conflicts with newly arriving keys.

Lemma 3.7 (Anisotropic memory transport)**.** Treat $\{x_j\}_{j>\ell}$ as drawn independently from a distribution with zero mean and covariance $\Sigma_x = \mathbb{E}[xx^\top] = \sum_i \lambda_i u_i u_i^\top$ where λ_i's are the eigenvalues and u_i's are the corresponding unit eigenvectors. Then

$$\mathbb{E}[P_{\ell\to t}]\, u_i = (\beta - \eta\lambda_i)^{t-\ell} u_i = \tilde{\beta}_i^{\,t-\ell} u_i, \qquad \tilde{\beta}_i := \beta - \eta\lambda_i. \tag{10}$$

Proof. By independence, $\mathbb{E}[P_{\ell\to t}] = \prod_{j=\ell+1}^{t} \mathbb{E}[\beta I - \eta x_j x_j^\top] = (\beta I - \eta\Sigma_x)^{t-\ell}$. Eigendecomposing Σ_x gives (10). □

Lemma 3.7 states that the expected memory of the buffer along direction u_i decays geometrically at rate $\tilde{\beta}_i$, with memory horizon

$$\tau_i = \frac{1}{1-\tilde{\beta}_i} = \frac{1}{(1-\beta) + \eta\lambda_i}. \tag{11}$$

High-density directions (large λ_i), where the local gradient changes quickly with θ, are forgotten on a short horizon. Nearly stationary low-density directions ($\lambda_i \approx 0$) keep an EMA-like horizon $\approx 1/(1-\beta)$. This is the data-driven schedule the function-space prediction loss prescribes, and this direction-dependent memory horizon is the unifying mechanism behind every result that follows.

Lemma 3.7 implies three advantages. Theorem 3.4 is its steady-state implication, with steady-state ratio of injection to forgetting $\eta/[(1-\beta) + \eta\lambda_i] = \eta\tau_i$ along u_i, and Sections 3.3 and 3.4 derive the per-direction and non-stationary advantages as direct corollaries.

3.3 Faster Per-Direction Convergence

We compare how fast AK-MOMENTUM and EMA drive the parameter error to zero along each input direction. In linear regression the comparison is exact, and the per-direction iteration matrix of AK-MOMENTUM has a strictly smaller determinant than that of EMA, which yields faster per-direction convergence in the underdamped regime. The determinant never uses the squared loss, so the statement holds beyond that setting as well.

Setup Consider the population objective $\mathcal{L}(W) = \frac{1}{2}\mathbb{E}[\|Wx - y\|^2]$ with $y = W_\star x$ and $x \sim \mathcal{D}_x$ with covariance Σ_x. Letting $E_t = W_t - W_\star$, the dynamics decouple along the eigenbasis of Σ_x, where we let $e_{t,i} = E_t u_i \in \mathbb{R}^m$, $\mu_{t,i} = M_t u_i \in \mathbb{R}^m$, into per-direction 2×2 iteration matrices acting on $(e_{t,i}, \mu_{t-1,i})^\top$:

$$A_i^{\mathrm{EMA}} = \begin{pmatrix} 1-\alpha(1-\beta)\kappa_i & -\alpha\beta \\ (1-\beta)\kappa_i & \beta \end{pmatrix}, \qquad A_i^{\mathrm{Delta}} = \begin{pmatrix} 1-\alpha\eta\kappa_i & -\alpha\tilde{\beta}_i \\ \eta\kappa_i & \tilde{\beta}_i \end{pmatrix}, \tag{12}$$

where $\tilde{\beta}_i = \beta - \eta\hat{\lambda}_i$ and $\kappa_i = u_i^\top \mathbb{E}[\|x\|\,\hat{x}\hat{x}^\top] u_i$, assuming $\hat{\Sigma}$ and $\mathbb{E}[\|x\|\,\hat{x}\hat{x}^\top]$ share eigenvectors (Appendix F.1). Normalizing the key splits one parameter into two, since the buffer forgets at a rate set by the directional density $\hat{\lambda}_i$ while the parameter error still enters through the raw input, whose per-direction size is κ_i. The split is confined to the trace, so the determinant depends only on $\hat{\lambda}_i$, and we write the EMA row with the same coupling so the two rows are read side by side.

From this we derive the determinant reduction per-direction of A_i^{Delta} over A_i^{EMA}.

Theorem 3.8 (Per-direction determinant reduction)**.** *For all $\eta > 0$ and $\hat{\lambda}_i > 0$, $\det(A_i^{\mathrm{Delta}}) = \tilde{\beta}_i = \beta - \eta\hat{\lambda}_i < \beta = \det(A_i^{\mathrm{EMA}})$, and neither determinant depends on the coupling κ_i.*

Proof. Direct computation is shown in Appendix F.2. The result is a direct consequence of Lemma 3.7, since the expected per-direction memory contraction is exactly $\tilde{\beta}_i$. □

Extension beyond Linear Regression. The loss enters (12) only at the coupling $\eta\kappa_i$ of the parameter error into the buffer, where κ_i is the curvature of the squared loss along u_i.

Lemma 3.9 (The determinant does not use the loss)**.** Let A be the per-direction iteration matrix (12) of either rule with $\eta\kappa_i$ replaced by any constant $c > 0$, implicated as η times the local curvature along u_i under whatever loss is minimized. Then $\det A = \tilde{\beta}_i$ for AK-MOMENTUM and $\det A = \beta$ for EMA momentum, for every c and every step size α. Consequently the best per-direction rate attainable over step sizes falls from the square root of β to the square root of $\tilde{\beta}_i$ on every direction with $c > 0$, whatever loss produced it.

Appendix I.1 proves the lemma, and Appendices I.2 and I.3 use it to extend Theorem 3.8 to high-dimensional linear regression with more input dimensions than samples and to generalized linear models including logistic regression. We state the rate over step sizes because at one fixed step size a smaller determinant does not by itself force a smaller spectral radius. Theorem F.2 shows that it does whenever the iteration is underdamped, a band condition on $\alpha\eta\kappa_i$ (Remark F.3).

3.4 Non-stationary Tracking Dynamics

The fixed-point analysis of Section 3.1 and the linear-regression dynamics of Section 3.3 both assume stationarity. In practical training, the population gradient $\bar{G}_t = \nabla_\theta \mathcal{L}(\theta_t)$ and the input-feature covariance $\Sigma_x(\theta_t) = \mathbb{E}[x_t x_t^\top]$ drift as the weights update. We now characterize the rate at which the momentum buffer adapts to such drift, and Appendix H states which of the assumptions of Section 3 describe a training run.

Theorem 3.10 (Direction-selective tracking under distribution shift)**.** *Suppose at step t_0 the population gradient and input-feature covariance shift to new fixed values $(\bar{G}_k, \Sigma_k)$, and let M^* be the corresponding new fixed point. Define the tracking error $T_{t,i} = (M_t - M^*)u_i$ along eigendirection u_i of Σ_k. Then*

$$\textit{EMA:} \quad \mathbb{E}[T_{t,i} \mid T_{t-1,i}] = \beta T_{t-1,i}, \quad \forall i, \tag{13}$$

$$\textit{Delta:} \quad \mathbb{E}[T_{t,i} \mid T_{t-1,i}] = \tilde{\beta}_i T_{t-1,i} = (\beta - \eta\lambda_i) T_{t-1,i}. \tag{14}$$

EMA contracts uniformly at rate β. AK-MOMENTUM *contracts along u_i at rate $\tilde{\beta}_i$, strictly faster for every direction with $\lambda_i > 0$.*

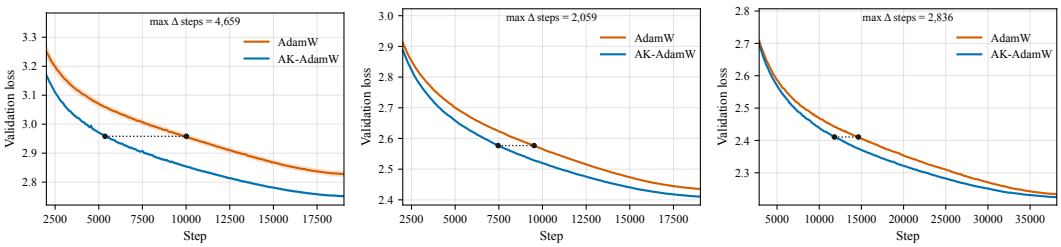
Theorem 3.10 follows immediately from Lemma 3.7, holds for any β and η in the stable range, and applies in expectation along every direction (Appendix G). Unrolling (14) gives the memory horizons (11), shorter than the uniform EMA horizon by a factor of $1 + \eta\lambda_i/(1-\beta)$, and Appendix G.2 gives tighter direction-uniform bounds and density-weighted mean contraction rates. Since the theorem constrains only the recursion that produces the buffer and places no condition on the loss landscape, it survives into non-convex training, where it sharpens into a lag bound.

Proposition 3.11 (Steady lag under drift)**.** If the target of the buffer drifts by at most Δ_i per step along u_i, then $\limsup_t \|\mathbb{E}[T_{t,i}]\| \le \tilde{\beta}_i \Delta_i/(1-\tilde{\beta}_i)$, against $\beta\Delta_i/(1-\beta)$ for EMA. Since $\tilde{\beta}_i < \beta$ on every direction with positive input density and $t \mapsto t/(1-t)$ is increasing, the bound is strictly smaller there, and the relaxation-time ratio is $1 + \eta\hat{\lambda}_i/(1-\beta)$.

Convexity never enters, so Proposition 3.11 holds for the deep non-convex losses of Section 4. Appendix I.4 connects the reduced lag back to the first-order loss decrease of Proposition D.1. It is a tracking result and gives no global convergence rate on non-convex losses.

4 Experiments

Section 3 predicts implicit input-side preconditioning at the fixed point (Theorem 3.4), direction-density-aware memory transport (Lemma 3.7), and a small constant FLOP overhead (Theorem M.3). We test whether these claims hold in deep, non-linear, non-stationary training. Section 4.1 compares AK-AdamW with AdamW in Llama-2-style language-model pretraining at 67M, 370M, and 1B parameters on FineWeb-Edu and adds a tuned Muon reference point. Section 4.2 probes the mechanism on the 67M run, and Section 4.3 checks generality with AK-SGD and AK-AdamW on CIFAR-10. Appendix N gives the full setup for every run.



(a) 67M, 3 seeds, 10BT. (b) 370M, 3 seeds, 10BT. (c) 1B, 1 seed, 20BT.

Figure 1: Validation loss against optimizer step on FineWeb-Edu. Curves at 67M and 370M are seed means with per-seed bands, and the 1B curve, on a Chinchilla-optimal budget (Appendix N), is a single seed. The annotation marks the largest step saving over matched validation-loss levels, measured on the seed-mean curve, and curves are cropped at the left to omit the early high-loss region.

4.1 Main Results: Validation Loss on FineWeb-Edu

Per-sample step efficiency. At all three scales AK-AdamW reaches AdamW's validation loss in substantially fewer steps and continues to a lower terminal value (Figure 1). The curves separate shortly after warmup and do not cross again, so AK-AdamW is below AdamW at every matched step over the entire training horizon. Table 2 reports the paired gap against AdamW across matched validation-loss levels. The sign of the paired gap is the same in every run at every scale, and at 370M the mean gap is 13.7 times its seed standard deviation. Every hyperparameter at 370M and 1B is carried over from the 67M sweep with no re-tuning, under the μP rule of Theorem Q.1. The two branches share initialization, data order, and schedule. The advantage is evident early and persists through the cosine decay, as Theorem 3.10 and Proposition 3.11 predict for a drifting target without assuming convexity or stationarity (Appendix H).

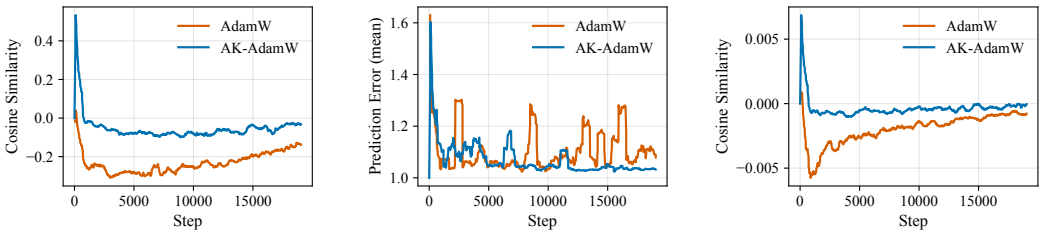
Table 2: AK-AdamW against AdamW on FineWeb-Edu across matched validation-loss levels, taken over the range plotted in Figure 1. Loss gap is in nats and step reduction is the fraction of AdamW steps saved to reach the same loss, each reported as mean and maximum over the matched levels with the seed standard deviation. The 1B run is a single seed on the Chinchilla-optimal budget Appendix N.

Scale (seeds, tokens)	Mean loss gap	Max loss gap	Mean step red.	Max step red.
67M (3 seeds, 10BT)	0.0936 ± 0.0150	0.1077 ± 0.0146	$40.85 \pm 4.99\%$	$46.64 \pm 4.52\%$
370M (3 seeds, 10BT)	0.0383 ± 0.0028	0.0501 ± 0.0016	$17.79 \pm 0.99\%$	$22.14 \pm 0.81\%$
1B (1 seed, 20BT)	0.0245	0.0361	13.47%	19.37%

Per-FLOP compute efficiency. Table 3 collects the cost side of the comparison, and Appendix M.6 gives the counted overhead and the measured step time at all three scales, with the wall-clock curves in Figure 4. Unlike the per-step comparison, each cost axis has a crossing point. At 67M both crossings come before step 2,000, where the matched levels begin. At 370M the FLOP crossing is at 4.50 EFLOPs of the 27.1 EFLOP horizon, about a sixth of the way through the run, and the wall-clock crossing comes later, near 7.0 hours, because step time sits above the FLOP count. Past its crossing on each axis AK-AdamW stays ahead at both scales.

Table 3: Cost of the deployed algorithm. Overhead is the per-step FLOP overhead counted by our training suite, and step time is measured in the training loop on a single H200 with the same microbatch size in both arms. Each savings column averages the matched validation-loss levels of the seed-mean curves past the crossing point on its own axis, with the best level in parentheses, at the two scales with a full matched-loss measurement (Appendix M).

Scale	FLOP overhead	Step time	FLOPs saved	Wall clock saved
67M	10.0%	1.1526×	35.1% (41.1%)	32.0% (38.3%)
370M	15.5%	1.1775×	5.6% (9.4%)	4.0% (7.7%)



(a) Momentum–gradient cosine similarity.

(b) Momentum–gradient prediction error.

(c) Gradient–prediction cosine similarity.

Figure 2: Mechanistic diagnostics on momentum buffer on the 67M run. AK-AdamW (blue) maintains **(a)** higher $\cos(M_t, g_t)$ and **(c)** higher $\cos(M_t\hat{x},\, \delta)$ than AdamW (orange) at every logged step, and **(b)** lower $e_t = \|M_t - g_t\|_F / \|g_t\|_F$ at most logged steps. Negative cosines come from stochastic descent itself, not the buffer rule, and occur for both optimizers (Appendix P.1). The diagnostic quantity is the gap between the two curves.

A reference point comparison among structured optimizers. Muon addresses anisotropy elsewhere, by post-processing the buffer rather than changing how it is formed. We add a Muon baseline tuned under the same Optuna protocol and run with 3 seeds at 67M and 370M. AK-AdamW sits below it at both scales, with a mean loss gap of 0.120 ± 0.008 nats at 67M and 0.109 ± 0.003 at 370M and the same sign in every run. Muon received the largest per-dimension search budget and was tuned under its own μP rule [24], and we report it as a baseline tuned inside our protocol rather than a general claim about Muon (Appendix O).

4.2 Mechanistic Diagnostics: Validating the Theoretical Picture

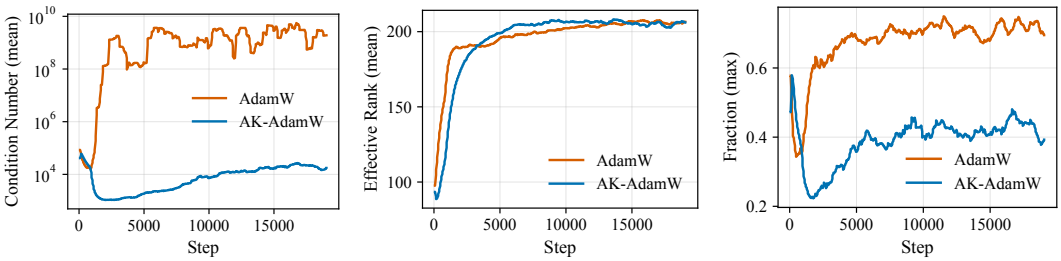
Section 3 predicts that three quantities differ between EMA momentum and AK-MOMENTUM, at three points of one mechanism. They are the buffer's alignment with the population gradient (Proposition D.1), its function-space prediction error (Definition 3.3), and the conditioning of the input-feature covariance (Theorem 3.4). We probe each at fixed evaluation steps on the 67M run with a per-layer diagnostic harness on a held-out probe batch.

Improved gradient-estimator quality. By Proposition D.1, the buffer's alignment with the population gradient governs per-step loss decrease. We measure the momentum–gradient cosine and the relative prediction error between the pre-step buffer M_t and current gradient g_t, on global parameters (Appendix P.1). Figure 2(a,b) shows that AK-AdamW keeps a *strictly higher* cosine than AdamW at every logged step and a lower relative prediction error e_t at most logged steps, with the largest values of e_t belonging to AdamW, consistent with the gradient-estimator interpretation of Section 3.1.

Function-space prediction error. By Definition 3.3, AK-MOMENTUM runs online stochastic gradient descent on the prediction loss $\frac{1}{2}\|\delta - M\hat{x}\|^2$, a loss that EMA does not optimize at all, so its buffer should predict the output-side error δ from the input $\hat{x}$ better. At each checkpoint we freeze M_t for both optimizers, predict $\widehat{\delta}_t = M_t\hat{x}_t$ per layer on a held-out probe batch, and report $\cos(\hat{\delta}, \delta)$ in Figure 2c.

