AGAM: ALIGNMENT-GATED MOMENTUM FOR DEEP LEARNING OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Momentum-based optimizers underpin modern deep learning training, relying on accumulated past gradients to provide a reliable descent direction. However, this expectation is routinely violated in practice due to the non-stationarity of mini-batch gradients, causing a base optimizer's *proposed update direction* to become misaligned with the current gradient. We introduce **AGAM**, a state-preserving, drop-in gating mechanism for momentum-based optimizers, implemented as a **two-line** modification with minimal computational overhead and no additional hyperparameters. AGAM transiently attenuates the contribution of historical momentum via a tensor-wise, alignment-adaptive gating coefficient. We instantiate AGAM with SGDM, AdamW, and Lion, characterize their exact relationship to quasi-hyperbolic optimization, and establish a sharp descent certificate for AGAM-SGD. Empirical evaluations across ResNet image classification, MAE ViT pre-training, and LLM fine-tuning demonstrate that AGAM-augmented optimizers consistently outperform both their base optimizers and alignment-aware baselines.

1 INTRODUCTION

Momentum-based optimizers have become ubiquitous in deep learning training, offering a reliable way to accelerate convergence and improve generalization (Sutskever et al., 2013; Jelassi & Li, 2022). Virtually all optimizers used to train modern neural networks maintain this momentum as a running average of past gradients, most commonly an exponential moving average (EMA). Under a stationary mean and independent gradient noise, momentum reduces the variance of the gradient estimate, while also accumulating progress along low-curvature directions (Goh, 2017). It is especially important in the large-batch, long-horizon regimes typical of contemporary pre-training (Zhang et al., 2019).

However, successful training still requires carefully selecting and tuning an optimizer. With the rise of large language models (LLMs), many momentum-based optimizers have been proposed with increasing levels of sophistication over the canonical SGD with momentum (SGDM), Adam and AdamW. Each optimizer typically introduces its unique hyperparameter(s) or operates robustly only within a specific hyperparameter range of its own, requiring method-specific tuning and discovery (Wen et al., 2025). This greatly hinders their ease of use and adoption by practitioners.

Moreover, nearly all momentum-based optimizers use a fixed momentum coefficient throughout training, typically $\beta = 0.9$. This keeps the effective historical horizon constant across tensors and training phases. There exists a tradeoff between selecting larger β values to enhance variance reduction and maintaining optimizer responsiveness to non-convex loss landscape dynamics. Increasing β decreases the relative importance of recent gradients, delaying the updates' adaptation to local changes in the landscape. The local geometry changes due to curvature shifts between regions (Pascanu et al., 2013), learning-rate phase transitions, or heavy-tailed gradient outliers. Stochastic mini-batch gradients $\boldsymbol{g}_t$ are therefore non-stationary along the optimization trajectory. Since each $\boldsymbol{g}_t$ describes the loss's local first-order behavior, past gradients become less representative of the current geometry, and the information accumulated in the momentum $\boldsymbol{m}_t$ grows stale as training progresses. That staleness is most acute when momentum is most needed, in the non-stationary, high-noise regimes of transformer pre-training and self-supervised learning. Together with mini-batch noise, this can cause the current $\boldsymbol{g}_t$ to disagree with the momentum-based update direction $\boldsymbol{u}_t$. We refer to this directional disagreement as *update-gradient misalignment* (or *momentum-gradient misalignment* when compared against $\boldsymbol{m}_t$). In particular, when $\langle \boldsymbol{g}_t, \boldsymbol{u}_t \rangle < 0$, the proposed step $-\eta_t \boldsymbol{u}_t$ has a positive first-order directional

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

derivative for the current mini-batch loss. This reasoning supports update-gradient alignment as a principled basis for identifying and adjusting destabilizing updates. However, disagreement with a single noisy gradient does not necessarily imply that the accumulated momentum has become uninformative and worth discarding. This motivates adapting how the historical momentum contributes to the current update while preserving its accumulated history.

Currently, the dominant misalignment mitigations are *destructive*. Momentum restarts, whether triggered adaptively or on a schedule, erase the buffer and discard the accumulated gradient history (O'Donoghue & Candès, 2015; Wang et al., 2022). Masking schemes zero out the coordinates whose update disagrees in sign with the current gradient (Liang et al., 2025), or drop entire parameter blocks at random and rescale the survivors by their alignment (Joo et al., 2026), discarding the masked part of the update. Methods that intervene at the momentum recurrence, such as Torque-Aware Momentum (TAM; Malviya et al., 2024) and Adaptive Memory (AM; Topollai & Choromanska, 2025), write their corrections into the momentum buffer, and TAM, like AM in its global form, computes a single *model-wide* correction. A model-wide cosine obscures the heterogeneity across tensors and layers. Indeed, we measure misalignment rates that differ by more than an order of magnitude across parameter groups and block depths at the same step, and a model-wide gate computed from the same gradients misses most of these conflicts. Empirically, it tracks the convolutional weights and ignores the normalization parameters on ResNet-50, and on a ViT, it never reports misalignment after epoch 21 of 300 while individual tensors still misalign (Appendix E, Figures 17 and 22).

Our approach. We argue that update-gradient misalignment should change how much of the historical momentum is *spent* on the current step, rather than how much is *kept*. We propose **Alignment-Gated Momentum (AGAM)**, which gates the contribution of historical momentum by the alignment between $\boldsymbol{g}_t$ and $\boldsymbol{u}_t$. When the two directions agree, the gate approaches one and AGAM approaches the base optimizer, recovering it exactly at full alignment. The more they conflict, the further the gate closes and the more the step follows the current gradient, acting as a soft restart that deletes no state. Unlike masking, AGAM rescales the contribution of momentum continuously instead of zeroing coordinates of the update. AGAM differs from TAM and AM on three main axes. *(i) Transient:* the gated momentum is used for the current step exclusively, while the stored buffer and other states advance exactly as in the base optimizer, so a single spurious mini-batch cannot erase the accumulated history. *(ii) Tensor-wise:* the alignment-adaptive gate $\gamma_t^{(k)}$ is derived for each tensor at every step, instead of a single model-wide correction. *(iii) Update-based:* alignment is measured against $\boldsymbol{u}_t^{(k)}$ which already has $\boldsymbol{g}_t^{(k)}$ folded in, rather than against the stored buffer $\boldsymbol{m}_{t-1}$. This makes $\gamma_t^{(k)}$ sensitive to relative norms and, in a Gaussian noise model, centers it at a strictly positive value set by β alone.

Our Key Contributions:

- **Methodological:** AGAM is a simple drop-in enhancement for momentum-based optimizers that improves training stability and generalization with no additional hyperparameters and negligible overhead. The AGAM-SGD and AGAM-Adam updates are convex combinations of the base update and a memoryless direction (AGAM-Lion forms the same mixture before its sign), which places AGAM in the quasi-hyperbolic optimization family with an adaptive, tensor-wise immediate-discount factor (Ma & Yarats, 2019).
- **Theoretical:** We establish a sharp descent certificate for the ideal AGAM-SGD gate: the gated step can oppose the gradient only if the proposed update is more than $3 + 2\sqrt{2} \approx 5.83$ times longer than it. We derive a high-dimensional Gaussian gate reference and prove a stationarity bound for AGAM-SGD and AGAM-Adam under explicit oracle and preconditioner assumptions.
- **Empirical:** AGAM-augmented optimizers consistently outperform both their base optimizers and the alignment-aware baselines on every task, using the base's tuning range, and substantially improve the downstream fine-tuning accuracy of MAE-pretrained ViTs.
- **Diagnostic:** We provide a per-tensor telemetry analysis of update-gradient misalignment for AGAM-SGD and AGAM-AdamW, showing that misalignment varies substantially with the learning-rate schedule, parameter group, and block depth, which prior alignment-aware methods report at most as a single model-wide statistic.