Robustness against anisotropic collapse of input features. Theorem 3.4 scales the steady-state buffer along an eigendirection u_i of the input-feature covariance by $\phi_i = 1/(\mu + \lambda_i)$ rather than by 1 as in EMA, which counteracts the dominance of high-density directions. If this operates in training, layers under AK-AdamW should keep a better-conditioned input-feature covariance than under AdamW. Figure 3 reports the layerwise means of the condition number $\lambda_{\max}/\lambda_{\min}$ and of the effective rank $\exp(H(\boldsymbol{\lambda}/\sum_i \lambda_i))$, with H the Shannon entropy, and the maximum top-1 variance fraction $\lambda_{\max}/\sum_i \lambda_i$. Past 3,000 steps, AK-AdamW keeps the condition number and the top-1 fraction lower, and it keeps the effective rank higher until the two meet near step 15,000. The diagonal second moment of AdamW registers only the diagonal of Σ_x, never its eigendirections, so these statistics separate the delta correction from it (Appendix K).

Interpretation. All three diagnostics move in the direction Section 3 predicts.



(a) Condition number (mean). (b) Effective rank (layerwise mean). (c) Top-1 variance fraction (max).

Figure 3: Input-feature covariance diagnostics on the 67M run. Past 3,000 steps, layers trained with AK-AdamW (blue) have a lower condition number and a lower top-1 fraction than layers trained with AdamW (orange), and a higher effective rank until the two meet near step 15,000, consistent with the implicit input-side preconditioning predicted by Theorem 3.4.

4.3 Generality Across Optimizer Families and Architectures

To verify that the gain transfers beyond Adam's diagonal preconditioner, we compare AK-SGD with SGD-momentum on a 3-hidden-layer MLP on CIFAR-10. AK-SGD reaches SGD's validation loss earlier and converges to a strictly lower terminal value, with the gap opening right after the first epoch and persisting throughout. Validation curves and test accuracy are in Appendix R.

To verify that it also transfers beyond language modeling and beyond a single architecture, we run ResNet-18 and ViT-Tiny on CIFAR-10 for 120 epochs, AK-AdamW against AdamW with 3 seeds and an identical log-scale learning-rate sweep for each optimizer. Table 4 reports the training-loss gap at matched epochs, averaged over the phase where the curves separate, and the sign of the gap is the same in all runs. Appendix R.1 gives the curves, the averaging window, and why each architecture is read on the curve that measures optimization rather than the inductive bias of the model. We read this as a generality check rather than a second pillar of evidence, and language-model pretraining remains where the claims of this paper are made.

Table 4: AK-AdamW against AdamW on CIFAR-10 over 120 epochs with 3 seeds. Gaps are training loss at matched epochs, in nats and relative to the AdamW loss at that epoch, averaged over the phase where the curves separate. We report training loss for both architectures here so the two rows are on the same footing, and Appendix R.1 gives the validation-loss curve for ResNet-18. Curves are in Appendix R.

Architecture	Mean gap (nats)	Mean gap (%)	Max gap (nats)	Max gap (%)
ViT-Tiny	0.079 ± 0.013	25.1 ± 2.7	0.140 ± 0.017	45.5 ± 3.5
ResNet-18	0.041 ± 0.006	11.1 ± 1.6	0.081 ± 0.016	15.1 ± 1.8

5 Conclusion

AK-MOMENTUM replaces the EMA accumulator with the classical delta rule, treating the first-moment buffer as an online linear associative memory of the factored gradient $\delta_t x_t^\top$. The anisotropic memory transport mechanism (Lemma 3.7) unifies its steady-state, per-direction convergence, and non-stationary behavior. AK-AdamW reduces steps-to-target loss by up to $46.64 \pm 4.52\%$ at 67M and $22.14 \pm 0.81\%$ at 370M on FineWeb-Edu, the gain holds at 1B on a Chinchilla-optimal budget, and AK-SGD, ResNet-18, and ViT-Tiny show the same pattern on CIFAR-10.

Limitations and Future Work. Experiments cover language modeling up to 1B parameters and CIFAR-10 classifiers. Larger scales and domains such as image generation and reinforcement learning remain open. The counted per-step cost rises with width to 18.1% at 1B, where measured step time follows it, and the gap between them is larger at smaller widths. Our two modalities, language and images, are trained separately, and a run that interleaves them is the natural next step (Appendix J). Buffer-level compositions with Muon, Shampoo, and SOAP, and a controlled comparison against a gradient-keyed delta rule on the buffer [5], which isolates the choice of key, are left to future work.

Acknowledgments and Disclosure of Funding

This work is supported by NSF Grants 2339112, 2512805, Jane Street, and Pennsylvania Infrastructure Technology Alliance.

References

[1] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[3] Jimmy Ba, Geoffrey E. Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in Neural Information Processing Systems*, 29:4331–4339, 2016.

[4] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. It's all connected: A journey through test-time memorization, attentional bias, retention, and online optimization. *arXiv preprint arXiv:2504.13173*, 2025.

[5] Ali Behrouz, Meisam Razaviyayn, Peilin Zhong, and Vahab Mirrokni. Nested learning: The illusion of deep learning architectures. *Advances in Neural Information Processing Systems*, 38: 46968–47002, 2025. Extended version: arXiv:2512.24695.

[6] Frederik Benzing. Gradient descent on neurons and its link to approximate second-order optimization. In *International Conference on Machine Learning*, pages 1817–1853. PMLR, 2022.

[7] Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex SGD. In *Advances in Neural Information Processing Systems*, volume 32, pages 15236–15245, 2019.

[8] Zhehang Du and Weijie Su. The Newton-Muon optimizer. *arXiv preprint arXiv:2604.01472*, 2026.

[9] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 55–65, 2019.

[10] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR, 2019.

[11] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[12] Kazuki Irie, Róbert Csordás, and Jürgen Schmidhuber. The dual form of neural networks revisited: Connecting test time predictions to training patterns via spotlights of attention. In *International Conference on Machine Learning*, pages 9639–9659. PMLR, 2022.

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

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, ON, Canada, 2009.

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[17] James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.

[18] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*, 2017.

[19] Yurii Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Doklady Akademii Nauk SSSR*, 269(3):543–547, 1983.

[20] Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. *arXiv preprint arXiv:1901.08244*, 2019.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8024–8035, 2019.

[22] Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

[23] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[24] Shikai Qiu, Zixi Chen, Hoang Phan, Qi Lei, and Andrew Gordon Wilson. Hyperparameter transfer enables consistent gains of matrix-preconditioned optimizers across scales. In *Advances in Neural Information Processing Systems*, 2025.

[25] Levent Sagun, Utku Evci, V Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

[26] Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pages 9355–9366. PMLR, 2021.

[27] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

[28] Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International conference on machine learning*, pages 4596–4604. PMLR, 2018.

[29] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. pmlr, 2013.

[30] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

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

[32] 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.

[33] Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. In James A. Anderson and Edward Rosenfeld, editors, *Neurocomputing: Foundations of Research*, pages 123–134. MIT Press, 1988. Reprinted from *1960 IRE WESCON Convention Record*, Part 4, pp. 96–104.

[34] Yangyang Xu and Yibo Xu. Momentum-based variance-reduced proximal stochastic gradient method for composite nonconvex stochastic optimization. *Journal of Optimization Theory and Applications*, 196(1):266–297, 2023.

[35] Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097, 2021.

[36] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

[37] Thomas T. Zhang, Alok Shah, Yifei Zhang, Vincent Zhang, Nikolai Matni, and Max Simchowitz. Double preconditioning (DoPr): Optimization for test-time performance, not validation loss. *arXiv preprint arXiv:2606.06418*, 2026.

A Symbols and Notation

Table 5: Notation used in the analysis. Hatted symbols belong to the normalized-key reading (16).

Symbol	Meaning	Where it enters
$x,\ \delta$	input activation, output-side error $\partial\mathcal{L}/\partial y$	key and value of the association
$\hat{x} = x/\|x\|_2$	normalized key	deployed update
$X, \hat{X}, \Delta$	batched inputs, normalized inputs, errors	Appendix M
$\Sigma_x = \mathbb{E}[xx^\top]$	input second moment	unnormalized analysis
$\hat{\Sigma} = \mathbb{E}[\hat{x}\hat{x}^\top]$	key second moment, $\operatorname{tr}\hat{\Sigma} = 1$	deployed analysis
$\lambda_i,\ \hat{\lambda}_i,\ u_i$	eigenvalues of Σ_x, of $\hat{\Sigma}$, shared eigenvectors	per-direction statements
$\kappa_i = u_i^\top \mathbb{E}[\|x\|\,\hat{x}\hat{x}^\top]u_i$	per-direction gradient coupling	trace of (12)
$\bar{G} = \mathbb{E}[\delta x^\top]$, $\hat{G} = \mathbb{E}[\delta\hat{x}^\top]$	mean gradient in raw and normalized keys	fixed points
G	minibatch weight gradient, the batch estimate of $\bar{G}$	deployed update
$M,\ M^*$	momentum buffer and its fixed point	Theorem 3.4
$\beta,\ \eta$	EMA decay, delta-rule coefficient	(4)
$\tilde{\beta}_i = \beta - \eta\hat{\lambda}_i$	per-direction memory contraction	Lemma 3.7
$\mu = (1-\beta)/\eta$	Tikhonov ridge	Theorem 3.4
$\tau_i = 1/((1-\beta) + \eta\hat{\lambda}_i)$	per-direction memory horizon	(11)
$r_i = \eta\hat{\lambda}_i/(1-\beta)$	forgetting speed-up over EMA	Appendix I
$m,\ n,\ N,\ \gamma$	output width, input width, tokens per step, gated-MLP ratio	Appendix M

B Extended Related Work

Momentum and acceleration. First-moment EMA momentum has been a near-universal component of deep-network optimizers since heavy-ball momentum [23], which is the source of the view of the buffer as inertia. A separate line starting from Nesterov's accelerated gradient [19] builds accelerated methods on a lookahead rather than on the EMA first moment used in SGD-momentum and Adam-style optimizers, so we place it here rather than on the buffer itself. The empirical importance of momentum in neural-network training was established by Sutskever et al. [29], together with its interaction with initialization and with momentum schedules. The reading of the buffer as gradient denoising is developed directly in the smoothing and variance-reduction line [7, 34]. The isotropic, scalar-decay form of EMA has been retained in essentially every adaptive optimizer derived from Adam [14, 16]. AK-MOMENTUM departs from this lineage by replacing the EMA update itself with a delta-rule update on a function-space prediction loss, keyed on the input activation. The delta-rule constructions of Behrouz et al. [5] depart from the same lineage, on a loss keyed on the gradient instead.

Online associative memory and the delta rule. The delta rule [33] is the canonical online stochastic gradient method for linear least squares with key–value targets. In recent deep-learning literature, the delta rule has reappeared as the backbone of linear-attention fast-weight programmers [26, 36], where a memory matrix stores key–value associations across a sequence and is updated online to minimize retrieval error. A linear layer trained by gradient descent is itself such a memory, written additively, since its weight is the initialization plus a sum of outer products of its inputs and errors and its output is an unnormalized attention read over that history [1, 12]. AK-MOMENTUM imports this online-associative-memory machinery into the optimizer, so the first-moment buffer plays the role of the memory matrix, the layer's input activations and output-side errors play the roles of keys and values, and the per-step update is exactly the regularized delta rule. The view of the optimizer and its momentum as an associative memory over gradients has also appeared in Behrouz et al. [4, 5]. They factor the layer gradient into an outer product of the input activation and a local error and carry that reading to the momentum term, writing the momentum as an associative memory whose objective is a free design choice rather than a fixed one. They then instantiate that choice with a delta rule keyed on the gradient. They also give a delta rule that does key on the input activation, but it replaces the descent step on the weights rather than the momentum buffer. This paper newly devises the activation-keyed instantiation of the momentum objective and the theory that it supports. What

the choice of objective decides is the form of the resulting rule. The dot-product objective $\langle Mx, \delta \rangle$ has gradient step $\delta x^\top$, so it returns the Hebbian accumulator of Section 2, which is EMA momentum with its single decay rate. The function-space prediction loss of Definition 3.3, on the same key and value, produces the state-dependent forgetting factor of (5) instead. Table 6 places the three rules side by side.

Table 6: Where the delta rule is applied and what keys it. The first two rows are the constructions of Behrouz et al. [5], written throughout in the notation of Table 5, so β and η denote the decay and the step size belonging to the rule in that row rather than a value shared across rows. The forgetting factor right-multiplies the object in the second column. Of the two buffer-level rules, only the activation-keyed one has a forgetting factor whose expectation is $\beta I - \eta \Sigma_x$, the matrix Lemma 3.7 and everything downstream of it act on.

Rule	Applied to	Key	Value	Forgetting factor
Weight-level [5]	weights	input activation x	output-side error δ	$I - \eta\, xx^\top$
Buffer-level [5]	momentum	gradient g	preconditioner	$\beta I - g^\top g$
AK-MOMENTUM	momentum	normalized input activation $\hat{x}$	output-side error δ	$\beta I - \eta\, \hat{x}\hat{x}^\top$

Anisotropy in deep-network optimization. The anisotropy of the deep-network loss landscape has been documented extensively. Hessian-eigenvalue-density studies [25, 10, 20] report a small bulk of large eigenvalues followed by a long, near-flat tail spanning several orders of magnitude. Activation-anisotropy studies in NLP [18, 9] show that learned representations concentrate disproportionately along a handful of dominant directions. These findings have motivated optimizers that explicitly account for anisotropy. K-FAC [17] approximates the Fisher information by Kronecker-factored input/output curvature. Shampoo [11] applies Kronecker-factored preconditioning to the gradient. SOAP [31] refines the eigenbasis used by Shampoo. Muon [13] applies Newton–Schulz orthogonalization to the EMA momentum to flatten its singular-value spectrum. Adafactor [28] factorizes the second-moment statistics. A unifying property of these methods is that they retain the EMA momentum update and apply external structural processing (preconditioning before, post-conditioning after) to address anisotropy. AK-MOMENTUM addresses anisotropy at a different level of the optimizer stack, inside the momentum update itself. Therefore, AK-MOMENTUM is complementary to, rather than in competition with, these methods.

Activation-side preconditioning. FOOF [6] preconditions the gradient by the inverse second moment of the input activations alone and derives this as gradient descent on neurons. Concurrent with this work, DoPr [37] pairs a gradient-based preconditioner such as Adam or Muon with an activation-based preconditioner in the style of K-FAC, on the argument that non-isotropic input activations degrade feature learning. The object it adds is the input-side Kronecker factor, chosen over the full K-FAC preconditioner to avoid the output-side factor and the extra optimizer state it needs. That is the factor Theorem 3.4 shows AK-MOMENTUM reaches implicitly, without inversion, damping, or state beyond the momentum the base optimizer already keeps. Newton-Muon [8] reaches the same factor from a different direction. It approximates the layer loss by a quadratic surrogate in the gradient, an output-side curvature matrix, and the matrix of input activations, and minimizing that surrogate right-preconditions the gradient by the inverse input second moment before the usual Muon pipeline. It reads standard Muon as a Newton-type method that drops this right preconditioner. Both methods differ from ours in level, since each forms an explicit preconditioner and applies it to a buffer that is still an EMA, while AK-MOMENTUM changes the buffer itself, so they compose with it rather than substitute for it. DoPr reports downstream test-time performance under rollout and Newton-Muon reports iterations to a target validation loss on a GPT-2 speedrun configuration, so neither set of results is directly comparable with ours.

AK-AdamW vs. AdamW is a controlled head-to-head that isolates the first-moment update. The empirical contribution of this paper is the delta-rule update applied to the first-moment buffer, and cleanly isolating that contribution requires a baseline that differs from AK-AdamW *only* in the first-moment update. AdamW is the unique such baseline. The second moment, bias correction, and decoupled weight decay are identical between AK-AdamW and AdamW, and setting the state-dependent forgetting factor of (5) to zero returns (1) up to the constant factor $\eta/(1-\beta)$, so AdamW is the null case of the mechanism and any difference in the loss curve is attributable to the EMA-vs-delta swap. Any other delta-rule buffer, including one keyed on the gradient, differs from AK-AdamW in

the choice of key rather than in whether the mechanism is present, so it isolates a different variable. Head-to-heads against Muon, Shampoo, or SOAP would conflate the first-moment contribution with Newton–Schulz orthogonalization or with Kronecker-factored preconditioning applied at a different level of the optimizer stack, which is precisely the level at which those methods address anisotropy and at which AK-MOMENTUM is complementary, not substitutive. The natural compositions, AK-MUON, AK-SHAMPOO, and AK-SOAP, inherit AK-MOMENTUM's first-moment update while retaining the structural processing of the base optimizer, and we identify them as future work in Section 5, and an isolated AK-AdamW vs. AdamW comparison is the prerequisite for those follow-ups, not a substitute for them. The AK-SGD vs. SGD-momentum result on CIFAR-10 (Section 4.3) provides a parallel controlled comparison in the family without an adaptive denominator, verifying that the gain is attributable to the delta-rule mechanism rather than to its interaction with Adam's diagonal preconditioner.

C Stability of the Normalized-key AK-MOMENTUM

The normalized-key variant gives input-scale-independent contraction under a width-independent condition $\eta < \beta$, unlike the unnormalized-key variant. This is the main practical reason for deploying it in deep-network training.

Proposition C.1 (Bounded directional second moment)**.** Let $x \in \mathbb{R}^n$ be any random vector with $\Pr(\|x\| > 0) = 1$, and let $\hat{x} = x/\|x\|$. Then $\hat{\Sigma} := \mathbb{E}[\hat{x}\hat{x}^\top]$ satisfies

$$\operatorname{tr}(\hat{\Sigma}) = 1 \qquad \text{and} \qquad 0 \le \hat{\lambda}_i \le 1 \text{ for all } i, \tag{15}$$

with $\sum_i \hat{\lambda}_i = 1$.

Proof. $\operatorname{tr}(\hat{\Sigma}) = \mathbb{E}[\operatorname{tr}(\hat{x}\hat{x}^\top)] = \mathbb{E}[\hat{x}^\top\hat{x}] = \mathbb{E}[\|\hat{x}\|^2] = 1$. Each eigenvalue is non-negative since $\hat{\Sigma}$ is positive semi-definite, and bounded above by the trace. □

Normalized-key variant. $\hat{x}_t := x_t/\|x_t\|$, giving

$$M_t = \beta M_{t-1} + \eta(\delta_t - M_{t-1}\hat{x}_t)\hat{x}_t^\top. \tag{16}$$

Theorem C.2 (Input-scale-independent contraction of the normalized variant)**.** *The normalized-key* AK-MOMENTUM *recurrence* (16) *contracts in expectation along every input direction whenever $\eta < \beta$,* independent of *input scale, layer index, or training step. In contrast, the original unnormalized-key* AK-MOMENTUM (4) *requires the input-dependent and time-varying condition $\eta < \beta/\lambda_{\max}(\Sigma_x)$.*

Proof. By Lemma 3.7, the expected per-direction memory contraction along the eigendirection $\hat{u}_i$ of $\hat{\Sigma}$ is $\tilde{\beta}_i = \beta - \eta\hat{\lambda}_i$. By Proposition C.1, $\hat{\lambda}_i \le 1$, hence $\tilde{\beta}_i \ge \beta - \eta > 0$ under the stated condition, so the buffer contracts along every direction in expectation. For the unnormalized variant the contraction along eigenvector u_i of Σ_x is $\beta - \eta\lambda_i(\Sigma_x)$, which requires $\eta < \beta/\lambda_{\max}(\Sigma_x)$ and depends on input scale. □

The decoupling of η from input scale is the main practical reason for using the normalized variant in deep-network training, where activation scales vary by orders of magnitude across layers and through training.

Remark C.3 (Why normalization)**.** The forgetting operator is $\beta I - \eta\,\hat{x}_t\hat{x}_t^\top$, and its rank-one part has spectral norm $\|\hat{x}_t\hat{x}_t^\top\|_{\text{op}} = 1$ at every width. This decouples η from width, since stability requires only $\eta < 1 + \beta$ and $\eta = \mathcal{O}(1)$ is admissible across all model widths. Without normalization, $\|x_t x_t^\top\|_{\text{op}} \sim n$, forcing $\eta = \mathcal{O}(1/n)$. This is the foundation of μP-clean width transfer for AK-MOMENTUM.

D Why the Momentum Buffer Is Necessarily a Gradient Estimator

Proposition D.1 (First-order optimal momentum)**.** Let $\mathcal{L}$ be three times continuously differentiable near θ_t with Hessian H_t. For any choice of M_t, the update $\theta_{t+1} = \theta_t - \alpha M_t$ satisfies

$$\mathcal{L}(\theta_{t+1}) = \mathcal{L}(\theta_t) - \alpha\,\langle M_t, \bar{G}(\theta_t)\rangle_F + \frac{\alpha^2}{2}\,\operatorname{vec}(M_t)^\top H_t\,\operatorname{vec}(M_t) + O(\alpha^3), \tag{17}$$

where $\bar{G}(\theta_t) = \nabla_\theta \mathcal{L}(\theta_t)$. To first order in α, the loss decrease $\mathcal{L}(\theta_t) - \mathcal{L}(\theta_{t+1})$ is maximized over M_t of fixed Frobenius norm $\|M_t\|_F$ at $M_t \propto \bar{G}(\theta_t)$.

Proof. Equation (17) is the second-order Taylor expansion of $\mathcal{L}$ at θ_t with displacement $\Delta\theta = -\alpha M_t$. It holds pointwise for any realization of M_t, and no probabilistic averaging is required. The first-order term $-\alpha\langle M_t, \bar{G}(\theta_t)\rangle_F$ is linear in M_t, and by Cauchy–Schwarz $\langle M_t, \bar{G}_t\rangle_F$ is maximized over M_t of fixed Frobenius norm by alignment $M_t \propto \bar{G}_t$. □

Proposition D.1 is a Taylor identity, not a probabilistic assumption, since any realized buffer M_t that deviates from $\bar{G}(\theta_t)$ pays a first-order shortfall

$$\Delta_{\text{shortfall}} = \alpha\,\langle \bar{G}_t - M_t,\, \bar{G}_t\rangle_F \;\geq\; 0 \quad \text{(for fixed } \|M_t\|_F = \|\bar{G}_t\|_F\text{)}, \tag{18}$$

in the realized loss decrease relative to the optimal direction $M_t = \bar{G}_t$. The quality of M_t as an estimator of $\bar{G}_t$ therefore directly governs optimization efficiency. What differentiates between EMA and AK-MOMENTUM is the loss function on which the buffer performs online stochastic gradient descent.

E Proofs for Estimation Objectives and Fixed Point Analysis

E.1 Divergent optima

Proof. Weight-space.

$J_w(M) = \frac{1}{2}\|M\|_F^2 - \langle M, \bar{G}\rangle_F + \frac{1}{2}\mathbb{E}[\|\delta x^\top\|_F^2]$, $\nabla_M J_w = M - \bar{G} = 0 \Rightarrow M_w^* = \bar{G}$.

Function-space.

$J_f(M) = \frac{1}{2}\mathbb{E}[\|\delta\|^2] - \operatorname{tr}(M^\top \bar{G}) + \frac{1}{2}\operatorname{tr}(M\Sigma_x M^\top)$, where $\mathbb{E}[\delta^\top M x] = \operatorname{tr}(M^\top \bar{G})$ and $\mathbb{E}[x^\top M^\top M x] = \operatorname{tr}(M\Sigma_x M^\top)$, $\nabla_M J_f = -\bar{G} + M\Sigma_x = 0 \Rightarrow M_f^* = \bar{G}\,\Sigma_x^{-1}$. □

E.2 Update rules as online SGD

Proof. (a) $\nabla_M \ell_t^w = M - g_t$, so one SGD step with step α' gives $M \leftarrow M - \alpha'(M - g_t) = (1-\alpha')M + \alpha' g_t$. Setting $\alpha' = 1 - \beta$ recovers EMA. *(b)* $\nabla_M \ell_t^f = -(\delta_t - M x_t)x_t^\top$, so one SGD step with step η gives $M \leftarrow M + \eta(\delta_t - M x_t)x_t^\top$. At stationarity, $\mathbb{E}[\nabla_M \ell_t^f] = -(\bar{G} - M\Sigma_x) = 0 \Rightarrow M^* = \bar{G}\,\Sigma_x^{-1}$. □

E.3 Fixed Point Convergence of AK-MOMENTUM

Proof. Take expectations of (5) conditional on M_{t-1}: $\mathbb{E}[M_t \mid M_{t-1}] = M_{t-1}(\beta I - \eta\Sigma_x) + \eta\bar{G}$. At stationarity $M^* = M^*(\beta I - \eta\Sigma_x) + \eta\bar{G}$, hence $M^*[(1-\beta)I + \eta\Sigma_x] = \eta\bar{G}$ and (6). Projection onto u_i gives (7). □

E.4 Memory-theoretic reading

Theorem 3.4 can be read directly off Lemma 3.7: along u_i, the steady-state ratio of injection to forgetting is $\eta/[(1-\beta) + \eta\lambda_i] = \eta\tau_i$, the product of per-query injection gain η and direction-dependent memory horizon τ_i. Directions queried more often have shorter memory horizons, which reduces the steady-state retained mass along those directions, and this down-weighting *is* input-side whitening.

Remark E.1 (K-FAC connection)**.** The K-FAC approximation of the Fisher information for a linear layer is $F \approx \mathbb{E}[\delta\delta^\top] \otimes \mathbb{E}[xx^\top]$, with natural-gradient update $\mathbb{E}[\delta\delta^\top]^{-1}\,\bar{G}\,\Sigma_x^{-1}$. The pure delta-rule fixed point $\bar{G}\,\Sigma_x^{-1}$ is exactly the input-side factor of this Kronecker product. AK-MOMENTUM thus performs implicit input-side natural-gradient preconditioning without explicitly computing or inverting Σ_x, and the regularization parameter μ provides a numerically stable interpolation toward this preconditioner. As in Remark 3.6, the deployed algorithm whitens $\hat{\Sigma}$ rather than Σ_x, so this reading holds up to the scalar identified in Corollary 3.5 rather than as an identity.

E.5 Estimator and Preconditioner as One Object

AK-MOMENTUM is primarily a gradient estimator, and the preconditioning reading is what that estimator looks like when it is written in weight coordinates. Neither reading is only a statement about the fixed point. What separates AK-MOMENTUM from EMA is the space in which the estimate is formed. A layer computes $y = Wx$, so a step $W \leftarrow W - \alpha M$ changes the output on input x by $-\alpha Mx$ while the loss asks for $-\alpha\delta$. The job of the buffer is therefore to estimate the map from input to desired output change, which is the regression of δ on $\hat{x}$. That is Definition 3.3, minimized by $\hat{G}\hat{\Sigma}^{-1}$, and (4) is online SGD on it. EMA estimates the raw weight-space gradient instead, which is Definition 3.2, minimized by $\hat{G}$. So $\hat{\Sigma}^{-1}$ is not an operator applied to a gradient after the fact. It is the normal equations of the regression that defines the estimate.

The preconditioning reading follows because weight space and function space are the same linear space under two inner products, $\langle A, B\rangle_F$ and $\mathbb{E}\langle A\hat{x}, B\hat{x}\rangle = \mathrm{tr}(A\hat{\Sigma}B^\top)$. The matrix $\hat{\Sigma}^{-1}$ converts between them, so writing the function-space estimate in weight coordinates produces the right-preconditioned form of Theorem 3.4. Substituting the buffer for EMA is appropriate because the two objectives are not in conflict. Both maximize the same linear functional $\mathbb{E}\langle\delta, M\hat{x}\rangle = \langle M, \hat{G}\rangle_F$, and they differ only in which norm is held fixed while doing so, $\|M\|_F$ for Definition 3.2 and the induced output change $\mathbb{E}\|M\hat{x}\|^2$ for Definition 3.3, with the Tikhonov term μ interpolating between them. The second constraint is the one the role of the buffer calls for, and for a linear layer the identification is exact. In expectation the substitution also cannot reverse a descent direction, since $((1-\beta)I + \eta\hat{\Sigma})^{-1} \succ 0$.

F Proofs for Linear Regression Dynamics

F.1 Per-Direction Iteration Matrices

Proof. The momentum update along u_i is, in expectation,

$$\text{EMA: } \mu_{t,i} = \beta\mu_{t-1,i} + (1-\beta)\lambda_i\, e_{t,i},$$
$$\text{Delta: } \mu_{t,i} = (\beta - \eta\lambda_i)\mu_{t-1,i} + \eta\lambda_i\, e_{t,i},$$

where Delta uses the conditional expectation $\mathbb{E}[M_{t-1}x_t x_t^\top \mid M_{t-1}] = M_{t-1}\Sigma_x$ projected onto u_i. The weight update is $e_{t+1,i} = e_{t,i} - \alpha\mu_{t,i}$. Substituting the momentum recurrence into the weight recurrence and arranging the state vector as $(e_{t,i}, \mu_{t-1,i})^\top$ gives (12) in the unnormalized reading.

For the normalized-key rule (16) the same derivation gives

$$\text{Delta: } \mu_{t,i} = (\beta - \eta\hat{\lambda}_i)\mu_{t-1,i} + \eta\kappa_i\, e_{t,i},$$

because the buffer is contracted by $\hat{\Sigma}$ while the injected value $\delta_t = E_t x_t$ still carries the raw input, so $\hat{G}u_i = \mathbb{E}[\delta\hat{x}^\top]u_i = e_{t,i}\,\kappa_i$ with $\kappa_i = u_i^\top\mathbb{E}[\|x\|\,\hat{x}\hat{x}^\top]u_i$, using $x\hat{x}^\top = \|x\|\,\hat{x}\hat{x}^\top$. This is the form stated in (12), and it requires that $\hat{\Sigma}$ and $\mathbb{E}[\|x\|\,\hat{x}\hat{x}^\top]$ share eigenvectors. Only the trace moves. The determinant is $\tilde{\beta}_i$ either way, so Theorem 3.8 and Theorem F.2 are unaffected by the split. □

F.2 Per-direction Determinant Reduction

Proof. Direct computation. For EMA, $\det A_i^{\mathrm{EMA}} = \beta(1-\alpha(1-\beta)\kappa_i) - (-\alpha\beta)(1-\beta)\kappa_i = \beta - \alpha\beta(1-\beta)\kappa_i + \alpha\beta(1-\beta)\kappa_i = \beta$. For AK-MOMENTUM, $\det A_i^{\mathrm{Delta}} = \tilde{\beta}_i(1-\alpha\eta\kappa_i) - (-\alpha\tilde{\beta}_i)(\eta\kappa_i) = \tilde{\beta}_i - \alpha\eta\kappa_i\tilde{\beta}_i + \alpha\eta\kappa_i\tilde{\beta}_i = \tilde{\beta}_i$. The coupling κ_i cancels in both, which is why the determinant is set by the memory statistic alone. The inequality $\tilde{\beta}_i < \beta$ is immediate from $\eta\hat{\lambda}_i > 0$. The unnormalized reading is the same computation with κ_i replaced by λ_i and $\hat{\lambda}_i$ by λ_i. □

F.3 Convergence Rate Reduction in AK-MOMENTUM's Underdamped Regime

Lemma F.1 (Determinant lower bound on the spectral radius)**.** Let $A \in \mathbb{R}^{2\times 2}$ with $\det A > 0$. Then

$$\rho(A) \;\geq\; \sqrt{\det A}, \tag{19}$$

with equality if and only if the discriminant $\Delta(A) = \mathrm{tr}(A)^2 - 4\det A$ satisfies $\Delta(A) \leq 0$ (i.e., A has complex-conjugate eigenvalues or a repeated real eigenvalue).

Proof. Let z_1, z_2 be the eigenvalues, with $z_1 z_2 = \det A > 0$, so they are either both real with the same sign or a complex-conjugate pair. By the AM–GM inequality applied to $|z_1|, |z_2| > 0$,

$$\rho(A) = \max(|z_1|, |z_2|) \;\geq\; \tfrac{1}{2}(|z_1| + |z_2|) \;\geq\; \sqrt{|z_1||z_2|} = \sqrt{\det A}.$$

Equality in the second inequality holds iff $|z_1| = |z_2|$, which for a real matrix A holds iff either (a) $z_1 = z_2$ (repeated real eigenvalue, $\Delta = 0$), or (b) $z_1 = \bar{z}_2$ is a complex-conjugate pair ($\Delta < 0$). □

Theorem F.2 (Per-direction rate reduction in the underdamped regime)**.** *Assume the stability condition $\eta\hat{\lambda}_i < \beta$ (so $\tilde{\beta}_i = \beta - \eta\hat{\lambda}_i > 0$). Suppose that along the eigendirection $\hat{u}_i$ of $\hat{\Sigma}$, the normalized* AK-MOMENTUM *operates in the underdamped regime, i.e. $\Delta_i^{\mathrm{Delta}} < 0$. Then,* regardless of EMA's regime*,*

$$\rho_i^{\mathrm{Delta}} \;=\; \sqrt{\beta - \eta\hat{\lambda}_i} \;<\; \sqrt{\beta} \;\leq\; \rho_i^{\mathrm{EMA}}. \tag{20}$$

The reduction is anisotropic, since directions with larger directional density $\hat{\lambda}_i$ enjoy a strictly larger reduction.

Proof. Under stability, $\det A_i^{\mathrm{Delta}} = \tilde{\beta}_i > 0$ and $\det A_i^{\mathrm{EMA}} = \beta > 0$ (Theorem 3.8). Since $\Delta_i^{\mathrm{Delta}} < 0$, the eigenvalues of A_i^{Delta} form a complex-conjugate pair with squared modulus $\det A_i^{\mathrm{Delta}}$, hence $\rho_i^{\mathrm{Delta}} = \sqrt{\tilde{\beta}_i} = \sqrt{\beta - \eta\hat{\lambda}_i}$. By Lemma F.1 applied to A_i^{EMA}, $\rho_i^{\mathrm{EMA}} \geq \sqrt{\det A_i^{\mathrm{EMA}}} = \sqrt{\beta}$, with equality only if EMA is also underdamped or critically damped along $\hat{u}_i$. Since $\eta\hat{\lambda}_i > 0$, $\sqrt{\beta - \eta\hat{\lambda}_i} < \sqrt{\beta} \leq \rho_i^{\mathrm{EMA}}$. □

Remark F.3 (Explicit underdamped condition)**.** The discriminant of A_i^{Delta} is $\Delta_i^{\mathrm{Delta}} = (1 + \tilde{\beta}_i - \alpha\eta\kappa_i)^2 - 4\tilde{\beta}_i$, so the underdamped condition $\Delta_i^{\mathrm{Delta}} < 0$ is equivalent to

$$\alpha\eta\kappa_i \;\in\; \Big((1 - \sqrt{\tilde{\beta}_i})^2,\; (1 + \sqrt{\tilde{\beta}_i})^2\Big). \tag{21}$$

The window has width $4\sqrt{\tilde{\beta}_i}$, approaching 4 as $\tilde{\beta}_i \to 1$, so the band is wide whenever $\tilde{\beta}_i$ is not far from 1. The same calculation with $\tilde{\beta}_i \to \beta$ yields the EMA condition $\alpha(1-\beta)\kappa_i \in ((1-\sqrt{\beta})^2, (1+\sqrt{\beta})^2)$, and for matched (α, β) the two regimes share the same directions to leading order in $\eta\hat{\lambda}_i$, so Theorem F.2 applies on exactly the directions where EMA is also underdamped, which is the comparison of interest. For AK-AdamW the relevant α in (21) is the per-direction *effective* step size $\alpha/(\sqrt{v_i} + \varepsilon)$.

Regime of validity. Theorem F.2 restricts the comparison to the regime where AK-MOMENTUM is underdamped along $\hat{u}_i$, and EMA may be in any regime. Two remarks. *(i) Practical relevance.* High-density directions ($\hat{\lambda}_i$ near 1) typically operate in this regime under reasonable hyperparameters, and these directions dominate early- and mid-training loss-curve dynamics, making Theorem F.2 the operative regime in practice. *(ii) Honest scope.* When AK-MOMENTUM is overdamped along $\hat{u}_i$, the per-direction comparison depends jointly on trace and determinant, and a determinant decrease at fixed trace can in principle increase ρ.

G Proofs for Non-stationary Tracking Dynamics

G.1 Direction-selective Tracking under Distribution Shift

Proof. **EMA.** $\mathbb{E}[M_t \mid M_{t-1}] = \beta M_{t-1} + (1-\beta)\bar{G} = \bar{G} + \beta(M_{t-1} - \bar{G}) = M^* + \beta(M_{t-1} - M^*)$. Subtracting M^* and projecting onto u_i yields (13).

AK-MOMENTUM. From (5), $\mathbb{E}[M_t \mid M_{t-1}] = M_{t-1}(\beta I - \eta\Sigma_k) + \eta\bar{G}_k$. Using the fixed-point identity $\eta\bar{G}_k = M^*[(1-\beta)I + \eta\Sigma_k] = M^* - M^*(\beta I - \eta\Sigma_k)$:

$$\begin{aligned}\mathbb{E}[M_t - M^* \mid M_{t-1}] &= M_{t-1}(\beta I - \eta\Sigma_k) + \eta\bar{G}_k - M^* \\ &= M_{t-1}(\beta I - \eta\Sigma_k) - M^*(\beta I - \eta\Sigma_k) \\ &= (M_{t-1} - M^*)(\beta I - \eta\Sigma_k).\end{aligned}$$

Projecting onto u_i and using $\Sigma_k u_i = \lambda_i u_i$ yields $\mathbb{E}[T_{t,i}] = \tilde{\beta}_i T_{t-1,i}$. □

G.2 Uniform Tracking-rate Bounds (Normalized Variant)

For completeness we record the direction-uniform tracking bounds for the normalized variant of AK-MOMENTUM. They follow directly from Theorem 3.10 and Proposition C.1.