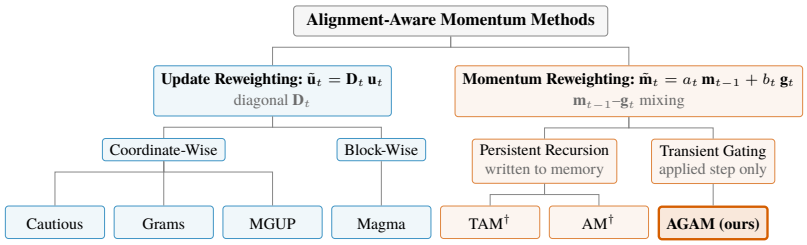


Figure 1: A taxonomy of alignment-aware momentum methods, classified by the intervention stage at which the alignment signal acts. **Update reweighting** applies a diagonal matrix $\mathbf{D}_t$ that masks or rescales the base optimizer's update $\mathbf{u}_t$ and leaves the optimizer state untouched. **Momentum reweighting** changes how the memory $\mathbf{m}_{t-1}$ and the mini-batch gradient $\mathbf{g}_t$ are combined, either inside the stored recurrence or only in the applied step. Methods within a branch are alternative intervention mechanisms addressing the same design question. †Benchmarked in our experiments.

2 Related Work

2.1 Momentum and Adaptive Methods

Heavy-ball momentum (Polyak, 1964) accumulates $\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \boldsymbol{g}_t$ from mini-batch gradients $\boldsymbol{g}_t = \nabla \ell(\boldsymbol{\theta}_t; \xi_t)$, and Nesterov acceleration (Nesterov, 1983) changes where momentum enters the gradient evaluation. Adaptive methods (Duchi et al., 2011; Tieleman & Hinton, 2012; Kingma & Ba, 2015) rescale each coordinate by a second-moment estimate, and AdamW (Loshchilov & Hutter, 2019) decouples weight decay. Stochastic momentum is well understood (Yan et al., 2018; Liu et al., 2020b; Gitman et al., 2019), whereas Adam can fail to converge under otherwise standard assumptions (Reddi et al., 2018), which is why our stationarity bound for AGAM-Adam (Appendix D) assumes explicit spectral bounds on the preconditioner. In all of these methods β (or β_1) is a constant, so the averaging horizon $\approx 1/(1-\beta)$ is the same for every parameter and every phase of training. AGAM keeps this stored horizon and adapts how much of the memory enters the applied step.

2.2 Quasi-Hyperbolic Momentum

Ma & Yarats (2019) decouple the moving-average recurrence from the immediate gradient–moment mixture, proposing $\boldsymbol{u}_t = (1-\nu)\boldsymbol{g}_t + \nu \boldsymbol{z}_t$ with the EMA $\boldsymbol{z}_t = \beta \boldsymbol{z}_{t-1} + (1-\beta)\boldsymbol{g}_t$ for QHM and its Adam counterpart QHAdam. QHM recovers plain SGD ($\nu = 0$), momentum ($\nu = 1$), and Nesterov momentum ($\nu = \beta$), and QHAdam recovers Adam ($\nu_1 = \nu_2 = 1$). In both forms, ν is a *fixed* hyperparameter shared by every parameter at every step, and the analysis of Gitman et al. (2019) covers prescribed, non-random coefficients. AGAM-Adam has exactly the QHAdam form with $\nu_2 = 1$ and AGAM-SGD the QHM form up to a gate-dependent step scale (Proposition D.1, Remark D.2), with the constant replaced by a time-dependent, per-tensor alignment gate. Lion (Chen et al., 2023) is itself quasi-hyperbolic before its sign, with $\nu = \beta_1/\beta_2$ (Section 3.4).

2.3 Alignment-Aware Methods

Figure 1 organizes methods that use gradient-alignment as a control signal. Methods within a branch operate at the same intervention stage and are alternative answers to the same design question; hence, they are largely mutually exclusive. Methods in different branches are complementary and can in principle be composed, for example by applying an update-reweighting rule $\mathbf{D}_t$ to AGAM's step.

Momentum reweighting. Torque-Aware Momentum (Malviya et al., 2024) computes a model-wide cosine between $\boldsymbol{g}_t$ and the *stale* buffer $\boldsymbol{m}_{t-1}$, smooths it with an EMA, and uses it to damp the incoming gradient. Adaptive Memory (Topollai & Choromanska, 2025) derives a memory coefficient from a model-based surrogate and modulates both terms. It is model-wide in its basic form, and its AdamW formulation also supports per-layer coefficients. Both introduce method-specific hyperparameters (a smoothing rate and a gradient floor for TAM, a proximal weight and a momentum cap for AM), and both write the modulation into the persistent state. In the terms of

Algorithm 1 AGAM (optimizer-agnostic). Two-line modification. PyTorch versions in Appendix L.

Require: parameter tensors $\{\boldsymbol{\theta}^{(k)}\}_{k=1}^{K}$, momentum map $\mathcal{M}$ and update map $\mathcal{U}$ of the base momentum-based optimizer, learning rate η_t, momentum coefficient β, remaining base-optimizer state Ψ_t if any (e.g. Adam's second moment)

initialize $\boldsymbol{m}_0^{(k)} \leftarrow \mathbf{0}$ for all k
for $t = 1, 2, \ldots, T$ **do**
  $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t)$
  for $k = 1, 2, \ldots, K$ **do**
    $\boldsymbol{m}_t^{(k)} \leftarrow \mathcal{M}\Big(\beta, \boldsymbol{m}_{t-1}^{(k)}, \boldsymbol{g}_t^{(k)}\Big)$ ▷ buffer advances exactly as in the base optimizer
    $\boldsymbol{u}_t^{(k)} \leftarrow \mathcal{U}\Big(\boldsymbol{m}_t^{(k)}, \Psi_t^{(k)}\Big)$
    $\gamma_t^{(k)} \leftarrow \big(1 + \mathrm{cossim}(\boldsymbol{g}_t^{(k)}, \boldsymbol{u}_t^{(k)})\big)/2$ ▷ tensor-wise alignment gate
    $\boldsymbol{m}_{\mathrm{eff},t}^{(k)} \leftarrow \mathcal{M}\Big(\gamma_t^{(k)}\beta, \boldsymbol{m}_{t-1}^{(k)}, \boldsymbol{g}_t^{(k)}\Big)$ ▷ gated momentum ($\beta_{\mathrm{eff},t}^{(k)} := \gamma_t^{(k)}\,\beta$)
    $\boldsymbol{u}_{\mathrm{eff},t}^{(k)} \leftarrow \mathcal{U}\Big(\boldsymbol{m}_{\mathrm{eff},t}^{(k)}, \Psi_t^{(k)}\Big)$ ▷ AGAM update direction
    $\boldsymbol{\theta}_{t+1}^{(k)} \leftarrow \boldsymbol{\theta}_t^{(k)} - \eta_t \boldsymbol{u}_{\mathrm{eff},t}^{(k)}$
  end for
end for

Figure 1, they are persistent momentum reweighting. AGAM belongs to the same branch but differs in its transient intervention and its update-based alignment signal, as Appendix C examines in detail.

Update reweighting. Cautious optimizers (Liang et al., 2025) zero the coordinates whose proposed update disagrees in sign with the gradient and rescale the rest. MGUP (Chang & Yuan, 2025) gives a larger step to the fraction of coordinates with the largest update–gradient product and a smaller, non-zero step to the rest. Grams (Cao et al., 2024) pairs the gradient's sign with the momentum's magnitude, and Magma (Joo et al., 2026) randomly masks whole parameter blocks and scales the surviving updates by a smoothed momentum–gradient alignment score. These rules act through signs, rankings, or random masks, and their analyses must handle the resulting discontinuities. MGUP requires a positive step for unselected coordinates by contrast, AGAM's gate is continuous.

Choice of baselines. We compare AGAM with TAM and AM because all three answer the same question: how much of the stored momentum should contribute to the current step. They are practically mutually exclusive and one can only pick one of them. Update-reweighting methods act one stage later on the update the base optimizer has already formed. Thus, they could technically be applied on top of AGAM rather than in its place. Therefore, our experiments compare against the base optimizer that AGAM augments and against TAM and AM, just as MGUP and Magma compare against Cautious. Since TAM and AM also differ from AGAM in their statistic and state handling, Appendix N ablates AGAM's design choices one at a time.