Corollary G.1 (Uniform tracking-rate bounds, normalized variant)**.** *For the normalized variant,* $\hat{\lambda}_i \in [0, 1]$ *implies*

$$\beta - \eta \;\le\; \tilde{\beta}_i \;\le\; \beta \quad \textit{for all } i, \tag{22}$$

so the fastest-tracking direction contracts at rate $\beta - \eta$*, and the slowest is bounded above by EMA's uniform rate* β *(a width- and layer-independent guarantee). Moreover, the directional-density-weighted mean contraction rate is*

$$\sum_i \hat{\lambda}_i \tilde{\beta}_i \;=\; \beta - \eta \,\|\hat{\Sigma}\|_F^2, \tag{23}$$

which by Cauchy–Schwarz satisfies $\sum_i \hat{\lambda}_i \tilde{\beta}_i \le \beta - \eta/n$*, with equality iff* $\hat{\Sigma}$ *is isotropic. Anisotropy of* $\hat{\Sigma}$ *strictly improves the density-weighted mean tracking rate beyond the isotropic baseline* $\beta - \eta/n$*.*

Proof. The bracket (22) is direct from $0 \le \hat{\lambda}_i \le 1$. For the density-weighted mean, $\sum_i \hat{\lambda}_i \tilde{\beta}_i = \beta \sum_i \hat{\lambda}_i - \eta \sum_i \hat{\lambda}_i^2 = \beta - \eta \,\|\hat{\Sigma}\|_F^2$, using $\mathrm{tr}(\hat{\Sigma}) = 1$. The Cauchy–Schwarz inequality $1 = (\sum_i \hat{\lambda}_i)^2 \le n \sum_i \hat{\lambda}_i^2$ gives $\|\hat{\Sigma}\|_F^2 \ge 1/n$, with equality iff all $\hat{\lambda}_i$ are equal. □

H Scope of the Theoretical Assumptions

The results of Section 3 carry different amounts of assumption. This appendix states which one describes a training run.

Lemma 3.7 needs only the key second moment and is an exact property of the deployed update, so it carries to any architecture, loss, and data. Theorem 3.10 lets both $\hat{G}$ and $\hat{\Sigma}$ move, imposes no regime restriction, and gives strictly faster expected tracking-error contraction along every direction with positive input density. The runs of Section 4 live there, so Theorem 3.10 and not Theorem 3.4 is what describes them.

Theorem 3.4 is the only result that needs quasi-staticity, and it needs half of it. The assumption covers two quantities, $\hat{G}$ and $\hat{\Sigma}$, and $\hat{\Sigma}$ alone sets the per-direction decay rates. Drift in $\hat{G}$ only makes the buffer lag a moving target, which is the tracking error Theorem 3.10 already bounds. The real condition is therefore that $\hat{\Sigma}$ moves slowly compared with the buffer memory window τ_i of (11), which is at most 100 steps at $\beta = 0.99$. The condition is local in time. It asks that $\hat{\Sigma}$ hold still over one window and reapplies as the window slides, so Theorem 3.4 describes the current phase of training rather than the whole run. This is the same locality that K-FAC and Shampoo rely on when they reuse an input-side factor for tens to hundreds of steps. It holds most comfortably after warmup and is weakest early in training and under abrupt input shifts, which are the cases Theorem 3.10 covers.

The linear-regression analysis of Section 3.3 is idealized by design. It exhibits a clean setting in which a strict per-direction improvement can be proved exactly rather than accounting for transformer training, and we know of no comparable result separating momentum schemes on non-convex losses. What carries to a transformer is the mechanism and not the closed form, and Lemma 3.9 and Proposition 3.11 are the two statements that carry it. Appendix I works out what they give beyond the squared loss, and Section 4.2 measures on the real 24-layer run the quantities that Lemma 3.7 and Theorem 3.10 govern.

I Guarantees Beyond Standard Linear Regression

Section 3.3 proves a strict per-direction improvement on standard linear regression. This appendix shows that the guarantee survives in two settings that the main text does not cover, high-dimensional regression with $p > n$ and canonical generalized linear models including logistic regression. Appendix I.4 then treats the non-convex losses deep networks produce, which are reached through

the tracking result of Section 3.4 rather than through the determinant. Throughout we work in the deployed normalized-key update based on Remark 3.1.

One ratio organizes all three. Along direction u_i the delta rule forgets at $(1-\beta)+\eta\hat{\lambda}_i$ per step against EMA's $(1-\beta)$, a ratio of

$$1 + r_i, \qquad r_i := \frac{\eta\hat{\lambda}_i}{1-\beta}, \tag{24}$$

and every gain below comes from that one factor. It shortens the memory window, it speeds relaxation toward a moving target, it shrinks the steady lag bound, and it scales direction i of the implicit preconditioner $(\mu I + \hat{\Sigma})^{-1}$ by $1/(1+r_i)$, which lowers the effective condition number on the data span.

I.1 A Loss-Free Determinant Lemma

The per-direction iteration of (12) was derived for a squared loss, but its determinant never uses the loss. Making that explicit is what lets the same comparison run in every setting below.

Lemma I.1 (Loss-free determinant, restating Lemma 3.9)**.** Along u_i, one step maps the pair (parameter error, buffer component) through

$$A = \begin{pmatrix} 1-\alpha c & -\alpha\tilde{\beta}_i \\ c & \tilde{\beta}_i \end{pmatrix}, \tag{25}$$

where α is the step size and c is the per-direction coupling from the parameter error into the buffer, equal to η times the local curvature along u_i. This is (12) with c left arbitrary, and $c = \eta\kappa_i$ recovers it. Then

$$\det A = \tilde{\beta}_i \quad \text{for AK-MOMENTUM}, \qquad \det A = \beta \quad \text{for EMA momentum}, \tag{26}$$

for any c and any α. The determinant depends only on the memory statistic, never on the loss that produced c. Consequently, by Lemma F.1,

$$\min_{\alpha} \rho(A) = \sqrt{\det A}, \tag{27}$$

so the best achievable per-direction rate falls from $\sqrt{\beta}$ to $\sqrt{\tilde{\beta}_i}$ on every direction with $c > 0$, whatever loss produced it.

Proof. Direct computation gives $\det A = \tilde{\beta}_i(1-\alpha c) + \alpha\tilde{\beta}_i c = \tilde{\beta}_i$, and the same computation with $\tilde{\beta}_i$ replaced by β gives $\det A = \beta$. Neither retains c. Lemma F.1 gives $\rho(A) \geq \sqrt{\det A}$ whenever $\det A > 0$, with equality exactly when the discriminant is non-positive, and Remark F.3 shows the discriminant is non-positive on a band of αc of width $4\sqrt{\tilde{\beta}_i}$, which is non-empty, so the bound is attained at the tuned step size. □

We state the comparison as a bound on the best achievable rate rather than as a claim at fixed α. At a step size held fixed across the two methods, a smaller determinant does not by itself force a smaller spectral radius, since an overdamped iteration can move the other way, which is the same scope Theorem F.2 states.

I.2 High-dimensional Linear Regression

In the regime $p > n$, where p is the input dimension and n the number of samples, the guarantee survives and sharpens.

Off the key span the rule is inert. If $\hat{x}^\top u = 0$ for every observed key, then $\hat{\Sigma}u = 0$ and $\hat{G}u = \mathbb{E}[\delta(\hat{x}^\top u)] = 0$, so both the correction term and the gradient term vanish along u. Neither the EMA buffer nor the AK-MOMENTUM buffer puts mass on a direction the data has not queried, and the difference between them is confined to the queried subspace.

The fixed point stays well posed. Corollary 3.5 gives $M^* = \hat{G}(\mu I + \hat{\Sigma})^{-1}$, which is defined for rank-deficient $\hat{\Sigma}$, exactly the regime in which the unregularized whitening $\hat{G}\hat{\Sigma}^{-1}$ is undefined. The rule therefore regularizes on its own, with the ridge $\mu = (1-\beta)/\eta$ supplied by the decay rather than chosen separately.

The gain grows with spread. Over the queried directions the improvement is the ratio $(\mu+\hat{\lambda}_{\max})/(\mu+\hat{\lambda}_{\min})$, so it is largest when the queried spectrum is spread out, which is the typical spiked scenario in high dimension. A ceiling is available for the normalized variant alone, since $\hat{\lambda}_i \in [0,1]$ gives

$$\operatorname{cond}(\mu I + \hat{\Sigma}) \;\le\; \frac{\mu+1}{\mu} \;=\; 1 + \frac{\eta}{1-\beta}, \tag{28}$$

a bound that is independent of width, of layer, and of the data.

I.3 Canonical GLMs

For a canonical generalized linear model with inverse link f, the per-sample gradient is $(f(w^\top x) - y)x^\top$, which is the same key-value structure the method is built on. Theorem 3.4, Lemma 3.7, and Theorem 3.10 therefore hold unchanged, since their proofs use only $\hat{G}$ and $\hat{\Sigma}$ and never a squared loss.

For the rate, linearize the joint dynamics of the parameter error and the buffer at the optimum $w_\star$. The scalar curvature c of Lemma I.1 becomes the matrix

$$K = \mathbb{E}\big[f'(w_\star^\top x)\, x\hat{x}^\top\big] = \mathbb{E}\big[f'(w_\star^\top x)\, \|x\|\, \hat{x}\hat{x}^\top\big]\,, \tag{29}$$

which is symmetric and positive semidefinite, using $x\hat{x}^\top = \|x\|\,\hat{x}\hat{x}^\top$ and $f' > 0$ for canonical links. If K and $\hat{\Sigma}$ share eigenvectors they diagonalize together, so the system splits into one block per direction, each a copy of (25) with the matching eigenvalue of K in place of c/η, and Lemma I.1 applies block by block. The fixed-point and tracking statements need no such condition.

I.4 Nonlinear Deep Neural Network Training

For non-convex deep losses, tracking is the right tool, because Theorem 3.10 constrains the recursion that produces the buffer and places no condition on the loss surface, so convexity never enters. We strengthen it with two results, stated against each buffer's own target so that scale differences between the two targets do not confound the comparison. We claim no global rate.

Proposition I.2 (Steady lag, restating Proposition 3.11)**.** Suppose the target of the buffer drifts by at most Δ_i per step along u_i. Then the expected tracking error satisfies

$$\limsup_{t\to\infty} \|\mathbb{E}[T_{t,i}]\| \;\le\; \frac{\tilde{\beta}_i \Delta_i}{1-\tilde{\beta}_i} \qquad \text{against EMA's} \qquad \frac{\beta\Delta_i}{1-\beta}. \tag{30}$$

Since $\tilde{\beta}_i < \beta$ on every direction with positive input density and $t \mapsto t/(1-t)$ is increasing on $[0,1)$, the bound is strictly smaller there, and the relaxation-time ratio is $1 + r_i$.

Proof. By Theorem 3.10, one step contracts the tracking error by $\tilde{\beta}_i$, and the drift adds at most Δ_i before the contraction, so $\|\mathbb{E}[T_{t,i}]\| \le \tilde{\beta}_i(\|\mathbb{E}[T_{t-1,i}]\| + \Delta_i)$. The fixed point of this recursion is $\tilde{\beta}_i\Delta_i/(1-\tilde{\beta}_i)$ and the recursion is a contraction, so the limit superior is bounded by it. The EMA statement is the same argument with $\tilde{\beta}_i$ replaced by β, which is (13). Monotonicity of $t/(1-t)$ and $\tilde{\beta}_i < \beta$ give the comparison. The relaxation time is $1/(1-\tilde{\beta}_i)$ against $1/(1-\beta)$, a ratio of $((1-\beta)+\eta\hat{\lambda}_i)/(1-\beta) = 1 + r_i$. □

Corollary I.3 (Uniform lag)**.** *If the drift follows directional density, $\Delta_i = c_0\hat{\lambda}_i$, then the* AK-MOMENTUM *lag is bounded by $\beta c_0/\eta$ uniformly over directions, while EMA's grows linearly in the density as $\beta c_0\hat{\lambda}_i/(1-\beta)$.*

Proof. Substituting into Proposition I.2, the lag is $\tilde{\beta}_i c_0\hat{\lambda}_i/((1-\beta)+\eta\hat{\lambda}_i)$. Bounding the numerator by $\beta c_0\hat{\lambda}_i$ and the denominator below by $\eta\hat{\lambda}_i$ gives $\beta c_0/\eta$, which does not involve i. The EMA bound is $\beta c_0\hat{\lambda}_i/(1-\beta)$, which is linear in $\hat{\lambda}_i$. □

The bridge back to the loss. Split the estimation error into two parts, $M - \hat{G} = (M - M^*) + (M^* - \hat{G})$. The first term is the lag, which is how far the buffer trails its own moving target. Proposition I.2 lowers its steady bound on every direction and Corollary I.3 caps it where EMA's grows with the density. The second term is the designed preconditioning bias of Theorem 3.4 rather than an error. So Proposition D.1 turns reduced lag on the high-density directions, which Figure 3c shows dominate the input variance, into reduced first-order shortfall in the realized loss decrease. Section 4.2 is the empirical form of that bridge, since it measures the gradient-estimator quality, the function-space prediction error, and the input-feature conditioning on the real 24-layer run.

J Mixed-Modality Inputs

A natural question is what happens when the input features are of mixed types, such as tabular data, text, and images together, which carry different natural metrics. The short answer is that nothing in Section 3 changes, because AK-MOMENTUM acts inside a layer on post-embedding activations and never sees raw input types.

The theory is untouched. Every result in Section 3 uses the input distribution only through $\hat{\Sigma}$ and $\hat{G}$. The key second moment of a mixture is the density-weighted combination of the second moments of its components, so heterogeneous input produces a valid $\hat{\Sigma} \succeq 0$ and every statement holds for any such $\hat{\Sigma}$. Mixing modalities changes the key spectrum, which is the very thing the mechanism adapts to.

There is one metric, not several. On the input side, after the embedding every modality lives in the same activation space under the same inner product. Per-sample ℓ_2 normalization also removes residual scale and unit differences before they reach the buffer while preserving directional density, which is scale free by Proposition C.1. On the output side the loss enters only through δ, so a different loss or metric per modality changes what is stored, not how it is stored, and no theorem in Section 3 uses a particular loss. An embedding is needed only in the sense that any network already has one, so no extra machinery and no new theory are required. Our experiments span two modalities trained separately, and an interleaved study is noted as future work in Section 5.

K Interaction with the Second Moment of AdamW

The delta correction and the second moment of AdamW act on different structure, so they compose rather than duplicate. This appendix gives the argument in full. We present it as a discussion rather than as a theoretical claim, since the central step is a heuristic factorization that we do not prove and that is exact only under an independence assumption a trained network does not satisfy.

The two act in different bases. The second moment v is diagonal in the coordinate basis and rescales entries of the update. The delta correction is diagonal in the eigenbasis of $\hat{\Sigma}$ and rescales input directions. Key normalization keeps the roles apart, since $\hat{\Sigma}$ has unit trace and therefore carries direction only, leaving overall scale to Adam.

Where the input side enters each. Heuristically, under $g = \delta x^\top$ with δ and x independent, the second moment factorizes as $v_{ij} \approx \mathbb{E}[\delta_i^2]\,\mathbb{E}[x_j^2]$, so the input-side factor that Adam sees is the diagonal of the input second moment, while the delta rule uses all of $\hat{\Sigma}$. The two therefore coincide only when $\hat{\Sigma}$ is near isotropic or coordinate aligned. Figure 3a rules that out in our models, where the per-layer input-feature condition numbers sit between 10^4 and 10^{10} throughout training. The independence assumption behind the factorization does not hold in a trained network, which is why we do not state this as a claim, but the conclusion it points to does not depend on the assumption being exact, only on $\hat{\Sigma}$ being far from isotropic.

What the delta rule adds on top of the second moment. Both arms of every comparison in Section 4 use the unchanged AdamW second moment, so whatever the delta rule contributes is measured on top of the scaling of Adam. The separations reported in Section 4.2 are measured on the input-feature covariance, which v registers only through its diagonal, so they are outside what v can produce on its own.

The one place they overlap. The two do overlap in the overall size of M relative to $\sqrt{v}$. Learning-rate tuning absorbs that overlap, which is the reason the tuned peak rate for AK-AdamW sits more than ten times below the AdamW value rather than near it (Appendix T). We read the offset as a scale correction and not as evidence that the two preconditioners are competing for the same structure.

L Deployed AK-MOMENTUM Algorithm

This appendix states the algorithm that produces every number reported in Section 4, so that the cost analysis of Appendix M and the empirical claims refer to the same object.

Per linear layer, with a batch of N tokens giving inputs $X \in \mathbb{R}^{N\times n}$, their normalized keys $\hat{X}$, and output-side errors $\Delta \in \mathbb{R}^{N\times m}$, write

$$G = \frac{\Delta^\top X}{N} \in \mathbb{R}^{m\times n}, \qquad \hat{\Sigma} = \frac{\hat{X}^\top \hat{X}}{N} \in \mathbb{R}^{n\times n}, \tag{31}$$

and update the buffer by

$$\boxed{M \;\leftarrow\; \beta M + \eta\big(G - M\hat{\Sigma}\big).} \tag{32}$$

This is the batch form of (4) with the normalized key in the correction term. AK-AdamW then applies the unchanged AdamW machinery to M, that is the second moment $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^{\odot 2}$ taken on the ordinary weight gradient, bias correction, decoupled weight decay, and the preconditioned update $\widehat{M}_t \oslash (\sqrt{\hat{v}_t} + \varepsilon)$. AK-SGD applies $\theta \leftarrow \theta - \alpha M$ directly. The CIFAR-10 MLP run of Section 4.3 instead forms the gradient term against the normalized keys, $G = \Delta^\top \hat{X}/N$, which makes (32) the batch form (34) of the normalized-key rule (16).

Two implementation points matter for the cost.

The gradient term is free. G is the weight gradient that the backward pass already computes, and the AdamW second moment is taken on the same G, so the only optimizer-specific work is the correction $M\hat{\Sigma}$. Appendix M counts exactly that.