2.4 Momentum restarts

Momentum restarts are the most direct precedent for reacting to misalignment. O'Donoghue & Candès (2015) restart accelerated gradient methods whenever the objective increases or the momentum step makes an obtuse angle with the negative gradient, and scheduled restarts reset the buffer at fixed intervals (Wang et al., 2022). Both erase the accumulated history, whereas a closed AGAM gate suppresses memory for a single step and leaves the buffer intact (further related work in Appendix A).

3 Method: Alignment-Gated Momentum

We now present AGAM (Algorithm 1) and its instantiations for SGDM, AdamW, and Lion. For each parameter tensor, AGAM derives a gate from the cosine similarity between the current gradient and the base optimizer's *proposed update*, and uses it to attenuate the momentum coefficient for the current step only, preserving the base momentum recursion.

3.1 Optimizer-Agnostic Gating

Let $\boldsymbol{u}_t^{(k)}$ and $\boldsymbol{g}_t^{(k)}$ denote respectively the base optimizer's proposed update direction and current gradient for parameter tensor k at step t, and let $s_t^{(k)}$ be the cosine similarity measure between them. AGAM *attenuates* the base optimizer's constant momentum coefficient β via a tensor-wise, step-wise gate $\gamma_t^{(k)} \in [0, 1]$, obtained by passing $s_t^{(k)}$ through the bounded, monotone map $\phi : [-1, 1] \to [0, 1]$:

$$\gamma_t^{(k)} = \phi(s_t^{(k)}) = \frac{1 + s_t^{(k)}}{2} = \cos^2\left(\frac{\theta(\boldsymbol{g}_t^{(k)}, \boldsymbol{u}_t^{(k)})}{2}\right). \tag{1}$$

The gated buffer is used for the current step and never stored (*transient* gating). Equation 1 serves as a suitable alignment control signal due to its invariance to positive rescaling of either argument (e.g., loss scaling) and independence on the learning rate, and is directly computable, cheaply, from existing optimizer state. Practically, AGAM is a **drop-in**, **two-line** modification with no new hyperparameters that inherits its base optimizer's states. Below we omit the tensor index (k) when no ambiguity arises.

3.2 AGAM-SGD

AGAM-SGD takes the momentum map $\mathcal{M}$ of Algorithm 1 to be the heavy-ball recursion and the update map $\mathcal{U}$ to be the identity:

$$\boldsymbol{m}_t = \mathcal{M}(\beta, \boldsymbol{m}_{t-1}, \boldsymbol{g}_t) = \beta\, \boldsymbol{m}_{t-1} + \boldsymbol{g}_t, \qquad \boldsymbol{u}_t^{\text{SGDM}} = \mathcal{U}(\boldsymbol{m}_t) = \mathrm{Id}(\boldsymbol{m}_t) = \boldsymbol{m}_t.$$

The *effective* momentum applied by AGAM-SGD is then:

$$\boldsymbol{m}_{\text{eff},t} = \mathcal{M}(\gamma_t\, \beta, \boldsymbol{m}_{t-1}, \boldsymbol{g}_t) = \gamma_t\, \beta\, \boldsymbol{m}_{t-1} + \boldsymbol{g}_t. \tag{2}$$

Since $\mathcal{U}$ is the identity, $\boldsymbol{m}_{\text{eff},t}$ is the applied update $\boldsymbol{u}_t^{\text{AGAM-SGD}}$ itself, which satisfies:

$$\boldsymbol{u}_t^{\text{AGAM-SGD}} = \gamma_t\, \boldsymbol{u}_t^{\text{SGDM}} + (1 - \gamma_t)\boldsymbol{u}_t^{\text{SGD}}, \tag{3}$$

where $\boldsymbol{u}_t^{\text{SGD}} := \boldsymbol{g}_t$ denotes the standard *memoryless* SGD update direction. The AGAM-SGD update is thus an adaptive convex combination of the SGDM update and the memoryless SGD update: perfect alignment ($\gamma_t = 1$) recovers $\boldsymbol{u}_t^{\text{SGDM}}$ exactly, and antiparallel directions ($\gamma_t = 0$) give $\boldsymbol{u}_t^{\text{SGD}}$, which ignores the conflicting momentum for that step. Similarly, the EMA version of SGDM with $\mathcal{M}(\beta, \boldsymbol{m}_{t-1}, \boldsymbol{g}_t) = \beta\, \boldsymbol{m}_{t-1} + (1 - \beta)\boldsymbol{g}_t$ also yields the identity in equation 3 unchanged.

3.3 AGAM-Adam

Following Algorithm 1, AGAM-Adam is derived by defining:

$$\boldsymbol{m}_t = \mathcal{M}(\beta_1, \boldsymbol{m}_{t-1}, \boldsymbol{g}_t) = \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t, \qquad \boldsymbol{u}_t^{\text{Adam}} = \mathcal{U}(\boldsymbol{m}_t, \Psi_t) = \mathbf{D}_t^{-1}\hat{\boldsymbol{m}}_t,$$

where $\hat{\boldsymbol{m}}_t = \boldsymbol{m}_t/(1 - \beta_1^t)$ and $\mathbf{D}_t = \mathrm{diag}(\sqrt{\hat{\boldsymbol{v}}_t}) + \epsilon\, \mathbf{I}$ is Adam's diagonal preconditioner, with the unbiased second moment $\hat{\boldsymbol{v}}_t = \boldsymbol{v}_t/(1 - \beta_2^t)$ normalized exactly as in Adam. The gate modulates both the retained momentum and the current-gradient contribution, yielding the effective first moment:

$$\boldsymbol{m}_{\text{eff},t} = \mathcal{M}(\gamma_t\, \beta_1, \boldsymbol{m}_{t-1}, \boldsymbol{g}_t) = \gamma_t \beta_1 \boldsymbol{m}_{t-1} + (1 - \gamma_t \beta_1)\boldsymbol{g}_t. \tag{4}$$

Because $\boldsymbol{m}_{t-1}$ is zero-initialized, its gradient weights in $\boldsymbol{m}_{\text{eff},t}$ sum to $\gamma_t\beta_1(1-\beta_1^{t-1})+(1-\gamma_t\beta_1) = 1 - \gamma_t\beta_1^t$. Adam's bias correction divides a moment by the total weight it places on past gradients, which is $1 - \beta_1^t$ for $\boldsymbol{m}_t$. Hence, to get the effective first-moment estimate and the AGAM-Adam update, we normalize by the realized mass:

$$\hat{\boldsymbol{m}}_{\text{eff},t} = \frac{\boldsymbol{m}_{\text{eff},t}}{1 - \gamma_t\, \beta_1^t}, \tag{5}$$

$$\boldsymbol{u}_t^{\text{AGAM-Adam}} = \mathbf{D}_t^{-1}\hat{\boldsymbol{m}}_{\text{eff},t}. \tag{6}$$

Writing $\boldsymbol{m}_{\text{eff},t} = \gamma_t \boldsymbol{m}_t + (1 - \gamma_t)\boldsymbol{g}_t$ and $\boldsymbol{m}_t = (1 - \beta_1^t)\hat{\boldsymbol{m}}_t$ (Appendix F.1) turns Equation 5 into

$$\hat{\boldsymbol{m}}_{\text{eff},t} = \nu_t\, \hat{\boldsymbol{m}}_t + (1 - \nu_t)\boldsymbol{g}_t, \tag{7}$$

where

$$\nu_t^{(k)} := \frac{\gamma_t^{(k)}(1-\beta_1^t)}{1-\gamma_t^{(k)}\beta_1^t}, \qquad \nu_t^{(k)} - \gamma_t^{(k)} \longrightarrow 0, \qquad \nu_t^{(k)} \in [0, \gamma_t^{(k)}] \subseteq [0,1], \tag{8}$$

and therefore:

$$\boldsymbol{u}_t^{\text{AGAM-Adam}} = \nu_t\, \boldsymbol{u}_t^{\text{Adam}} + (1-\nu_t)\, \mathbf{D}_t^{-1}\, \boldsymbol{u}_t^{\text{SGD}}. \tag{9}$$

AGAM-Adam's update is thus an adaptive convex combination of the Adam update and the preconditioned memoryless update $\mathbf{D}_t^{-1}\boldsymbol{u}_t^{\text{SGD}}$, recovering $\boldsymbol{u}_t^{\text{Adam}}$ upon exact alignment. Equation 9 is *exactly* the QHAdam form (with $\nu_2 = 1$), with the fixed ν_1 replaced by the adaptive, tensor-wise immediate discount $\nu_t^{(k)}$, whose difference from $\gamma_t^{(k)}$ vanishes with the initialization correction. Normalizing the realized mass is an algebraic correction, not a claim that the gated estimate is statistically unbiased. Likewise, AGAM-SGD with the EMA buffer has the QHM form with the fixed ν replaced by $\gamma_t^{(k)}$. The same holds with the heavy-ball buffer up to a gate-dependent rescaling of the step (Remark D.2). AGAM is therefore an adaptive generalization of QHM's interpolation structure.