The contraction order is chosen per layer. The product $M\hat{\Sigma}$ can be formed either as $M(\hat{X}^\top \hat{X})$, which builds the Gram matrix once and reuses it across every layer reading the same input, or as $(M\hat{X}^\top)\hat{X}$, which never materializes an $n \times n$ matrix. The first is cheaper for square and tall layers and the second for wide ones, and the dispatch rule is given in Appendix M. The second moment $\hat{\Sigma}$ is taken on a tensor the step already forms and adds no GEMM of its own.

M FLOPs Count and Measurement Setup

M.1 Notation and FLOP Convention

For a single linear layer $y = Wx$ with weight $W \in \mathbb{R}^{m\times n}$ where $m = d_{\text{out}}$ and $n = d_{\text{in}}$, batched over $N = B \cdot L_{\text{seq}}$ tokens, with B the batch size in sequences and L_{seq} the sequence length:

Symbol	Meaning	Shape
X	Input activations	$N \times n$
∇Y	Output-side gradient	$N \times m$
W	Weight matrix	$m \times n$
M	Momentum buffer	$m \times n$
V	Second-moment buffer (AdamW)	$m \times n$
$\hat{X}$	Row-normalized inputs, $\hat{x}_i = x_i/\|x_i\|_2$	$N \times n$
d	Transformer hidden dimension	—
γ	Gated-MLP expansion ratio, $d_{\text{ffn}} = \gamma\, d$	—
L_{seq}	Sequence length, distinct from the layer count L	—

We adopt the standard convention that one fused multiply-add (FMA) counts as 2 FLOPs. Hence a matrix multiply of an $a \times b$ and $b \times c$ matrix is counted as $2abc$ FLOPs.

M.2 Standard Linear-Layer FLOPs

The baseline cost of a linear layer (forward + backward, excluding the optimizer step since it is comparatively negligible) is

$$\mathrm{FLOPs}^{\text{layer}}_{\text{baseline}} = \underbrace{2mnN}_{\text{forward } Y=XW^\top} + \underbrace{2mnN}_{\nabla X=\nabla Y\, W} + \underbrace{2mnN}_{\nabla W=\nabla Y^\top X} = 6mnN. \tag{33}$$

M.3 The Normalized-Key AK-MOMENTUM Update

Definition M.1 (Normalized-Key AK-MOMENTUM)**.** The normalized-key delta-rule update is

$$\boxed{M_{t+1} = \beta\, M_t + \frac{\eta}{N}\Big(\nabla Y^\top \hat{X} - M_t\, \hat{X}^\top \hat{X}\Big),} \tag{34}$$

or per-sample,

$$M_{t+1} = \beta M_t + \eta\big(v_t - M_t \hat{x}_t\big)\hat{x}_t^\top, \qquad v_t \coloneqq \delta_t, \tag{35}$$

where $G_t = \delta_t x_t^\top$ is the per-sample gradient and $\delta_t = \partial\mathcal{L}/\partial y_t$.

The factored implementation evaluates (34) as

$$\Delta M = \tfrac{1}{N}\big(\nabla Y^\top \hat{X} - M\hat{X}^\top \hat{X}\big) \equiv \tfrac{1}{N}\big(\nabla Y^\top - M\hat{X}^\top\big)\hat{X}. \tag{36}$$

M.4 Per-Layer Cost of the Correction

The deployed update is (32), and Appendix L explains why only the correction $M\hat{\Sigma}$ is optimizer-specific. This subsection counts that correction per layer, then Appendix M.5 aggregates it over a block.

M.4.1 Cost of Key Normalization

Lemma M.2 (Normalization FLOPs)**.** Computing $\hat{X}$ from X costs

$$\mathrm{FLOPs}_{\mathrm{norm}}(n, N) = 3nN + 2N \approx 3nN. \tag{37}$$

Proof. Per row i, computing $\|x_i\|_2^2 = \sum_j x_{ij}^2$ takes n multiplies plus $n-1$ adds, which is $2n-1 \approx 2n$ FLOPs, the square root takes 1, the reciprocal takes 1, and scaling all n entries takes n multiplies. Aggregating over N rows gives $N(3n+2) = 3nN + 2N$. □

This is sub-leading against a single GEMM at $2mnN$ by a factor of $3/(2m)$, which is about 0.15% at $m = 1024$. The normalization is computed once per distinct input and shared by every layer reading it.

M.4.2 Two Contraction Orders and the Dispatch Rule

The correction $M\hat{\Sigma}$ admits two evaluation orders with different costs.

Gram order. Form $\hat{G}_{\mathrm{gram}} = \hat{X}^\top \hat{X}$ at $2n^2N + n^2$, then $M\,\hat{G}_{\mathrm{gram}}$ at $2mn^2$, for a total of $2n^2N + 2mn^2$ up to lower-order terms. The Gram matrix is formed once per distinct input and reused by every layer that reads that input, so the $2n^2N$ term is shared while the $2mn^2$ term is per layer. Its spectrum lies in $[0, N]$ before the $1/N$ scaling, so the product is numerically stable.

Through-N order. Form $M\hat{X}^\top$ at $2mnN$, then multiply by $\hat{X}$ at $2mnN$, for a total of $4mnN$, never materializing an $n \times n$ matrix.

Comparing the two leading terms, $2n^2N$ against $4mnN$, the Gram order is cheaper exactly when $n < 2m$. This gives the dispatch rule used throughout.

$$M\hat{\Sigma} \;=\; \begin{cases} M\,(\hat{X}^\top \hat{X}) & \text{if } n < 2m \quad \text{(Gram order)}, \\ (M\hat{X}^\top)\,\hat{X} & \text{otherwise} \quad \text{(through-}N\text{ order)}. \end{cases} \tag{38}$$

M.5 Per-Block Count and the Overhead Band

M.5.1 Baseline Per-Block FLOPs

A standard transformer block with a gated MLP of expansion ratio γ holds four attention projections at $d \times d$, two gated-MLP matrices at $\gamma d \times d$, and one down projection at $d \times \gamma d$. Using (33) for each,

$$\mathrm{FLOPs}_{\text{block}}^{\text{baseline}}(\gamma) \;=\; \underbrace{(24 + 18\gamma)\, d^2 N}_{\text{linear (QKVO + gated MLP)}} \;+\; \underbrace{12\, d L_{\text{seq}} N}_{\text{attention scores/values}} \;. \tag{39}$$

For the Llama-2 ratio $\gamma = 8/3$, the linear term is $72\, d^2 N$.

M.5.2 Aggregating the Correction

Applying (38) layer by layer, and sharing each Gram matrix across the layers that read the same input, gives the following.

Group	Shape	Order	d^2N coeff.	d^3 coeff.
W_Q, W_K, W_V	$d \times d$	Gram, shared over one input	2	6
W_O	$d \times d$	Gram	2	2
$W_{\text{up}}, W_{\text{gate}}$	$\gamma d \times d$	Gram, shared over one input	2	4γ
W_{down}	$d \times \gamma d$	through-N	4γ	0
Total			$\mathbf{6 + 4\gamma}$	$\mathbf{8 + 4\gamma}$

The first three rows each contribute one shared $2d^2N$ term rather than one per layer, because W_Q, W_K, W_V read one and the same input and $W_{\text{up}}, W_{\text{gate}}$ read another. Sharing applies to the Gram matrix only. The multiplication by M that follows it is per layer at $2mn^2$, which carries no factor of N and is what the d^3 column collects. The down projection maps the expanded hidden state back to the model width, so $m = d$ and $n = \gamma d$ there. Every gated MLP in use has $\gamma \geq 2$, which makes $n \geq 2m$, so this layer takes the through-N order at $4mnN = 4\gamma d^2 N$. Key normalization costs $3nN$ per distinct input by Lemma M.2, which over the four inputs above, of dimensions d, d, d, and γd, gives $(9 + 3\gamma)dN$. The block count is therefore

$$\Delta\mathrm{FLOPs}_{\text{block}}(\gamma) = (6 + 4\gamma)\, d^2 N \;+\; (8 + 4\gamma)\, d^3 \;+\; (9 + 3\gamma)\, dN. \tag{40}$$

The two lower-order terms enter the ratio as $(8 + 4\gamma)d/N$ and $(9 + 3\gamma)/d$, which at $N = 524{,}288$ tokens per step and our widths of 384, 1024, and 1792 move it by about a tenth of a percentage point, so the leading term is the block-level number.

M.5.3 The Overhead Band

Theorem M.3 (Per-block overhead of the deployed rule)**.** *Under the dispatch rule* (38) *with normalized keys and $\gamma \geq 2$, the per-block cost of the correction relative to the linear forward and backward of* (39) *is*

$$\boxed{f(\gamma) \;=\; \frac{6 + 4\gamma}{24 + 18\gamma},} \tag{41}$$

which is strictly decreasing on $\gamma \geq 0$ and therefore satisfies

$$22.22\% \;=\; \tfrac{4}{18} \;<\; f(\gamma) \;\leq\; \tfrac{6}{24} \;=\; 25.0\%, \tag{42}$$

with $f(8/3) = 50/216 = 23.148\%$ at the Llama-2 ratio used in all our models.

Proof. The ratio follows from (40) and (39) by dropping the two lower-order terms. The endpoints are $f(0) = 6/24 = 25.0\%$ and $f(\gamma) \to 4/18 = 22.22\%$ as $\gamma \to \infty$. Endpoints alone do not give a band, so we add that $f'(\gamma) = -12/(24 + 18\gamma)^2 < 0$ for every $\gamma \geq 0$, which makes f strictly decreasing and puts every value between the two. The down projection is the only layer whose order depends on γ, so the count above holds for $\gamma \geq 2$ and the attainable values sit in the lower part of the range. □

The band is what makes the cost predictable across architectures. Any gated-MLP shape lands inside a range of under three percentage points, so the block-level number barely moves when the expansion ratio changes. Every gated MLP in use has $\gamma \geq 2$, so the attainable values sit in the narrower range from 22.22% to $f(2) = 23.33\%$, and the 25.0% endpoint is the limit at $\gamma = 0$ rather than a shape any such block takes.

M.6 Realized Per-Step Overhead and Measurement Setup

Theorem M.3 is a block-level statement about linear layers. Two effects at finite scale place the realized per-step overhead below it.

1. the attention $\mathcal{O}(dL_{\text{seq}}N)$ term of (39) inflates the baseline at $L_{\text{seq}} = 2048$.
2. the output projection maps the width d to the vocabulary size n_{vocab}, so it adds $6\,d\,n_{\text{vocab}}N$ to the baseline but only about $2d^2N$ to the correction, since its Gram matrix is $d \times d$. It therefore enters the ratio at about $d/(3\,n_{\text{vocab}})$, which is 0.4%, 1.1%, and 1.9% at our three widths and far below the block value. The input embedding is a lookup rather than a GEMM and enters neither side of the count.

Both enlarge the denominator of the overhead ratio far more than the numerator, and both shrink relative to the leading d^2N term as width grows, which is why the realized number rises with scale toward the band.

Measurement setup. We do not estimate the realized overhead from the closed form. Our training suite counts it with PyTorch's FLOP counter over the forward and backward of a full optimizer step, which charges every matrix multiplication there, including those that form the correction, and charges attention densely, without halving it for the causal mask and including the recomputation of the attention scores in its backward pass, which gives $14\,dL_{\text{seq}}N$ per layer against the $12\,dL_{\text{seq}}N$ of (39). The counter does not charge elementwise work, so the key normalization adds time but no counted FLOPs, and for Gram-order layers the once-per-step product of M with the Gram matrix runs in the optimizer step, outside the counted region. These are the two lower-order terms of the per-block count above, which move the ratio by about a tenth of a percentage point. The count does not depend on how a step is split into microbatches, and the cost axes count training FLOPs only, since evaluation adds the same amount to both arms. All three models use $L = 24$, $d_{\text{head}} = 64$, $n_{\text{vocab}} = 32{,}000$, untied input and output embeddings, and $\gamma \approx 2.67$, with $d_{\text{ffn}} = \lfloor 2.67d \rfloor$, $B = 256$ and $L_{\text{seq}} = 2048$, hence $N = 524{,}288$ tokens per step. Step time is measured in the training loop on a single H200, with activation tracing for monitoring turned off and the same microbatch size in both arms. Each value comes from at least three repeated measurements of a few hundred steps past warmup, each summarized by the median of its fastest quarter of steps, leaving out steps next to evaluations and checkpoint writes. Cumulative wall clock in Figure 4 is this per-step time times the step count, so it covers training steps only and leaves out startup, evaluations and checkpoint writes, which add less than 1% to each run in the figure.

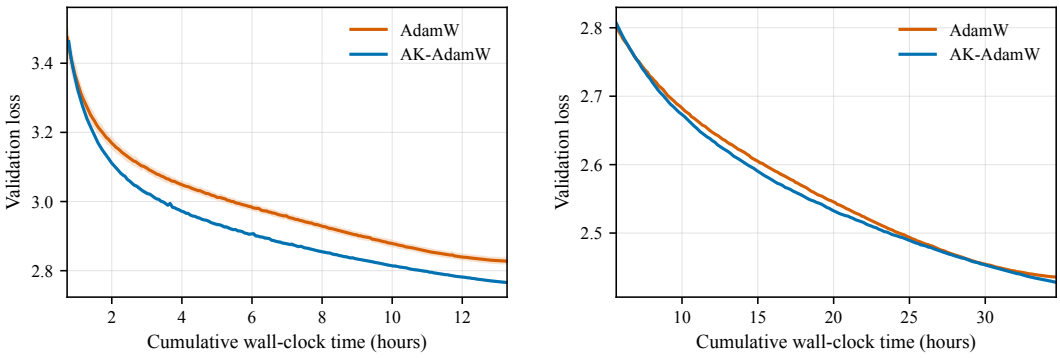
Results.

Scale	d	d_{ffn}	Realized per-step FLOP overhead	Measured step time
67M	384	1025	$\mathbf{10.0}\%$	$1.1526\times$ AdamW
370M	1024	2734	$\mathbf{15.5}\%$	$1.1775\times$ AdamW
1B	1792	4784	$\mathbf{18.1}\%$	$1.1844\times$ AdamW

The counted overhead rises with width across all three scales, from 10.0% to 18.1%, approaching the band of Theorem M.3 from below as the two finite-scale effects above shrink relative to the leading d^2N term. Measured step time rises with it, from $1.1526\times$ to $1.1844\times$, and sits above the count by 5.3, 2.2, and 0.3 percentage points at 67M, 370M, and 1B. The gap comes from the rate at which the correction runs rather than from its amount of work. Every GEMM of the correction runs on tensor cores, but at small width the through-N product of the down projection, whose output has only d rows, is too thin to fill the GPU, and the key normalization reads and writes the activations while adding almost no FLOPs. The AK-AdamW step runs at 95.4% of the FLOP rate of the AdamW step at 67M. Both effects shrink as width grows, and this rate reaches 98.1% at 370M and 99.7% at 1B, where step time tracks the count. Reported step times use our PyTorch implementation, which

compiles the key normalization into one fused kernel and forms each correction product with a single GEMM call. A kernel that normalizes keys as it loads them, tiled for the thin products at small width, would narrow the remaining gap.

What this costs at matched loss. Table 3 reports the end-to-end consequence, measured over the full horizon with identical data order and initialization in both arms. Averaged over the matched validation-loss levels past the crossing point on each axis, AK-AdamW reaches the same loss with 35.1% fewer FLOPs and 32.0% less wall clock at 67M, and 5.6% fewer FLOPs and 4.0% less wall clock at 370M. Figure 4 shows the wall-clock curves. Because the correction is paid from the first step, the two arms cross once on each cost axis and AK-AdamW is ahead of AdamW everywhere after that point. The cost curves are the seed means of Figure 1, since every seed has the same cost per step.



(a) 67M. (b) 370M.

Figure 4: Validation loss against cumulative wall clock on a single H200, from step 2,000 to the end of the AdamW run. Curves are the seed means of Figure 1 with per-seed bands, and cumulative wall clock is the per-step time of Appendix M.6 times the step count, with the same microbatch size in both arms. At 67M AK-AdamW is ahead throughout. At 370M it starts behind, because the correction is paid from the first step, crosses near 7.0 hours, and stays ahead for the rest of the run. When AdamW finishes, AK-AdamW is 0.062 nats lower at 67M and 0.008 nats lower at 370M.

M.7 Peak-Memory Derivation

M.7.1 Persistent State

Optimizer	Buffers	Bytes/param (fp32)
SGD-momentum	M	4
AdamW	M, V	8
AK-SGD (norm-key)	M	**4**
AK-AdamW	M, V	**8**

Result M.4 (Persistent overhead = 0)**.** The persistent optimizer-state memory overhead of normalized-key AK-SGD against SGD-momentum, and of normalized-key AK-AdamW against AdamW, is exactly zero.

M.7.2 Saved Activations

The standard backward already saves X for ∇W. The factored delta also requires X (to build $\hat{X}$). Hence:

Result M.5 (Saved-activation overhead = 0)**.** No additional activation tensors must be saved between forward and backward.

M.7.3 Transient Memory During the Optimizer Step

Factored path.

Tensor	Shape	Bytes (fp32)
$\|x_i\|_2$	N	$4N$
$\hat{X}$	$N \times n$	$4Nn$
$T = M\hat{X}^\top$	$m \times N$	$4mN$
$R = \nabla Y^\top - T$	$m \times N$	in-place over T
U	$m \times n$	accumulates into M
Peak transient		$\mathbf{4Nn + 4mN + 4N}$

Gram path.

Tensor	Shape	Bytes (fp32)
$\|x_i\|_2$	N	$4N$
$\hat{X}$	$N \times n$	$4Nn$
$\hat{G}$	$n \times n$	$4n^2$
$C = M\hat{G}$	$m \times n$	accumulates
Peak transient		$\mathbf{4Nn + 4n^2 + 4N}$

For tall/square layers with $N \gg n^2/m$, the Gram path is strictly more memory-efficient. The cost from key normalization itself is the $4Nn$ term, but if we permit *in-place* replacement of X by $\hat{X}$ (the standard backward no longer needs X once $\hat{X}$ is formed and ∇W has been computed for Gram layers), this term is absorbed into the existing X allocation.

M.7.4 Per-Block Transient Peak Overhead

Under fused backward+optimizer execution (per-layer immediate update, no global ∇W retention across the network):

$$\boxed{\text{PeakMemOverhead}_{\text{block}}^{\text{norm-key}} \le 4\,N \cdot n_{\max} \text{ bytes (fp32)}} \tag{43}$$

where $n_{\max} = \gamma d$ for the FFN-down input.

Result M.6 (Concrete peak overhead)**.** For $d = 1024$, $\gamma = 8/3$, fp16 storage, and a microbatch of $N = 8192$ tokens rather than the 524,288 tokens of a full optimizer step, since only one microbatch of $\hat{X}$ is live at a time, the largest normalized-key transient allocation is

$$2 \cdot 8192 \cdot (8/3) \cdot 1024 \approx 44.7 \text{ MB}, \text{ per block.} \tag{44}$$

This is fully transient, since only one block's $\hat{X}$ is live at a time.

M.8 Theoretical Lower Bound

M.8.1 Sequential Read–Write Lower Bound

The delta rule's intrinsic operation per sample comprises:

1. *Read:* $\hat{v}_t = M_t\hat{x}_t$, costing $2mn$ FLOPs.
2. *Write:* $M \mathrel{+}= \eta(v_t - \hat{v}_t)\hat{x}_t^\top$, costing $2mn$ FLOPs.

Aggregating over N samples gives a lower bound of $4mnN$ for batched algorithms, or $2mnN$ for fully sequential ones, which is the through-N order of (38). The Gram order beats it whenever $n < 2m$ by moving the N contraction into a matrix that several layers share.

M.8.2 Tightness of the Asymptotic Result

The dispatch rule together with cross-layer Gram sharing yields (40), which is the theoretical minimum under three constraints, GPU-batched execution, a standard transformer architecture with no architectural relaxation, and independent block computation with no cross-block sharing. The

Table 7: Model architectures. All three belong to the same width-only μP family, with d_{head}, n_{layers}, and the SwiGLU expansion ratio fixed across scales, so scale enters only through d_{model}.

Scale	d_{model}	n_{layers}	n_{heads}	d_{head}	Vocab	Params
67M (proxy)	384	24	6	64	32,000	$\sim$67M
370M (target)	1024	24	16	64	32,000	$\sim$370M
1B (target)	1792	24	28	64	32,000	$\sim$1B

leading coefficient $(6+4\gamma)$ saturates two structurally independent lower bounds at once, the per-layer read and write bound above and the Gram-shared bound from cross-layer input reuse. Further reduction would require relaxing one of the three, for example low-rank or sparse keys, a custom fused weight-gradient kernel, or cross-block input sharing, which the architecture does not allow.

N Experimental Setup

Architectures. Across all language modeling experiments we use a Llama-2-style [30] decoder-only transformer with SwiGLU FFNs (expansion ratio 2.67), rotary position embeddings (RoPE, base 10,000), pre-RMSNorm, no biases on linear layers, and untied input/output embeddings. Table 7 lists the three scales used in the main experiments. All three belong to the same width-only μP family: $d_{\text{head}} = 64$, $n_{\text{layers}} = 24$ and the SwiGLU expansion ratio are held fixed across scales so that scale enters only through d_{model}. The 67M model is the proxy at which all hyperparameters are tuned, and the 370M and 1B models are the μP transfer targets.

Data and training protocol. We pretrain on the 10BT subset of FINEWEB-EDU [22], tokenized with the Llama-2 SentencePiece tokenizer (vocabulary size 32,000). Sequence length is 2048 and the effective batch size is 256 sequences (524,288 tokens per optimizer step), giving 19,073 optimization steps for one full pass over the 10B-token budget. Training uses bfloat16 mixed precision, gradient norm clipping at 1.0, a cosine learning-rate schedule with linear warmup over the first 500 steps and decay to a floor of 10^{-2} of the peak rate, and no dropout. The validation set is a held-out 0.5% slice of FineWeb-Edu (not seen during training). The 1B run uses the 20BT budget, which is Chinchilla-optimal at that size, with the cosine cycle stretched to the longer horizon and the warmup held fixed at 500 steps, identically for both optimizers.

The matched-level measurement window. Every loss gap, step reduction, and cost saving we report is measured over matched validation-loss levels taken from step 2,000 onward, which is also the range the figures plot. The window drops the warmup and the transient right after it, where the two curves have not yet separated and a small difference in loss maps to a very large difference in steps, so a metric read there says more about the schedule than about the buffer. The window was fixed once, before any of the numbers below were computed, and the same window is applied at every scale, to every arm, and to both cost axes.

Baselines and AK-MOMENTUM instantiation. Our headline comparison is AK-AdamW against AdamW. AK-AdamW is obtained by replacing AdamW's first-moment EMA accumulator with the AK-MOMENTUM update of Equation (32), and the second moment, bias correction, and decoupled weight decay are unchanged. The layer inputs that the correction needs are captured per linear layer through forward hooks, and the captured tensors are released immediately after the optimizer step (Appendix M). The pure delta limit ($\beta = 1$) is not used in our main runs, and we always operate with $\beta \in [0.9, 1)$ so that AK-MOMENTUM admits a uniform decay floor along every direction.

Hyperparameter selection via μP transfer. All hyperparameters are tuned on the 67M proxy and transferred to 370M and 1B via the μP scaling rules for hidden weights: $\text{lr}_{\text{eff}} = \text{lr}_{\text{proxy}} \cdot (d_{\text{base}}/d_{\text{model}})$ for both AdamW and AK-AdamW, while the delta coefficient η and the EMA coefficient β_1 transfer directly without rescaling. The η-invariance follows from the per-sample key normalization $\|\hat{x}_t\| = 1$ (Theorem Q.1 and Theorem C.2), and is verified empirically by a μP coordinate check at widths $\{128, 256, 512, 1024, 2048\}$ (Appendix S).

We use Optuna [2] with a TPE sampler and a SuccessiveHalvingPruner to search over learning rate, β_1, η (AK-AdamW only), and the auxiliary learning rate for the embedding and output-head group, with weight decay fixed at 10^{-1} and $\beta_2 = 0.99$. The auxiliary rate is tuned as its own axis

Table 8: Optuna-selected hyperparameters at the 67M proxy and their μP-transferred values at the two targets. The η coefficient and β_1 do not rescale with width. "aux lr" is the auxiliary rate for the embedding and output-head group, tuned as its own axis.

optimizer	lr@67M	lr@370M	lr@1B	aux lr@67M	β_1	β_2	η	wd
AdamW	4.0×10^{-3}	1.5×10^{-3}	8.6×10^{-4}	—	0.95	0.99	—	0.1
AK-AdamW	3.0×10^{-4}	1.13×10^{-4}	6.4×10^{-5}	1.24×10^{-2}	0.99	0.99	0.37	0.1
Muon	8.0×10^{-3}	4.90×10^{-3}	—	1.28×10^{-2}	0.95	—	—	0.1

for AK-AdamW and for Muon, because tying it to the main rate starves those parameters in both. AdamW uses one rate for all parameter groups, which is the standard configuration for it and the one its own μP rule expects, so it carries no separate auxiliary axis. Each trial trains the 67M proxy for 4,000 steps with the LR scheduler period set to the full 19,073-step horizon so that trial losses are comparable to early-training losses of the full run. Appendix T gives the final range of every axis, the pruner settings, the budget accounting, and the boundary audit. The configurations selected for the headline runs are listed in Table 8.

Compute and seeds. All experiments are run on H200 GPUs (local cluster). On a single H200, a 67M training run takes roughly 13 hours with AdamW and 15 hours with AK-AdamW, a 370M run roughly 35 and 41 hours, and a 1B run roughly 112 and 132 hours. These figures are the cost of running each configuration and not a throughput comparison across scales, because the microbatch size is not the same at every scale and wider models run their matrix multiplications closer to the peak rate of the GPU. The like-for-like timing comparison is the training-loop step time of Section M.6, which is measured on a single H200 with the same microbatch size in both arms. The 67M and 370M FineWeb-Edu comparisons are each repeated over 3 seeds (42, 777, 1234) for both AdamW and AK-AdamW, and we report per-seed bands at both scales. The Muon arm uses 3 seeds at each of 67M and 370M, with the same seed set (42, 777, 1234). The 1B comparison is single-seed because of compute, and we report it as a scale check rather than a significance claim.

O Comparison with Structured Optimizers

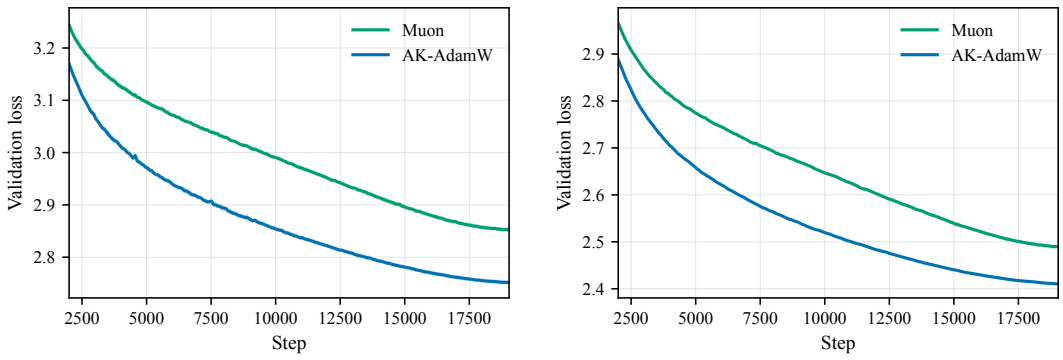
This appendix reports the tuned Muon comparison in full and states why Shampoo and SOAP are absent. Section 4.1 summarizes the finding.

Table 9: AK-AdamW against a tuned Muon baseline on FineWeb-Edu, across matched validation-loss levels, in the same format as Table 2. Loss gap is in nats and step reduction is the fraction of Muon steps saved to reach the same loss, both reported as mean and maximum over the matched levels with the seed standard deviation. Matched levels are taken over the same step-2,000 window as Table 2.

Scale (seeds)	Mean loss gap	Max loss gap	Mean step red.	Max step red.
67M (3 seeds)	0.120 ± 0.008	0.140 ± 0.007	$49.4 \pm 2.4\%$	$55.8 \pm 1.9\%$
370M (3 seeds)	0.109 ± 0.003	0.132 ± 0.003	$41.6 \pm 0.8\%$	$48.1 \pm 0.7\%$

Why AdamW is the isolating baseline. The empirical contribution of this paper is the delta-rule update applied to the first-moment buffer, and isolating it requires a baseline that differs from AK-AdamW only in that update. AdamW is the unique such baseline. A head-to-head against a structured optimizer instead conflates the first-moment contribution with Newton–Schulz orthogonalization or with Kronecker-factored preconditioning, which act at a different level of the stack. That is why Section 4.1 leads with AdamW and treats Muon as a reference point rather than as the isolating comparison, and why the Muon numbers are reported here rather than beside the headline result.

Muon protocol. Muon was tuned under the same Optuna setup as the other two arms, with the same sampler, the same pruner, and the same trial horizon. Budgets were 33, 32, and 38 full-trial equivalents per tuned dimension for AdamW, AK-AdamW, and Muon, so Muon received the largest per-dimension budget of the three. Two further points make the tuning fair rather than merely equal in budget. Muon was tuned under its own μP rule, $\alpha = \Theta(n^{-1/2})$ with the $\sqrt{m/n}$ prefactor [24], rather than under an Adam-family rule [35], since transferring an Adam rule to a spectrally normalized update is not a like-for-like transfer. And the auxiliary AdamW learning rate for the embedding and



(a) 67M, 3 seeds.

(b) 370M, 3 seeds.

Figure 5: AK-AdamW against a tuned Muon baseline on FineWeb-Edu, validation loss against optimizer step, seed means with per-seed bands. The sign of the gap is consistent in every run, and the two curves do not cross at either scale.

output-head group was decoupled and tuned as its own axis for Muon exactly as for AK-AdamW. The 370M Muon run transfers the 67M champion with no re-tuning under that same rule, which is the protocol the other two optimizers also follow.

Scope of the Muon comparison. We make no general claim about Muon from Table 9. We report it as a baseline tuned inside our protocol rather than as a ceiling for the method.

Shampoo and SOAP. We did not run Shampoo or SOAP. Both keep a scalar-decay EMA first moment and add Kronecker-factored preconditioning on top of it, so the delta rule changes a different component of the stack and would slot into their buffers unchanged. The comparison that isolates our contribution is the EMA-against-delta swap at a fixed base optimizer, which is the argument of Appendix B, and we read Shampoo and SOAP as complementary rather than competing. The same holds for Muon, where the delta buffer would replace the EMA that feeds Newton–Schulz. AK-SHAMPOO, AK-SOAP, and AK-MUON are identified as future work in Section 5.

P Mechanistic Diagnostics of Momentum Dynamics

P.1 Explanation of Momentum–Gradient Cosine Similarity

The buffer–gradient cosine in stochastic optimization is *not* expected to approach $+1$, and is in general not even of definite sign. A Taylor expansion of $g_t = \nabla\mathcal{L}(\theta_t)$ around θ_{t-1} together with the update rule $\theta_t = \theta_{t-1} - \alpha\, M_{t-1}/(\sqrt{V_{t-1}} + \varepsilon)$ yields

$$\langle M_{t-1}, g_t\rangle \;=\; \langle M_{t-1}, g_{t-1}\rangle \;-\; \alpha\,\frac{\langle M_{t-1}, H_{t-1}\, M_{t-1}\rangle}{\sqrt{V_{t-1}} + \varepsilon} \;+\; \langle M_{t-1}, \xi_t\rangle, \tag{45}$$

where $H_{t-1} \succeq 0$ is the local Hessian and ξ_t is stochastic gradient noise. The middle term is non-negative and biases $\cos(M_{t-1}, g_t)$ downward by an amount that depends on local curvature and step size, not on the buffer rule. This deterministic "stepping-rebound" effect applies symmetrically to EMA and AK-MOMENTUM, since it is a property of stochastic descent itself rather than of the optimizer. The diagnostic quantity is therefore the *gap between optimizers at matched training step*, and equivalently the prediction error e_t, which is monotone in $1 - \cos(M_t, g_t) \cdot \|M_t\|_F/\|g_t\|_F$.

Q Maximal-Update (μP) Scaling

We derive the width-scaling rule for the delta-rule coefficient η in the normalized variant under maximal update parameterization [35], showing that η is width-invariant and that the base optimizer's μP learning-rate rule is inherited unchanged. The analysis is presented for a single hidden linear layer, and the multi-layer extension follows by the same per-layer reasoning, since the delta-rule correction is layer-local.

Q.1 Setup and Per-Coordinate Scales

Consider a hidden linear layer $W \in \mathbb{R}^{m\times n}$ at width n with $m = \Theta(n)$. Under μP, the input $x \in \mathbb{R}^n$ has per-coordinate scale $\Theta(1)$, so $\|x\|^2 = \Theta(n)$ and the unit key satisfies $\|\hat{x}\|^2 = 1$ with per-coordinate scale $\Theta(n^{-1/2})$. The output-side error $\delta \in \mathbb{R}^m$ has per-coordinate scale $\Theta(1)$ for hidden layers under μP. The gradient $g = \delta x^\top$ has per-coordinate scale $\Theta(1)$, and the EMA momentum buffer $M^{\mathrm{EMA}} = \mathbb{E}[g]$ has per-coordinate scale $\Theta(n^{-1/2})$ under the random-sign cancellation of Corollary Q.2.

Target scale for the AK-MOMENTUM buffer. The normalized-key rule (16) uses the normalized key $\hat{x}$ in both terms. The natural injection signal is $\eta\delta_t\hat{x}_t^\top$, which has per-coordinate scale $\Theta(\eta)\cdot\Theta(1)\cdot\Theta(n^{-1/2}) = \Theta(\eta/\sqrt{n})$. Setting $\eta = \Theta(1)$ gives a buffer with per-coordinate scale $\Theta(n^{-1/2})$. We show below that this is the consistent fixed-point scale under the recurrence (16) and that this scale matches the postprocessing requirements of each base optimizer, recovering the standard μP learning-rate rules.

Q.2 Width Invariance of η

Theorem Q.1 (μP width invariance of η)**.** *Under μP with input x of per-coordinate scale $\Theta(1)$ and output error δ of per-coordinate scale $\Theta(1)$, the normalized* AK-MOMENTUM *recurrence*

$$M_t = \beta M_{t-1} + \eta(\delta_t - M_{t-1}\hat{x}_t)\hat{x}_t^\top \tag{46}$$

admits a fixed-point per-coordinate scale of $\Theta(n^{-1/2})$ if and only if the delta-rule coefficient is width-invariant: $\eta = \Theta(1)$.

Proof. We track per-coordinate scales of M_t in steady state. Suppose M_{t-1} has per-coordinate scale s, i.e. $|M_{t-1,ij}| = \Theta(s)$. We compute the per-coordinate scale of each term in (46).

Decay term. βM_{t-1} has per-coordinate scale $\Theta(s)$.

Injection term. $\eta\,\delta_t\hat{x}_t^\top$ has per-coordinate scale $\Theta(\eta)\cdot\Theta(1)\cdot\Theta(n^{-1/2}) = \Theta(\eta/\sqrt{n})$.

Retrieval term. The vector $M_{t-1}\hat{x}_t \in \mathbb{R}^m$ has i-th coordinate $\sum_{j=1}^n M_{t-1,ij}\hat{x}_{t,j}$, a sum of n terms each of order $\Theta(s)\cdot\Theta(n^{-1/2}) = \Theta(s/\sqrt{n})$. Under random sign cancellation (the standard μP fluctuation argument applied to the inner product between a buffer row and the unit key), the magnitude of this sum is $\Theta(\sqrt{n}\cdot s/\sqrt{n}) = \Theta(s)$. Multiplying by $\eta\hat{x}_t^\top$ then gives per-coordinate scale $\Theta(\eta)\cdot\Theta(s)\cdot\Theta(n^{-1/2}) = \Theta(\eta s/\sqrt{n})$.

Fixed-point balance. At steady state $|M_t| = |M_{t-1}| = \Theta(s)$, so the largest term on the right-hand side of (46) must scale as $\Theta(s)$. Since the decay term is $\Theta(s)$ and the retrieval term is $\Theta(\eta s/\sqrt{n}) = o(s)$ for any $\eta = O(1)$, the binding constraint comes from the injection term: we need $\Theta(\eta/\sqrt{n}) = \Theta(s)$, giving $s = \Theta(\eta/\sqrt{n})$. For the resulting M_t to have a width-invariant per-coordinate *ratio* relative to its expectation, which is the standard μP feature-learning condition, we require $\eta = \Theta(1)$, hence $s = \Theta(n^{-1/2})$.