All Adam-based experiments in Section 4 use the decoupled weight-decay version AGAM-AdamW: $\boldsymbol{u}_t^{\text{AGAM-AdamW}} = \boldsymbol{u}_t^{\text{AGAM-Adam}} + \lambda\boldsymbol{\theta}_t$. As the shrinkage leaves both moments, $\boldsymbol{g}_t$, γ_t, and $\boldsymbol{u}_t^{\text{AGAM-Adam}}$ unchanged, the identities above apply to $\boldsymbol{u}_t^{\text{AGAM-AdamW}}$ with the shrinkage added separately.

3.4 AGAM-Lion

Lion (Chen et al., 2023) applies the sign of an immediate mixture of its stored EMA and the current gradient, and advances the EMA with a second, slower coefficient:

$$\boldsymbol{u}_t^{\text{Lion}} = \operatorname{sign}\big(\beta_1\boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t\big), \qquad \boldsymbol{m}_t = \beta_2\boldsymbol{m}_{t-1} + (1-\beta_2)\boldsymbol{g}_t,$$

with $0 \le \beta_1 \le \beta_2 < 1$ (0.9 and 0.99 by default). In Algorithm 1, Lion's gated map $\mathcal{M}$ corresponds to the immediate mixture and $\mathcal{U} = \operatorname{sign}$, except that its stored EMA advances with β_2 instead of through $\mathcal{M}$. AGAM-Lion computes the gate against Lion's proposed signed update $\boldsymbol{u}_t^{\text{Lion}}$ and gates only the immediate coefficient β_1:

$$\boldsymbol{u}_t^{\text{AGAM-Lion}} = \operatorname{sign}\big(\gamma_t\beta_1\boldsymbol{m}_{t-1} + (1-\gamma_t\beta_1)\boldsymbol{g}_t\big). \tag{10}$$

The persistent state $\boldsymbol{m}_t$ advances exactly as in Lion. Substituting $\boldsymbol{m}_{t-1} = \big(\boldsymbol{m}_t - (1-\beta_2)\boldsymbol{g}_t\big)/\beta_2$ reveals that the *argument* of the sign is an adaptive quasi-hyperbolic mixture:

$$\gamma_t\beta_1\boldsymbol{m}_{t-1} + (1-\gamma_t\beta_1)\boldsymbol{g}_t = \nu_t\,\boldsymbol{m}_t + (1-\nu_t)\,\boldsymbol{u}_t^{\text{SGD}}, \qquad \nu_t = \gamma_t\,\frac{\beta_1}{\beta_2} \in \Big[0, \frac{\beta_1}{\beta_2}\Big], \tag{11}$$

for $0 < \beta_2 < 1$ and $0 \le \beta_1 \le \beta_2$, as with Lion's defaults. Lion itself is the case $\gamma_t = 1$, a quasi-hyperbolic mixture with the fixed discount $\nu = \beta_1/\beta_2$ before the sign, and a closed gate ($\gamma_t = 0$) yields signSGD (Bernstein et al., 2018). Because the sign is applied after the mixing, the applied step is not a convex combination of two update directions, so the descent certificate of Lemma D.3 does not apply to AGAM-Lion. Mixing toward the gradient can nevertheless never turn an agreeing coordinate sign into a disagreeing one (Proposition D.4).

3.5 Design Rationale

Three choices distinguish AGAM from prior methods. Appendix B further discusses each in detail.

Transient gating. AGAM leaves the base recursion $\boldsymbol{m}_t = \mathcal{M}(\beta, \boldsymbol{m}_{t-1}, \boldsymbol{g}_t)$ *untouched* and alters only the applied step. Because the gate depends on a single mini-batch gradient, storing the gated momentum would (potentially) allow one spurious measurement to permanently attenuate the history that momentum needs in ill-conditioned ravines (Goh, 2017). The proof of Theorem D.6 controls the gate's indirect effect on future momentum through the tracking error of the stored moment.

Tensor-wise gating. One alignment gate $\gamma_t^{(k)}$ computed independently for each parameter tensor k at every step enables tensors to retain different effective coefficients $\beta_{\text{eff},t}^{(k)} = \gamma_t^{(k)}\beta$. A model-wide cosine is dominated by the tensors with the largest gradient and update norms, so agreement in some can mask disagreement in others (Appendix E).

Alignment with the proposed update. The first-order change of the mini-batch loss along $-\boldsymbol{u}_t$ is $\propto -\langle \boldsymbol{g}_t, \boldsymbol{u}_t \rangle$, so gradient–update alignment decides whether a step ascends, whereas disagreement between $\boldsymbol{m}_{t-1}$ and $\boldsymbol{g}_t$ does not. Folding $\boldsymbol{g}_t$ into the probe also makes the gate *norm-sensitive*: for heavy ball, $\gamma_t < \frac{1}{2}$ exactly when:

$$\begin{aligned} \gamma_t < \tfrac{1}{2} &\iff \beta \langle \boldsymbol{g}_t, \boldsymbol{m}_{t-1} \rangle < -\|\boldsymbol{g}_t\|^2 \\ &\iff \beta \underbrace{\frac{\|\boldsymbol{m}_{t-1}\|}{\|\boldsymbol{g}_t\|}}_{\text{relative norm}} \underbrace{\operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{m}_{t-1})}_{\text{angular alignment}} < -1, \end{aligned} \tag{12}$$

so misalignment requires both angular disagreement and a large relative momentum norm. Under independent zero-mean Gaussian noise, the stale-buffer $\boldsymbol{m}_{t-1}$ cosine reaches zero in expectation, whereas AGAM's cosine concentrates around a strictly positive value set by β in high dimension (Proposition D.5). If gradient norms fall faster than the stored buffer adapts (approaching convergence), the norm condition becomes easier to meet. Learning-rate decay alone does not guarantee this ordering, but misalignment does rise again late in training under cosine decay and not at a constant learning rate as we show empirically in Figure 11 and Figure 12.

4 Empirical Evaluation

To empirically verify the effectiveness of AGAM as a drop-in enhancement, we ask four questions. **Q1:** Does AGAM improve on its base optimizer and on the alignment-aware methods closest to it? **Q2:** Is it more robust across learning rates and batch sizes? **Q3:** Does it reach a target accuracy in fewer epochs under? **Q4:** Does it transfer across base optimizers and training modalities?

Setup. *(i) Image classification:* ResNet-18 on CIFAR-10 and ResNet-50 on CIFAR-100 (He et al., 2016; Krizhevsky, 2009), trained from scratch for 200 epochs over full grids of doubling learning rates and batch sizes, $\eta \in \{0.025, \ldots, 0.8\} \times B \in \{64, \ldots, 4096\}$ (6×7) and $\eta \in \{0.025, \ldots, 0.8\} \times B \in \{128, \ldots, 4096\}$ (6×6). *(ii) MAE pre-training:* a masked autoencoder (He et al., 2022) with a ViT-Tiny encoder (Dosovitskiy et al., 2021), pre-trained for 300 epochs on 64×64 Tiny-ImageNet (Le & Yang, 2015) at batch size 1024. *(iii) LLM fine-tuning:* full-parameter fine-tuning of SmolLM2-135M (Ben Allal et al., 2025) on WikiText-2 (Merity et al., 2017) for 5 epochs, and LoRA (Hu et al., 2022) fine-tuning of Qwen2.5-3B (Qwen Team, 2024) on GSM8K (Cobbe et al., 2021) for 3 epochs. The base optimizer is SGDM for the ResNets and AdamW (Loshchilov & Hutter, 2019) or Lion for the transformers. The baselines are TAM (TAM, AdaTAMW) and AM (AM-MSGD, AM-AdamW), the two published methods in AGAM's branch of the taxonomy (Figure 1). Neither has a Lion variant.

Protocol. Every method sees the same grid, schedule, budget, and three seeds. Optimizer families work in different hyperparameter regimes: Lion needs a much smaller learning rate and a larger weight decay than AdamW (Chen et al., 2023), so each family has its own grid. TAM and AM have their extra hyperparameters at the published defaults. AGAM is never tuned separately as it merely inherits its base optimizer's momentum coefficient(s) and hyperparameter grid. On the classification grids, we report the final-epoch test accuracy and a *validation-selected* test accuracy, whose checkpoint is chosen on a held-out 15% split. MAE configurations are selected by validation probe accuracy on a held-out 10% split and LLM checkpoints by validation loss (Appendix M).

4.1 Image Classification

ResNet-18 on CIFAR-10. AGAM-SGD attains the best cell over the grid under both protocols, with 95.75% after the final epoch and 95.48% under validation selection (Q1, Appendix O). Its clearest gains over SGDM appear at large learning rates (Q2), and it also achieves the best performance at small batch sizes. Under validation selection, SGDM collapses at $\eta = 0.8$ for $B \leq 256$, reaching only 40.7–55.5%, whereas AGAM stays between 92.1% and 94.2% (Figure 23). Under the same protocol, however, AGAM does not win every row. At $B = 4096$, TAM's 88.13% edges AGAM's 87.91%.

ResNet-50 on CIFAR-100. AGAM-SGD still attains the best cell of the full grid, 79.41% against 78.82% for SGDM, 79.03% for TAM, and 78.79% for AM, and it is the best optimizer in every batch-size row (Figure 2, Q1). The same holds under validation selection (Figure 24). Its advantage

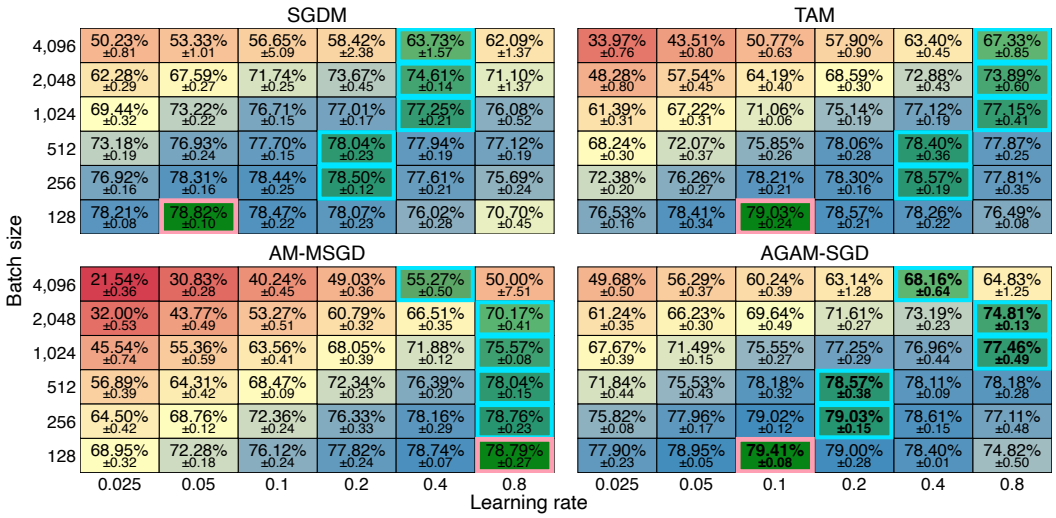
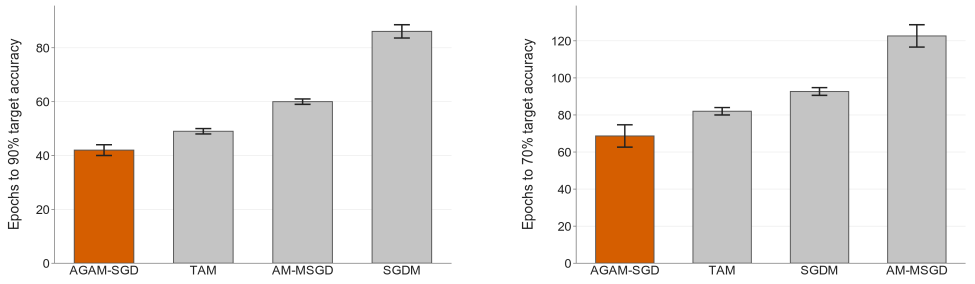


Figure 2: Final test accuracy (%) of ResNet-50 on CIFAR-100 over the 6×6 learning-rate $\times$ batch-size grid (mean $\pm$ s.e. over three seeds). Cyan marks each optimizer's best learning rate per batch size, pink its best cell overall, and **bold** the best cell among all optimizers in each row. AGAM-SGD holds the *best* cell overall and in *every* row.



(a) ResNet-18/CIFAR-10, target 90%.

(b) ResNet-50/CIFAR-100, target 70%.

Figure 3: Epochs until the validation accuracy first reaches a fixed target at $\eta = 0.1$, $B = 256$ (mean $\pm$ s.d. over three seeds). AGAM needs 42.0 epochs on CIFAR-10, against 49.0 for TAM, 60.0 for AM, and 85.3 for SGDM. It needs 68.6 on CIFAR-100, against 82.0, 122.7, and 92.7, respectively.

is again largest where SGDM struggles (Q2). At $B = 4096$, AGAM's best performance exceeds SGDM's by 4.4 points (68.16% against 63.73%). TAM and AM fail in the opposite corner of the grid: more precisely, at $\eta = 0.025$ and $B = 4096$ they drop to 33.97% and 21.54%, far below the 50.23% of the base optimizer they modify, while AGAM stays close to SGDM at 49.68%. Because the gate only attenuates β, AGAM takes slightly shorter effective steps than SGDM at the same η (Remark D.2), and its best learning rate in each row equals SGDM's or lies one grid point above it.

Convergence speed. AGAM-SGD also reaches a fixed validation accuracy in the fewest epochs on both tasks (Q3). At $\eta = 0.1$ and $B = 256$, it reaches 90% on CIFAR-10 in 42.0 epochs, against 49.0 for TAM, 60.0 for AM, and 85.3 for SGDM, which is about half as many as SGDM (Figure 3). On CIFAR-100, it reaches 70% in 68.6 epochs, against 82.0, 122.7, and 92.7, respectively, or $1.35\times$ fewer epochs than SGDM. This ordering is unchanged at $\eta = 0.05$ and $B = 128$ (Figure 26), and similar speedups appear across the other hyperparameter values in the grid. The gate is computationally light, adding only one inner product and two norms per tensor to each step.

4.2 MAE PRE-TRAINING AND FINE-TUNING

Table 1: MAE-pretrained ViT-Tiny on Tiny-ImageNet in % (mean ± s.e., three seeds, best in **bold**).

Optimizer	Linear probe	Fine-tune top-1	Fine-tune top-5
AdamW	28.66 ± 0.30	52.71 ± 0.05	77.86 ± 0.21
AdaTAMW	27.90 ± 0.25	51.05 ± 0.20	76.40 ± 0.18
AM-AdamW	29.17 ± 0.40	52.62 ± 0.26	77.73 ± 0.11
AGAM-AdamW	$\mathbf{29.77 \pm 0.16}$	$\mathbf{58.42 \pm 0.10}$	$\mathbf{81.99 \pm 0.09}$
Lion	28.22 ± 0.80	51.47 ± 0.40	76.98 ± 0.36
AGAM-Lion	$\mathbf{28.99 \pm 0.31}$	$\mathbf{59.51 \pm 0.26}$	$\mathbf{82.55 \pm 0.02}$

Table 1 evaluates the pre-trained encoders in two ways, with a linear probe on frozen class-token features and with end-to-end fine-tuning under one fixed 100-epoch AdamW recipe. With either base optimizer, AGAM produces the best encoder under both protocols (Q1, Q4). Although the probe gains are modest, $+1.11$ points over AdamW and $+0.77$ over Lion, they grow after fine-tuning to 5.71 and 8.04 points in top-1 accuracy over the AdamW- and Lion-based baselines, respectively. AdaTAMW and AM-AdamW, in contrast, stay within 1.7 points of AdamW.