Conversely, if $\eta = \Theta(n^{-c})$ for any $c > 0$, the buffer scale becomes $\Theta(n^{-1/2-c})$, which violates feature learning. If $\eta = \omega(1)$, the retrieval term diverges relative to the decay term and stability fails. Hence $\eta = \Theta(1)$ is necessary and sufficient. □

The deployed rule. The deployed rule (32) has the forgetting operator of (46), and its gradient term is the weight gradient of the backward pass. Its fixed point M^* is (8) with $\hat{G}$ replaced by $\bar{G}$ (Section 3.1), so $M^*u_i = \bar{G}u_i/(\mu + \hat{\lambda}_i)$ for each eigenvector u_i of $\hat{\Sigma}$, where $\bar{G}$ is the fixed point of the EMA buffer, and since $0 \le \hat{\lambda}_i \le 1$ (Proposition C.1) the factor $1/(\mu+\hat{\lambda}_i)$ lies between $1/(1+\mu)$ and $1/\mu$. Stability requires $\eta < 1 + \beta$ (Appendix C), so $\mu = (1-\beta)/\eta$ is bounded below, and the deployed buffer has the per-coordinate scale $\Theta(n^{-1/2})$ of the EMA buffer if and only if η does not vanish with width, that is $\eta = \Theta(1)$.

Q.3 Inherited Base-optimizer Learning-Rate Rules

Corollary Q.2 (Inherited μP learning-rate rules)**.** *With $\eta = \Theta(1)$, the normalized* AK-MOMENTUM *buffer M_t has per-coordinate scale $\Theta(n^{-1/2})$, identical to the per-coordinate scale of the EMA momentum buffer (because EMA's buffer estimates the* population gradient $\bar{G} = \mathbb{E}[\delta x^\top]$*, whose per-coordinate expectation has scale $\Theta(n^{-1/2})$ under μP with random-sign cancellation, and the per-sample buffer scale is $\Theta(1)$ while its expectation exhibits the same $\Theta(n^{-1/2})$ shrinkage). Consequently, every postprocessing operation on M_t inherits the same scaling as it would under EMA, and the base-optimizer's standard μP learning-rate rule applies unchanged:*

- AK-SGD: $\alpha = \Theta(1)$ *for hidden layers, $\Theta(1/n)$ for output heads (matching SGD-with-momentum under μP).*
- AK-AdamW: *the preconditioned update $M_t/\sqrt{v_t + \varepsilon}$ has per-coordinate scale $\Theta(1)$ (because both M_t and $\sqrt{v_t}$ scale as $\Theta(n^{-1/2})$), so $\alpha = \Theta(1/n)$ for hidden layers, matching AdamW under μP.*
- AK-Muon: *the Newton–Schulz output $\mathrm{NS}(M_t)$ has unit singular values and per-entry scale $\Theta(n^{-1/2})$ for an $m \times n$ matrix with $m = \Theta(n)$, so $\alpha = \Theta(n^{-1/2})$ with the $\sqrt{m/n}$ prefactor of Muon, matching Muon under μP.*

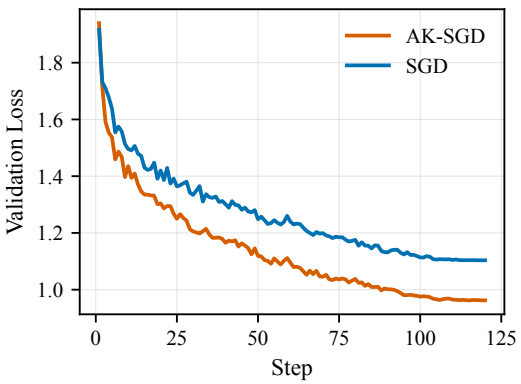
Proof. Per-coordinate scale of M_t is $\Theta(n^{-1/2})$ by Theorem Q.1. The Adam second moment v_t averages $g_t^{\odot 2}$ for the weight gradient g_t of each step, whose entries are averages over the N tokens of the step. Since N is far larger than the width n, these entries have the per-coordinate scale $\Theta(n^{-1/2})$ of $\bar{G}$. Hence $\sqrt{v_t} = \Theta(n^{-1/2})$ and $M_t/\sqrt{v_t} = \Theta(1)$. With this unit-scale preconditioned update, AdamW's standard μP learning-rate rule $\alpha = \Theta(1/n)$ for hidden weights gives $\alpha M_t/\sqrt{v_t} = \Theta(1/n)$ per coordinate, i.e. feature learning. Newton–Schulz produces an orthogonal-like matrix with unit singular values, and for an $m \times n$ matrix with $m, n = \Theta(n)$ this forces per-entry scale $\Theta(n^{-1/2})$, matching the standard Muon analysis. SGD without preconditioning uses $\alpha = \Theta(1)$ on the buffer of scale $\Theta(n^{-1/2})$, giving $\alpha M_t = \Theta(n^{-1/2})$ per coordinate, the standard μP feature-learning scale. □

Remark Q.3 (μTransfer)**.** Theorem Q.1 implies that η, once tuned at any width, transfers without rescaling to any other width. Combined with Corollary Q.2, this means the AK-MOMENTUM family is *μP-transparent*: it adds exactly one additional hyperparameter (η) over the base optimizer, and that hyperparameter requires no width-dependent scaling. We empirically verify width invariance of η via coordinate checks in Appendix S.

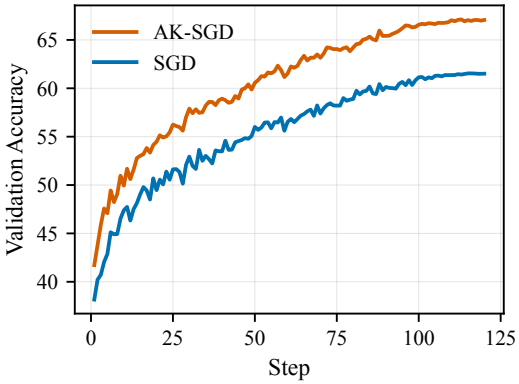
Remark Q.4 (Why normalization matters for μP)**.** The width-invariance of η rests on the unit norm $\|\hat{x}\| = 1$, which makes the spectral norm of the rank-one operator $\hat{x}_t \hat{x}_t^\top$ equal to 1 independently of n. For the unnormalized variant, $\|x\|^2 = \Theta(n)$, and the corresponding operator $x_t x_t^\top$ has spectral norm $\Theta(n)$, to maintain stability we would require $\eta = \Theta(1/n)$, which alters the steady-state scaling of M_t and the inherited base-optimizer learning-rate rule. The normalized variant avoids these complications entirely, and all μP scalings of the base optimizer carry over verbatim.

R Generality Across Optimizer Families and Architectures on CIFAR-10

This appendix complements Section 4.3 with full setup details and the corresponding test-accuracy comparison. We compare AK-SGD with momentum against vanilla SGD with momentum on a 3-hidden-layer MLP (4096, 4096, 2048 units per each layer, respectively, with ReLU activations) trained on CIFAR-10 [15] for 120 epochs at batch size 256. We manually sweep the learning rate independently for each optimizer over ($\alpha \in \{5 \times 10^{-2}, 1 \times 10^{-1}, 3 \times 10^{-1}\}$ for SGD and $\alpha \in \{0.9, 1.3, 1.7, 2.1\}$ for AK-SGD) and report the best configuration. The basin of α for SGD was found in $\alpha = 0.1$ as achieving the minimal loss, whereas for AK-SGD, $\alpha = 1.7$ was the identified learning rate achieving minimal loss, and the momentum coefficient β is swept within range $\{0.9, 0.95, 0.99\}$ for both optimizers, where $\beta = 0.9$ was preferable for SGD and $\beta = 0.99$ for AK-SGD. For AK-SGD we additionally search $\eta \in \{0.1, 0.5, 0.9, 0.99\}$ and find the optimal point in $\eta = 0.99$. Figure 6 shows the training and validation trajectories, and Table 10 reports final test accuracy.



(a) Validation loss vs. epoch. The horizontal axis is the epoch index.



(b) Validation accuracy vs. epoch. The horizontal axis is the epoch index.

Figure 6: AK-SGD vs. SGD-momentum on CIFAR-10 with a 3-layer MLP. AK-SGD matches or exceeds SGD throughout training and achieves a higher final test accuracy, demonstrating that the delta-rule mechanism transfers beyond Adam's diagonal-preconditioner setting.

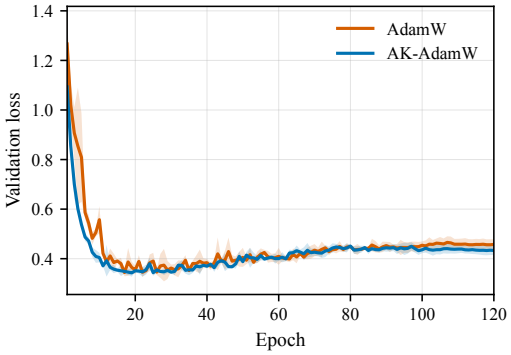
Table 10: Final test accuracy on CIFAR-10 (3-hidden-layer MLP, 120 epochs). Best per-optimizer learning rate selected from an independent sweep.

Optimizer	Final test acc. (%)	Δ vs. SGD
SGD-momentum	61.51	—
AK-SGD	67.06	+5.55

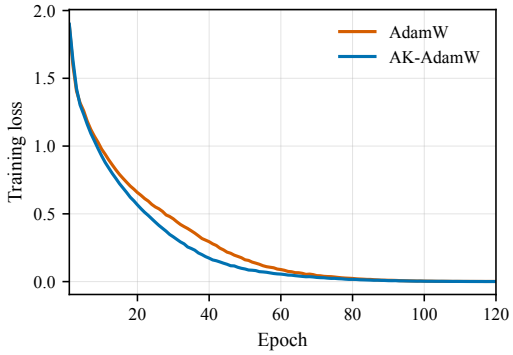
The qualitative pattern matches the language-modeling result, since the gain appears early, is sustained throughout training, and translates into a positive final-metric difference. This experiment is a controlled test of whether the delta-rule mechanism transfers across optimizer families, run on a 3-layer MLP for that purpose, and the answer is positive.

R.1 Standard Vision Architectures

This subsection complements Section 4.3 and Table 4. We train ResNet-18 and ViT-Tiny on CIFAR-10 for 120 epochs, AK-AdamW against AdamW, with 3 seeds per optimizer and an identical log-scale learning-rate sweep for each. All other settings are held at the values that the architecture's usual recipe prescribes and are the same across the two arms.



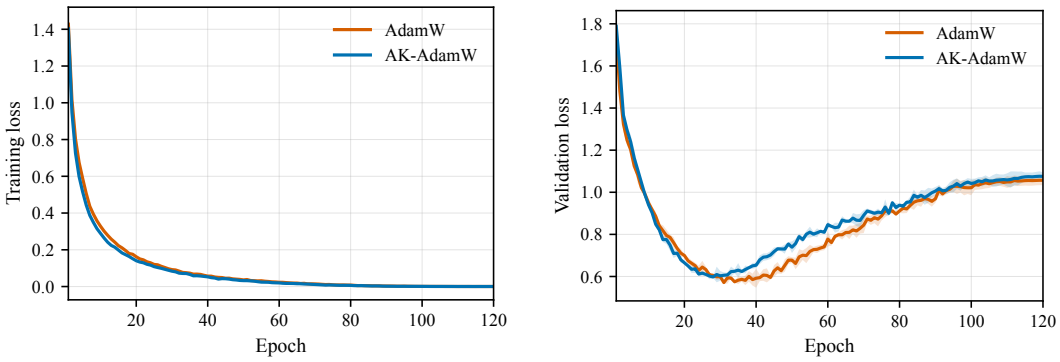
(a) ResNet-18, validation loss.



(b) ViT-Tiny, training loss.

Figure 7: AK-AdamW against AdamW on CIFAR-10 over 120 epochs, 3 seeds, seed means with per-seed bands. We read ResNet-18 on validation loss and ViT-Tiny on training loss, for the reason given below.

We read the two architectures on different curves, and the reason is what each curve can measure. ResNet-18 carries the locality and weight-sharing priors that CIFAR-10 rewards, so it generalizes well at this dataset size and its validation curve tracks optimization progress rather than a generalization gap. That makes validation loss both meaningful and the stricter of the two readings, so we use it. ViT-Tiny has no such inductive bias and is trained here from scratch without the large-scale pretraining or heavy augmentation that vision transformers normally rely on, so on a dataset of this size it fits the training set long before its validation loss reflects anything about the optimizer. Its validation curve is then dominated by the generalization gap of the architecture, which is a property of the model and not of the buffer rule, so we read ViT-Tiny on training loss, where the optimization question the paper asks is the one being measured. The complementary pair of curves is in Figure 8. The ResNet-18 training loss has the same ordering, and the ViT-Tiny validation loss of AK-AdamW is lower than that of AdamW only until about epoch 28.



(a) ResNet-18, training loss.

(b) ViT-Tiny, validation loss.

Figure 8: The complementary pair to Figure 7, reported for completeness. The ResNet-18 training loss has the same sign of the gap as in Figure 7. The ViT-Tiny validation loss of AK-AdamW is lower than that of AdamW until about epoch 28 and higher after about epoch 30. Both ViT-Tiny validation curves rise after their minima near epoch 30 while the training losses in Figure 7 keep decreasing. This rise is the generalization gap of the architecture on a dataset of this size rather than a difference between the optimizers.

We report the gap in training loss at matched epochs rather than a final metric, because both architectures saturate on CIFAR-10. Training accuracy reaches 99.99% and test accuracy sits near 84% for ViT-Tiny and 94% for ResNet-18 under either optimizer, so final performance cannot separate the two and convergence speed is the quantity that can be measured. We average over the phase in which the curves separate, which is roughly the first 60 epochs for ViT-Tiny and the first 20 for ResNet-18, and we report the gap both in nats and relative to the AdamW loss at the same epoch. The sign of the gap is the same in all runs. On validation loss the ordering holds for ResNet-18 over its averaging window, and for ViT-Tiny only until about epoch 28, after which both validation losses rise and that of AK-AdamW is higher. This supports that the mechanism is not specific to language modeling, while language-model pretraining remains where the claims of this paper are made.

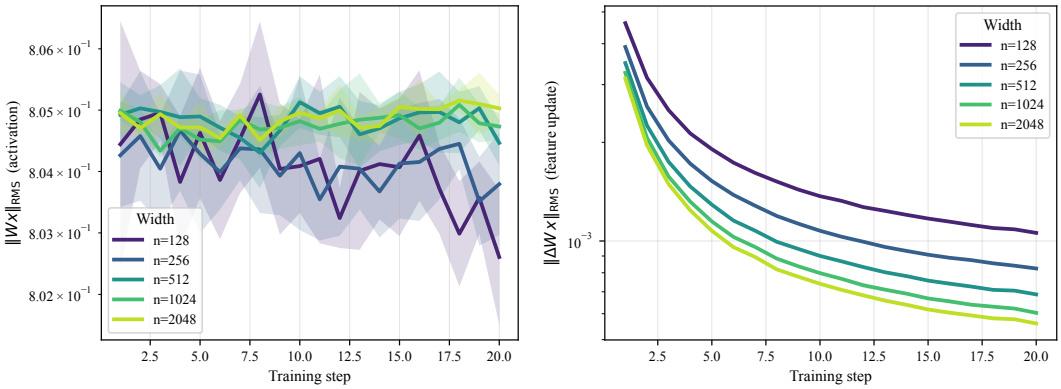
S μP Coordinate Check

Theorem Q.1 (Appendix Q) predicts that the delta-rule coefficient η is width-invariant under maximal-update parameterization. We verify this empirically with the standard μP coordinate-check protocol, training the same Llama-2-style architecture at five widths $d_{\text{model}} \in \{128, 256, 512, 1024, 2048\}$ for 20 optimization steps across five random seeds, holding all hyperparameters fixed except for the width-variant hyperparameters such as learning rate. In particular η and β_1 are transferred without rescaling, at $\eta = 0.5$ and $\beta_1 = 0.99$. The specific value of η is immaterial to this test, which asks only whether a single value transfers across widths, and it sits in the flat basin of Appendix U. We measured two per-coordinate scales whose width-invariance certifies μP-cleanness:

- **Activation RMS**: the root-mean-square per-coordinate magnitude of each layer's pre-activation, $\mathrm{RMS}(y_t) = (\frac{1}{m}\sum_i y_{t,i}^2)^{1/2}$, which should remain $\Theta(1)$ across widths.

- **Feature update RMS**: the per-coordinate magnitude of the change the realized weight update makes to the layer output, $\mathrm{RMS}(\Delta W x)$ with $\Delta W = W_{\text{after}} - W_{\text{before}}$, which should remain $\Theta(1)$ across widths under AK-AdamW's inherited μP learning-rate rule (Corollary Q.2).

Both quantities are measured in feature space, meaning projected onto the layer input, and that choice is deliberate. The parameter-space counterparts $\mathrm{RMS}(\Delta\theta)$ and $\mathrm{RMS}(\alpha\nabla W)$ are $\Theta(1/n)$ under correct μP and shrink with width, so a collapse test on them certifies nothing. The gradient is also the wrong proxy for the update here, since the second moment of Adam rescales it before it reaches the weights, so $\alpha\nabla W \neq \Delta W$ for the optimizers we compare. Taking the realized update and projecting it onto the input recovers the quantity the theory actually constrains. Under correct μP both scales are $\Theta(1)$ in width, so the curves for different widths should lie on one another, and a width-dependent trend in either one would open a spread that grows with n. We read the check by that collapse.



(a) Activation RMS vs. training step, by width.

(b) Feature update RMS vs. training step, by width.

Figure 9: μP coordinate check for AK-AdamW at widths $d_{\text{model}} \in \{128, 256, 512, 1024, 2048\}$ with η held fixed across widths. Activation RMS lies on one curve at every width. Feature-update RMS decreases with width by a factor that shrinks with each doubling, so the curves converge as width grows, which supports that η requires no width-dependent rescaling and that the inherited μP scaling rule applies unchanged.

The convergence of the per-coordinate-scale curves as width grows in Figure 9 supports the theoretical prediction that a single η tuned at one width transfers without modification to other widths. This is the practical content of Theorem Q.1.

T Hyperparameter Sweep

We use Optuna [2] with a TPE sampler and a SuccessiveHalvingPruner for hyperparameter selection on the 67M proxy. The search space, search budget, and selection criterion are summarized below.