The protocols measure different properties. All checkpoints are selected by probe accuracy and fine-tuned identically, so the gap comes from the pre-trained weights. The probe reads only the class token, which the reconstruction loss never supervises directly. Fine-tuning pools the patch tokens and adapts every layer, so it measures the encoder's value as an initialization, and MAE features are known to fine-tune far better than they probe (He et al., 2022). AGAM's gate is most active early in warmup, when 25–48% of tensor-steps are misaligned (Appendix E.3).

4.3 LLM FINE-TUNING

Table 2: SmolLM2-135M on WikiText-2, mean ± s.e. over three seeds.

Optimizer	Test PPL ↓
AdamW	14.405 ± 0.001
AdaTAMW	14.947 ± 0.003
AM-AdamW	14.402 ± 0.008
AGAM-AdamW	$\mathbf{14.388 \pm 0.002}$

Table 3: Qwen2.5-3B LoRA on GSM8K, mean ± s.e. over three seeds.

Optimizer	Exact match (%) ↑
AdamW	68.68 ± 0.31
AdaTAMW	68.15 ± 0.42
AM-AdamW	67.96 ± 0.34
AGAM-AdamW	$\mathbf{69.37 \pm 0.36}$
Lion	67.81 ± 0.21
AGAM-Lion	$\mathbf{69.56 \pm 0.50}$

On WikiText-2, AGAM-AdamW reaches the lowest test perplexity, although the margins remain small after 780 steps (Table 2). The differences are larger on GSM8K, where AGAM improves exact match by 0.69 points over AdamW and 1.75 points over Lion, while AdaTAMW and AM-AdamW both fall below plain AdamW (Table 3). AGAM is therefore the only alignment-aware method evaluated that improves on its base optimizer in both tasks (Q1, Q4).

5 CONCLUSION

We introduced AGAM, a simple drop-in mechanism that attenuates the contribution of historical momentum while preserving the base optimizer's state recurrences. Its augmented optimizers require no additional hyperparameters or special tuning. Our analysis establishes exact connections to quasi-hyperbolic optimization and a sharp descent certificate for AGAM-SGD. Across various tasks, AGAM consistently improves performance over its base optimizers and the evaluated alignment-aware baselines. These results support a practical design principle: adapt how accumulated momentum influences the current update while preserving the history available to future steps.

Reproducibility statement

Algorithm 1 specifies AGAM for a generic momentum-based optimizer, Equations 2, 5, 6, and 10 give the SGDM, AdamW, and Lion instantiations, and their PyTorch implementations are provided in Algorithm 2. Appendix M lists the architecture, data pipeline, training recipe, hyperparameter grid, selection rule, and seeds of every experiment, including the frozen linear-probe and fine-tuning protocols used to evaluate the MAE encoders. The complete assumptions and proofs of Lemma D.3, Proposition D.5, and Theorem D.6 are in Appendix F and Appendix G, and Appendix K describes the numerical checks of every derived identity against the implementation. We will release the implementation of AGAM's variants and of every baseline on GitHub.

Ethics statement

This work studies a general-purpose optimization algorithm and does not involve human subjects, personally identifiable information, or new data collection. All datasets and pre-trained models we use are publicly available and are used under their released licenses. We are not aware of any conflicts of interest.

AI use statement

We used generative AI tools in several parts of this work. They assisted with literature search, helping identify pertinent methods and candidate baselines. We also used them to proofread, polish, and restructure the text. Further assistance involved developing plotting code and checking analyses. The tools also helped check and refine mathematical derivations and proofs. We reviewed all AI-assisted contributions and take full responsibility for the final manuscript, including its text, claims, and results.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, et al. SmolLM2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning (ICML)*, 2018.

Yang Cao, Xiaoyu Li, and Zhao Song. Grams: Gradient descent with adaptive momentum scaling. *arXiv preprint arXiv:2412.17107*, 2024.

Da Chang and Ganzhao Yuan. MGUP: A momentum-gradient alignment update policy for stochastic optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 702–703, 2020.

Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of Adam and Adagrad. *Transactions on Machine Learning Research*, 2022.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

Igor Gitman, Hunter Lang, Pengchuan Zhang, and Lin Xiao. Understanding the role of momentum in stochastic gradient methods. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Gabriel Goh. Why momentum really works. *Distill*, 2017. doi: 10.23915/distill.00006.

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

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16000–16009, 2022.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Samy Jelassi and Yuanzhi Li. Towards understanding how momentum improves generalization in deep learning. In *International Conference on Machine Learning (ICML)*, 2022.

Taejong Joo, Wenhan Xia, Cheolmin Kim, Ming Zhang, and Eugene Ie. On the surprising effectiveness of masking updates in adaptive optimizers. *arXiv preprint arXiv:2602.15322*, 2026.

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.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Ya Le and Xuan Yang. Tiny ImageNet visual recognition challenge. *CS 231N course report, Stanford University*, 2015.

Jaerin Lee, Bong Gyun Kang, Kihoon Kim, and Kyoung Mu Lee. Grokfast: Accelerated grokking by amplifying slow gradients. *arXiv preprint arXiv:2405.20233*, 2024.

Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. Cautious optimizers: Improving training with one line of code. In *International Conference on Learning Representations (ICLR)*, 2025.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations (ICLR)*, 2020a.

Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020b.

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

Jerry Ma and Denis Yarats. Quasi-hyperbolic momentum and Adam for deep learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Pranshu Malviya, Gonçalo Mordido, Aristide Baratin, Reza Babanezhad Harikandeh, Gintare Karolina Dziugaite, Razvan Pascanu, and Sarath Chandar. Torque-aware momentum. *arXiv preprint arXiv:2412.18790*, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Yurii Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Brendan O'Donoghue and Emmanuel Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, 2015.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning (ICML)*, 2013.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of Adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2018.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning (ICML)*, 2018.

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 (ICML)*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, 2013.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude, 2012. COURSERA: Neural Networks for Machine Learning.

Kristi Topollai and Anna Choromanska. Adaptive memory momentum via a model-based framework for deep learning optimization. *arXiv preprint arXiv:2510.04988*, 2025.

Bao Wang, Tan Nguyen, Tao Sun, Andrea L. Bertozzi, Richard G. Baraniuk, and Stanley J. Osher. Scheduled restart momentum for accelerated stochastic gradient descent. *SIAM Journal on Imaging Sciences*, 15(2):738–761, 2022.

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

Yuxin Wu and Kaiming He. Group normalization. In *European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Yan Yan, Tianbao Yang, Zhe Li, Qihang Lin, and Yi Yang. A unified analysis of stochastic momentum methods for deep learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *International Conference on Learning Representations (ICLR)*, 2020.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George E. Dahl, Christopher J. Shallue, and Roger Grosse. Which algorithmic choices matter at which batch sizes? Insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI Conference on Artificial Intelligence*, volume 34, pp. 13001–13008, 2020.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James S. Duncan. AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

A Extended Related Work

Momentum and adaptive methods. We consider the stochastic minimization problem $\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) = \mathbb{E}_{\xi\sim\mathcal{D}}[\ell(\boldsymbol{\theta};\xi)]$, with stochastic gradient $\boldsymbol{g}_t = \nabla\ell(\boldsymbol{\theta}_t;\xi_t)$. Heavy-ball momentum (Polyak, 1964) accumulates $\boldsymbol{m}_t = \beta\boldsymbol{m}_{t-1} + \boldsymbol{g}_t$, and Nesterov acceleration (Nesterov, 1983) changes where momentum enters the gradient evaluation. Adaptive methods (Duchi et al., 2011; Tieleman & Hinton, 2012; Kingma & Ba, 2015) additionally rescale each coordinate by a second-moment estimate, a running sum in AdaGrad and an EMA $\boldsymbol{v}_t = \beta_2\boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t{}^2$ in RMSProp and Adam. Adam updates $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t\hat{\boldsymbol{m}}_t/(\sqrt{\hat{\boldsymbol{v}}_t}+\epsilon)$ with bias-corrected EMAs of both moments, and AdamW (Loshchilov & Hutter, 2019) decouples weight decay from this adaptive rescaling. On the theory side, Yan et al. (2018) give a unified stationarity analysis of stochastic momentum, which Liu et al. (2020b) sharpen. Gitman et al. (2019) characterize asymptotic convergence, local stability, and the stationary distribution of momentum methods, and Reddi et al. (2018) show that Adam can fail to converge under otherwise standard assumptions, which is why our stationarity bound for AGAM-Adam (Appendix D) assumes explicit spectral bounds on the preconditioner. In their canonical implementations β (or β_1) is a constant, so the averaging horizon $\approx 1/(1-\beta)$ is the same for every parameter and every phase of training. AGAM keeps this stored horizon and adapts how much of the memory enters the applied step.

Coefficient schedules. A separate line of work changes Adam's moment coefficients or statistics by a fixed rule: increasing β_2 over time (Reddi et al., 2018), rectifying the variance of the adaptive step early in training (Liu et al., 2020a), or replacing the second moment with the gradient's deviation from its EMA (Zhuang et al., 2020). Convergence proofs for adaptive momentum often lean on coefficient schedules: Défossez et al. (2022) take $\beta_2 = 1 - 1/T$, and MGUP's convergence theorem takes $\beta_{1,t} = 1 - t^{-1/2}$. Both tie the coefficient to the horizon or the step count rather than to the iterate. AGAM is instead *state-dependent*, and Theorem D.6 allows a fixed $\beta = 0.9$ under its step-size and spectral restrictions, at the price of an explicit stochastic residual.

The unit of adaptation. Treating each tensor as the unit of adaptation also has strong precedent: LARS (You et al., 2017) and LAMB (You et al., 2020) set per-layer trust ratios from weight and update norms, Adafactor (Shazeer & Stern, 2018) factorizes second-moment state per tensor, and Muon (Jordan et al., 2024) orthogonalizes updates matrix-wise. The granularity is attractive for the same reason here: it captures genuine heterogeneity across the network while keeping the bookkeeping to a handful of scalars and avoiding the noise of per-coordinate statistics. AGAM adopts the same unit for a different quantity: the temporal alignment between the optimizer's proposed step and the gradient that step is meant to follow.

Magnitude-based controls. Further from AGAM, signSGD (Bernstein et al., 2018) and Lion (Chen et al., 2023) discard update magnitude entirely, Grokfast (Lee et al., 2024) reshapes the frequency content of the gradient with fixed coefficients, and gradient clipping (Pascanu et al., 2013) bounds magnitude but is blind to direction.

B Design Rationale in Detail

This appendix expands Section 3.5.

Preserving the base momentum recursion. AGAM leaves the base optimizer's persistent momentum recursion $\boldsymbol{m}_t = \mathcal{M}(\beta, \boldsymbol{m}_{t-1}, \boldsymbol{g}_t)$ *untouched*: the gate is applied transiently, altering exclusively the applied step. This preserves the accumulated gradient history needed to navigate ill-conditioned, high-curvature ravines (Goh, 2017). Because the gate depends on a stochastic mini-batch gradient, this design avoids permanently attenuating the stored history in response to a single potentially spurious misalignment measurement. The gate can still affect future momentum by changing the parameter trajectory, and the proof of Theorem D.6 controls this effect through the tracking error of the stored moment.

Tensor-wise gating. AGAM computes one scalar alignment gate $\gamma_t^{(k)}$ independently for each parameter tensor k at every step, so different tensors can retain different effective momentum

coefficients $\beta_{\text{eff},t}^{(k)} = \gamma_t^{(k)}\beta$ rather than sharing a single model-wide response. A model-wide cosine aggregates tensors that may be in substantially different alignment states: it is dominated by the tensors with the largest gradient and update norms, so agreement in some tensors can offset disagreement in others. Appendix E shows that AGAM's gates indeed follow different trajectories across parameter groups and depths.

Alignment with the proposed update. AGAM evaluates alignment against the base optimizer's proposed direction $\boldsymbol{u}_t$ (including any preconditioning), *after* incorporating the current gradient, rather than against the previous, stale momentum $\boldsymbol{m}_{t-1}$. The justification for this choice is three-fold: intuitive, theoretical, and empirical. First, because deep learning training is inherently stochastic, each gradient is computed on a slightly different landscape whose topology is defined by the mini-batch sampled at that step. Hence, $\boldsymbol{m}_{t-1}$ reflects the average landscape seen up to step $t-1$, whereas $\boldsymbol{g}_t$ reflects only the most recent one. To avoid mistaking mini-batch noise for genuine directional misalignment, we fold $\boldsymbol{g}_t$ into the buffer before measuring alignment, which yields an up-to-date reference direction. Moreover, the first-order change in the mini-batch loss along $-\boldsymbol{u}_t$ is proportional to $-\langle \boldsymbol{g}_t, \boldsymbol{u}_t \rangle$, which motivates gradient–update alignment as the gating signal. Disagreement between $\boldsymbol{m}_{t-1}$ and $\boldsymbol{g}_t$, by contrast, does not imply that the resulting proposal opposes $\boldsymbol{g}_t$.

Second, incorporating the current gradient makes the probe deliberately conservative and *norm-sensitive*. Consider heavy-ball momentum, for which $\boldsymbol{u}_t = \boldsymbol{m}_t = \beta\boldsymbol{m}_{t-1} + \boldsymbol{g}_t$. For nonzero $\boldsymbol{g}_t$ and $\boldsymbol{m}_{t-1}$ satisfying geometric orthogonality, $\langle \boldsymbol{g}_t, \boldsymbol{m}_{t-1} \rangle = 0$, we obtain $\operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{m}_t) = \|\boldsymbol{g}_t\|/\|\boldsymbol{m}_t\| = \left(1 + \beta^2 \|\boldsymbol{m}_{t-1}\|^2/\|\boldsymbol{g}_t\|^2\right)^{-1/2} > 0$. Thus, the cosine still yields a positive alignment despite the orthogonality, because the proposed-direction statistic reflects the relative norms. Only when $\beta\|\boldsymbol{m}_{t-1}\| \gg \|\boldsymbol{g}_t\|$, so that adding the current gradient leaves the proposed direction essentially unaltered, does the similarity approach zero and γ_t approach $1/2$. In that case, the momentum coefficient is halved ($\beta_{\text{eff},t} \approx 0.5\,\beta$) to reduce its influence on the update, making it more gradient-driven. More generally, for $0 < \beta < 1$ and nonzero vectors, the proposed update is misaligned precisely when

$$\gamma_t < \tfrac{1}{2} \iff \beta\langle \boldsymbol{g}_t, \boldsymbol{m}_{t-1} \rangle < -\|\boldsymbol{g}_t\|^2 \iff \beta \underbrace{\frac{\|\boldsymbol{m}_{t-1}\|}{\|\boldsymbol{g}_t\|}}_{\text{relative norm}} \underbrace{\operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{m}_{t-1})}_{\text{angular alignment}} < -1. \tag{13}$$

Unlike the sign of the stale-buffer cosine alone, this condition depends jointly on the angular disagreement and on the relative magnitude of the historical component. Appendix D makes the argument quantitative: under the independent zero-mean Gaussian model, the stale-buffer cosine has expectation zero, while the updated-probe cosine concentrates around a positive value in high dimension (Proposition D.5). Appendix E compares this reference value with training telemetry.