Search space. We run separate Optuna studies for AdamW, AK-AdamW, and Muon with single seed (42). We report the final range of each axis, which is the range the reported optimum was selected inside. Ranges were refit whenever an optimum approached a boundary, and each prior optimum was re-enqueued as an anchor trial before sampling resumed. For AdamW, $\alpha \in [2{\times}10^{-3},\ 8{\times}10^{-3}]$ on log scale and $\beta_1 \in \{0.9, 0.95\}$. For AK-AdamW, $\alpha \in [5{\times}10^{-5},\ 4.5{\times}10^{-4}]$ on log scale, $\beta_1 \in \{0.95, 0.99\}$, and η TPE-sampled from $[0.05,\ 1.0]$ on log scale. For Muon, $\alpha \in [3{\times}10^{-3},\ 3{\times}10^{-2}]$ on log scale and $\beta_1 \in \{0.9, 0.95\}$. For AK-AdamW and for Muon we additionally search the auxiliary AdamW learning rate for the embedding and output-head group as its own axis rather than tying it to the main rate, since tying it starves those parameters, over $[2{\times}10^{-3},\ 4{\times}10^{-2}]$ for AK-AdamW and $[2{\times}10^{-3},\ 5{\times}10^{-2}]$ for Muon. All studies hold $\beta_2 = 0.99$ and weight decay 10^{-1} fixed. Each trial trains the 67M proxy for up to 4,000 steps, evaluating every 80 steps, with the cosine LR scheduler period set to the full 19,073-step horizon so that trial losses are comparable to early-training losses of the headline run. The SuccessiveHalvingPruner uses `min_resource` $= 600$, `reduction_factor` $=$ 3, and `min_early_stopping_rate` $= 1$, which places exactly one rung inside the trial budget, realized at step 1,840 of 4,000. Champions are selected from full-budget completions only and never

from rung values, because rung rank and final rank are close to uncorrelated among survivors in our studies.

Sweep budget. We count budget in full-trial equivalents, where one equivalent is one trial run to the full 4,000 steps, summing the steps every trial actually consumed. The AdamW studies ran 101 trials with 37 completions for 66.3 equivalents over 2 tuned dimensions, the AK-AdamW studies ran 207 trials with 72 completions for 129.2 equivalents over 4 dimensions, and the Muon studies ran 249 trials with 62 completions for 115.3 equivalents over 3 dimensions. Normalized by the number of tuned dimensions, the three arms received 33, 32, and 38 full-trial equivalents per dimension for AdamW, AK-AdamW, and Muon respectively. Muon therefore received the largest per-dimension budget of the three and our own method the smallest, which is the direction that matters for reading the comparison. Roughly 65 of the Muon trials are zero-cost prunes from a co-scheduling out-of-memory incident and consumed essentially no steps, leaving about 184 informative trials, and they are excluded from the equivalents above rather than counted as search effort. Most remaining trials are pruned, which is by design, since TPE concentrates samples near the basin and successive halving halts the rest at the rung. Total consumed compute is not equal across arms, since dimensionality and the number of refit rounds differ, so we claim identical protocol and per-dimension parity rather than equal GPU-hours. The Optuna SQLite study databases are released with the code.

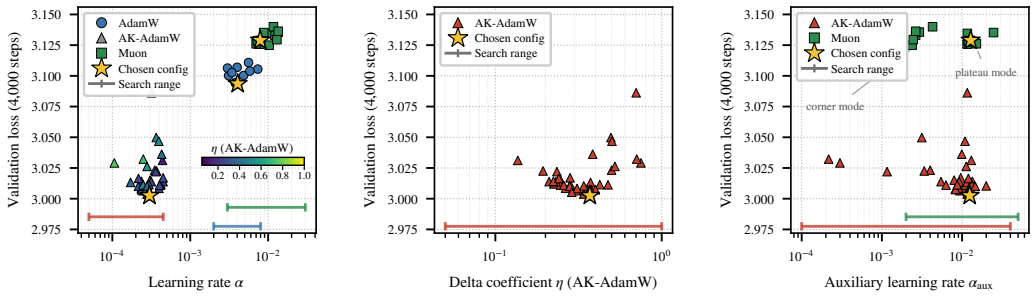
Selection criterion. The objective is the validation loss at the end of each trial. The configuration with the lowest validation loss is selected for the headline run, and no test-set information is used in selection.

Selected configurations. The selected hyperparameters at 67M and the corresponding μP-transferred values at the two targets are reported in Table 8. AdamW selects $\alpha = 4.0{\times}10^{-3}$ and $\beta_1 = 0.95$. AK-AdamW selects $\alpha = 3.0{\times}10^{-4}$, $\beta_1 = 0.99$, and $\eta = 0.37$. Every completed AK-AdamW trial used $\beta_1 = 0.99$, which is larger than AdamW prefers, and the recipe at the end of this appendix explains why the per-direction memory coefficient makes that the expected choice. The selected η sits inside the flat basin characterized in Appendix U, where the final loss moves by at most 0.015 nats across $\eta \in [0.10,\ 0.99]$, so neighbouring values in that basin do not change the headline configuration.

The AK-AdamW peak learning rate is more than ten times smaller than AdamW's, and the fixed point of the deployed rule, (8) with $\hat{G}$ replaced by $\bar{G}$ (Section 3.1), accounts for it. With the second moment unchanged, the fixed point of the buffer along u_i is $1/(\mu + \hat{\lambda}_i)$ times that of the EMA buffer, a factor between 0.97 and 37 at the selected $\beta_1 = 0.99$ and $\eta = 0.37$.

Every reported optimum is interior to its final range, with completed trials on both sides of it, so no reported value rests on a boundary. Writing the position of each optimum as a fraction of its final range on the log axis, where 0 is the lower bound and 1 the upper, AdamW α sits at 0.51, AK-AdamW $(\alpha, \alpha_{\text{aux}}, \eta)$ at $(0.82, 0.61, 0.67)$, and Muon $(\alpha, \alpha_{\text{aux}})$ at $(0.43, 0.58)$. The horizontal bars in Figure 10 draw these ranges. The categorical β_1 choices of AdamW and AK-AdamW both rest on completed trials rather than on defaults, including a controlled probe that runs one of the two optimizers at the β_1 value the other adopted.

Recommended tuning recipe. The marginal tuning cost over the EMA momentum is one coefficient. Based on the studies above, we recommend setting $\beta = 0.99$ and $\eta = 0.4$, keeping β_2 and weight decay at the AdamW settings, and setting the learning rate near one tenth of the tuned AdamW value and tuning only that. Three facts make this cost-efficient. First, $\beta = 0.99$ is larger than AdamW prefers, because the per-direction memory coefficient is $\tilde{\beta}_i$ rather than β, so every direction with positive input density forgets faster than nominal and a larger nominal β recovers the same effective horizon. Second, η is easy to select, since the final loss moves by at most 0.015 nats across $\eta \in [0.10, 0.99]$ (Appendix U) and η enters Theorem 3.4 only through the ridge μ, where the fixed point interpolates between the plain and the whitened gradient. Anything in $[0.3, 0.5]$ is a safe default. Normalized keys also remove any data-dependent constraint, since $\operatorname{tr} \hat{\Sigma} = 1$ places $\tilde{\beta}_i$ inside $[\beta - \eta, \beta]$ for any data, so $\eta \le \beta$ keeps every coefficient non-negative and stability needs only $\eta < 1 + \beta$. Third, the learning rate does not carry over from AdamW because the buffer is a Wiener predictor rather than an average of gradients. Both η and β are width-invariant under μP (Theorem Q.1) and the learning rate is transferred by the standard μP rule, so the whole recipe is tuned once at the smallest proxy.



(a) Trial loss vs. learning rate α. (b) Trial loss vs. delta coefficient η. (c) Trial loss vs. auxiliary rate α_{aux}.

Figure 10: Optuna trial losses on the 67M proxy, with the chosen configuration of each arm starred and the final range of each axis (Appendix T) drawn as a horizontal bar. (a) The three arms occupy well-separated learning-rate basins, and each optimum is interior to its own final range with completed trials on both sides. (b) The η surface is flat over most of a decade, which is the trial-level counterpart of the sensitivity ablation in Appendix U. (c) The auxiliary rate for the embedding and output-head group separates into two modes. The corner mode collects trials that pin the auxiliary rate near the low end of the interval and starves those parameters, and the plateau mode collects the trials that do not, which is why we tune this rate as its own axis rather than tying it to the main rate.

U η Sensitivity Ablation

Theorem 3.4 relates the regularization parameter $\mu = (1-\beta)/\eta$ to the fixed-point structure of the buffer, and this in turn predicts that final loss should depend smoothly on η, with $\eta \to \beta$ approaching the unregularized (pure delta-rule) Wiener fixed point. To verify this prediction and empirically characterize the role of η, we sweep η on the 67M proxy at the otherwise hyperparameters-fixed AK-AdamW configuration of Table 8. Each run uses batch size 256 and context length 2048 (524K tokens/step), trained for 5,440 steps ($\approx$ 2.85B tokens, $\approx 2.12\times$ Chinchilla optimal for this model size).

Table 11: Final validation loss on FineWeb-Edu (2.85BT, 67M proxy) as a function of the delta-rule coefficient η, with all other AK-AdamW hyperparameters fixed at the values in Table 8. The configuration selected by Optuna ($\eta = 0.37$) was chosen independently of this ablation and lies between the 0.30 and 0.50 grid points.

η	0.01	0.10	0.30	0.50	0.70	0.99
Final val. loss	3.063	3.029	3.034	3.035	3.035	3.020

The dependence on η is relatively flat for $\eta \geq 0.1$, with a wide basin that contains the selected value. The final loss varies by at most 0.015 nats across the wide interval $\eta \in [0.10, 0.99]$, and degrades smoothly at the limit ($\eta \approx 0$) while the pure-delta-rule limit ($\eta \to 1$) showed the best final validation loss by a small amount. Table 11 shows that $\eta = 0.99$ yields the lowest final loss in this $5{,}440$-step ablation, ahead of the 0.30 and 0.50 grid points that bracket the selected value by 0.014 and 0.015 nats and ahead of the next best point by 0.009 nats. Both margins are no larger than the 0.0150-nat seed standard deviation we measure on the 67M loss gap in Table 2, and this ablation is single-seed, so we do not interpret the $\eta = 0.99$ advantage as significant. This insensitivity is the practical correlate of the Tikhonov interpolation of Theorem 3.4, where a wide range of η produces buffers that interpolate between the EMA and Wiener targets without sharply changing the optimization behavior.

The ablation is run at the configuration of Table 8 with everything except η held fixed, so the sweep isolates η and any offset common to the row cancels in the spread. Its grid brackets the selected $\eta = 0.37$ between the 0.30 and 0.50 points, both of which sit in the flat basin. This is what makes the recipe of Appendix T safe to state as the interval $[0.3, 0.5]$ rather than as a point.

V Broader Impacts

AK-MOMENTUM is a general-purpose optimization method, not a deployed system. Its potential positive impact is improved training efficiency, which may reduce the compute cost of neural-network training and make experimentation more accessible. Its potential negative impact is indirect, since more efficient optimization could also lower the cost of training models for harmful or dual-use applications. The work does not release a high-risk model or dataset.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the method, its theoretical guarantees, its cost, and its empirical gains, and each claim matches the body of the paper. Section 3 and its appendices prove every theoretical claim, and Appendix H states which assumptions each result carries and which of them describe a training run. The overhead band in the abstract, 22.2% to 25.0% of a gated-MLP block's linear cost, comes from Theorem M.3, beside which Appendix M.6 reports the realized per-step overhead and the measured step time at all three scales. The quoted step reductions are the maxima over matched validation-loss levels in Table 2 over three seeds, with their seed standard deviation, and the matched-loss cost comparison of Table 3 covers 67M and 370M. The 1B run is a single seed, as Figure 1, Table 2, and Appendix N state, the Muon result is reported as a baseline tuned inside our protocol rather than a general claim about Muon (Appendix O), and the CIFAR-10 results serve as a generality check, with language-model pretraining as the setting of the claims (Section 4.3).

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the limitations and future work paragraph of Section 5. The experiments cover language-model pretraining up to 1B parameters and CIFAR-10 classifiers, so larger scales and domains such as image generation and reinforcement learning remain open, and our two modalities are trained separately rather than interleaved (Appendix J). The counted per-step overhead rises with width to 18.1% at 1B, and measured step time sits above the count at smaller widths, where the correction runs below the peak rate of the GPU (Appendix M.6), a gap that a kernel normalizing the keys as it loads them would narrow. Because Appendix M counts the overhead as a share of each step set by the layer shapes, its total grows with the training budget in proportion to the cost of training. Buffer-level compositions with Muon, Shampoo, and SOAP and a controlled comparison against a gradient-keyed delta rule on the buffer are left to future work, Muon is the only structured optimizer we compare against (Appendix O), and the 1B comparison is a single seed. Appendix H states which theoretical assumptions describe a training run and that the quasi-static condition of Theorem 3.4 is weakest early in training and under abrupt input shifts.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Each theoretical result states its assumptions in its statement or in the setup it follows, namely the quasi-static approximation and the spectral-radius condition of Theorem 3.4, the independently drawn keys of Lemma 3.7, the linear-regression setup and shared-eigenvector hypothesis of the per-direction convergence analysis with its underdamped band condition for the rate result, the shift to new fixed values of Theorem 3.10, and the bounded per-step drift of Proposition 3.11. Appendix H states which assumptions each result carries and which of them describe a training run. Section 3 includes the short proof of Lemma 3.7, on which the other results build, and the appendices give the complete proofs, covering the buffer as a gradient estimator (Appendix D), the estimation objectives and fixed points (Appendix E), the linear-regression dynamics (Appendix F), the tracking analysis (Appendix G), the extensions to high-dimensional regression, generalized linear models, and non-convex training (Appendix I), the stability of the normalized key (Appendix C), the FLOP count, overhead band, peak memory, and cost lower bound (Appendix M), and the maximal-update scaling (Appendix Q).

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: AK-MOMENTUM, its normalized-key form, and the AK-AdamW and AK-SGD instantiations are specified in Section 3, and Appendix L states the batch form of the

deployed rule that produces every number in Section 4. Appendix N gives the architectures (Table 7), data, tokenizer, validation split, training protocol, matched-level measurement window, baselines, μP transfer rules, compute, and seeds of the language-model runs. Appendix T gives the search space, pruner settings, budget, and selection rule of the hyperparameter search, with the selected values in Table 8 and a tuning recipe for the new coefficient, Appendix O gives the Muon protocol, Appendix R the CIFAR-10 settings, and Appendix M the FLOP and memory count together with the setup behind the measured step times.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is publicly available at `https://github.com/euijinh/AK-Momentum`. It contains the AK-MOMENTUM implementation of the deployed rule in Appendix L, the AK-AdamW and AK-SGD optimizers, the training and baseline scripts, the configuration files and commands for the FineWeb-Edu and CIFAR-10 experiments, environment instructions, and the Optuna study databases of Appendix T. Both datasets are public, the repository documents how to obtain and preprocess them, and Appendix N gives the full experimental settings.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 states the setting of each experiment to the level needed to read its results, and the appendices give the full details. Appendix N covers the architectures at all three scales, the data and tokenizer, the held-out 0.5% validation slice of FineWeb-Edu, the training protocol including the 20B-token budget of the 1B run, the matched-level window from step 2,000, the optimizer instantiations, and the μP transfer rules. Appendix T gives the search space, pruner, budget, and selection rule of the Optuna protocol that AdamW, AK-AdamW, and Muon share, with per-dimension budgets of 33, 32, and 38 full-trial equivalents, and Table 8 lists the selected values. Appendix O gives the Muon protocol, Appendix R the CIFAR-10 architectures and learning-rate sweeps, and Appendix U the setting of the η ablation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The FineWeb-Edu comparisons at 67M and 370M are each repeated over three seeds (42, 777, and 1234) for AdamW, AK-AdamW, and Muon, and Table 2 and Table 9 report every mean and maximum over matched validation-loss levels with its standard deviation across the three seeds, one standard deviation and not a standard error. Each seed sets the initialization and the data order, which AdamW and AK-AdamW share at a given seed, so the error bars capture variability across seeds, and the sign of the paired gap is the same in every run, with the 370M mean gap at 13.7 times its seed standard deviation (Section 4.1). Figure 1 plots seed means with per-seed bands, and on CIFAR-10 the ResNet-18 and ViT-Tiny comparison uses three seeds per optimizer under the same convention (Table 4). The 1B run, the hyperparameter studies, and the η ablation are single-seed, the AK-SGD comparison on the MLP is reported without error bars, and we make no distributional assumption and report no formal confidence intervals or significance tests.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix N states that all experiments run on H200 GPUs (141 GB of memory each) in a local cluster and gives the time of one run on a single H200 for AdamW and AK-AdamW at each scale, roughly 13 and 15 hours at 67M, 35 and 41 hours at 370M, and 112 and 132 hours at 1B, together with the seed count of every arm. Summed over seeds, the AdamW and AK-AdamW runs of Table 2 take about 560 H200 GPU-hours. Appendix T gives the cost of the hyperparameter search in full-trial equivalents, one equivalent being one 4,000-step trial of the 67M proxy, and Appendix M.6 gives the counted FLOP overhead, the measured step time, and its measurement setup at all three scales. The project also used compute for exploratory runs during method development that the paper does not report.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics, and the research conforms to it in every respect. The paper proposes and evaluates an optimizer on public benchmark datasets and involves no human subjects, crowdsourcing, private or sensitive personal data, or release of high-risk models or datasets.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in Appendix V. AK-MOMENTUM is a general-purpose optimization method rather than a deployed system, so its potential positive impact is more efficient and more accessible neural-network training, and its potential negative impact is indirect, since cheaper optimization could also lower the cost of training models for harmful or dual-use applications. The work releases no high-risk model or dataset.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper releases no data or model with a high risk of misuse, such as a pretrained language-model checkpoint, an image generator, or a scraped dataset. The release is optimizer and training code, and the experiments use existing public datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the creators of every existing asset we use and follow its terms. FineWeb-Edu [22] is released under the Open Data Commons Attribution License (ODC-By) v1.0 and is also subject to the Common Crawl terms of use, the Llama 2 tokenizer [30] is distributed under the Llama 2 Community License Agreement, and the creators of CIFAR-10 [15] ask users to cite its technical report, which we do. The software we build on is PyTorch [21], under a BSD-style license, and Optuna [2], under the MIT license, and the repository lists the version and source of each asset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset is the code release at `https://github.com/euijinh/AK-Momentum`, which contains the AK-MOMENTUM implementation, the AK-AdamW and AK-SGD optimizers, the training and baseline scripts, the configuration files, and the Optuna study databases. Its documentation covers the code structure, environment, reproduction commands, the license of the code, and how to obtain and preprocess the public datasets. We release no new dataset, trained model, or checkpoint.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper involves no crowdsourcing or research with human subjects. The experiments evaluate an optimizer on existing language-modeling and image-classification benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper involves no human subjects, participant recruitment, or collection of participant data, so there are no participant risks to disclose and no IRB or equivalent approval applies.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method does not involve LLMs as an important, original, or non-standard component. The paper proposes and analyzes an optimizer, and the only language models in it are the decoder-only transformers we pretrain to evaluate that optimizer.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.