Third, because the misalignment condition in Equation 13 is norm-sensitive, it requires the scaled momentum norm $\beta\|\boldsymbol{m}_{t-1}\|$ to exceed the gradient norm. If gradient norms fall faster than the stored buffer adapts, this relative norm grows and misalignment becomes reachable, although learning-rate decay alone does not imply that ordering. Consistent with this, the share of misaligned steps rises again late in training under warmup and cosine decay, but not under a constant learning rate: on ResNet-50, the misalignment rate of AGAM-SGD's normalization parameters bottoms out near 0.2% around epochs 20–30 and climbs back to about 2% as the learning rate anneals (Figure 11), and the biases of AGAM-AdamW during MAE pre-training rise to 1–3% over the final fifty epochs (Figure 20). The gate is therefore most active in the standard practical setting, which highlights a practical benefit of folding the current gradient into the buffer before computing the similarity.

C Connections to Related Methods

AGAM does not write the gated momentum to the state because the gate is driven by a single stochastic mini-batch gradient: a strong but spurious misalignment would otherwise permanently erase the buffer's accumulated memory, discarding the weighted gradient history. AGAM also introduces no additional hyperparameters and merely inherits those of its base optimizer, whereas TAM adds a smoothing rate and a gradient floor, and AM a proximal weight and a momentum cap. A related distinction is that AGAM *attenuates* the coefficient used in the immediate direction rather than replacing the stored one: $\beta_{\text{eff},t}^{(k)} = \gamma_t^{(k)}\beta \in [0, \beta]$, while the stored recurrence keeps β, and the

base direction is recovered exactly at $\gamma_t^{(k)} = 1$. AM instead adapts the coefficient inside the stored recurrence, clipped to $[0, \beta_{\max}]$, which turns the cap $\beta_{\max}$ into a new hyperparameter, and TAM keeps its stored coefficient fixed and damps the incoming gradient.

The formulation of AGAM's adaptive gating coefficient in Equation 1 uses the same affine remapping ϕ of s_t as the TAM optimizer (Malviya et al., 2024). However, unlike a model-wide gate, used by TAM and by the global AM variants (Topollai & Choromanska, 2025), AGAM computes one scalar alignment gate $\gamma_t^{(k)}$ independently for each parameter tensor k at every optimization step. This fine-grained resolution lets each tensor respond to its own local optimization geometry. Consequently, different tensors may retain substantially different effective momentum coefficients $\beta_{\text{eff},t}^{(k)} = \gamma_t^{(k)}\beta$ at the same step and follow different evolution profiles, as we show in Appendix E. A model-wide cosine, in contrast, aggregates tensors that may be in very different alignment states. Writing it as $\sum_k w_k \operatorname{cossim}(\boldsymbol{g}_t^{(k)}, \boldsymbol{u}_t^{(k)})$ with $w_k = \|\boldsymbol{g}_t^{(k)}\|\|\boldsymbol{u}_t^{(k)}\|/(\|\boldsymbol{g}_t\|\|\boldsymbol{u}_t\|)$ shows how tensors with large gradient and update norms can dominate it. These weights sum to at most one and need not sum to one, so cancellation across tensors and norm imbalance can obscure tensor-level disagreement. AM also has a per-layer AdamW variant, and our AM-AdamW baseline uses per-tensor coefficients, so tensor-wise adaptation alone does not distinguish AGAM from every AM formulation.

A further distinction is that AGAM acts entirely through the base optimizer's momentum map $\mathcal{M}$, applied transiently, whereas TAM damps the incoming gradient inside its recursion and AM modulates both terms through a weighted combination within a fixed framework. AGAM and AM resolve the momentum-gradient conflict in the same direction: strong disagreement shrinks the effective momentum coefficient, making the update more gradient-driven and less dependent on stale history. TAM instead shrinks the mini-batch gradient upon misalignment, giving precedence to the momentum direction, which is the opposite resolution of the same conflict. TAM smooths its stale-buffer statistic with an EMA.

Perhaps the most consequential difference, however, is that AGAM derives its gate from the *updated* direction $\boldsymbol{u}_t$ proposed by the base optimizer, rather than from the prior, stale buffer $\boldsymbol{m}_{t-1}$. To see why this matters, consider the zero-mean isotropic noise model of the steady state, $\boldsymbol{g}_t \sim \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$, with gradients at different iterations independent, $\boldsymbol{g}_t \perp\!\!\!\perp \boldsymbol{g}_{t-1}, \boldsymbol{g}_{t-2}, \ldots$. Then $\boldsymbol{g}_t$ is independent of $\boldsymbol{m}_{t-1}$ and symmetric about zero, so

$$\mathbb{E}[\boldsymbol{m}_{t-1}^\top \boldsymbol{g}_t] = 0 \qquad \text{and} \qquad \mathbb{E}[\operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{m}_{t-1})] = 0.$$

At equilibrium, $\boldsymbol{g}_t$ and $\boldsymbol{m}_{t-1}$ are therefore approximately orthogonal, especially in high dimensions, and the stale-buffer cosine carries little information about the relationship between the two directions. Consistent with this, Malviya et al. (2024, App. A.2.1) report that TAM's smoothed correlation $\hat{s}_t$ saturates near zero as training progresses. Since TAM also initializes $\hat{s}_0 = 0$, its damping factor $(1 + \hat{s}_t)/2$ stays close to $1/2$: on average, TAM simply halves the incoming mini-batch gradient before committing it to memory, giving precedence to the variance-reduced momentum over the instantaneous stochastic direction.

By absorbing the current gradient into the momentum buffer first, AGAM's statistic is instead *norm-aware*. In the same orthogonal case, the heavy-ball probe gives $\operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{m}_t) = \left(1 + \beta^2 \|\boldsymbol{m}_{t-1}\|^2/\|\boldsymbol{g}_t\|^2\right)^{-1/2} > 0$ (Appendix B): a gradient whose norm is comparable to or larger than the scaled momentum norm still yields positive alignment, indicating that it still shapes the effective update direction. Only when the momentum's relative norm is much larger does adding the current gradient leave the proposed direction essentially unaltered, driving the similarity to zero and the gate to $1/2$, which halves the momentum coefficient and makes the update more gradient-driven.

D Theoretical Analysis

We distinguish exact update identities, geometric statements about the observed gradient, and convergence under an explicit stochastic oracle model. The analysis uses fixed first-moment coefficients in $[0, 1)$ and the default transient gate with exponent one, no norm matching, and no Nesterov term. AGAM-SGD uses the undampened heavy-ball buffer, and AGAM-Adam uses complementary gradient weighting and the coefficient-mass normalization in equation 5. Decoupled weight decay is excluded from the convergence theorem and treated in Appendix G.1.

The identities and geometric results apply to individual parameter tensors. The convergence theorem instead treats the concatenated parameter vector, with a separate gate for each tensor and a global smoothness constant. Mere coordinatewise smoothness does not justify replacing this constant by separate block constants when all blocks move simultaneously.

D.1 Exact representations and state preservation

Proposition D.1 (Quasi-hyperbolic representations)**.** *For each tensor, the AGAM-SGD direction is*

$$\boldsymbol{u}_{\mathit{eff},t} = (1-\gamma_t)\boldsymbol{g}_t + \gamma_t \boldsymbol{m}_t, \qquad \boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \boldsymbol{g}_t.$$

For AGAM-Adam, with $\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$*,*

$$\boldsymbol{u}_{\mathit{eff},t} = \mathbf{D}_t^{-1}\big[(1-\nu_t)\boldsymbol{g}_t + \nu_t \hat{\boldsymbol{m}}_t\big], \qquad \nu_t = \frac{\gamma_t(1-\beta_1^t)}{1-\gamma_t\beta_1^t} \in [0,\gamma_t].$$

Thus AGAM-Adam has the QHAdam update form with $\nu_2 = 1$ *and an adaptive first-moment mixture. AGAM-SGD has a QHM representation with the gain given in Remark D.2. AGAM-Lion applies the sign of the QHM mixture in equation 11, but the signed step itself is not a QHM step.*

For SGD and Adam, setting every gate to one recovers the corresponding base update, and setting it to zero removes the historical first-moment contribution for that step. Adam's second-moment history remains present. These identities concern the gradient-driven direction, and identical decoupled weight decay can be added to both sides. Normalizing Adam's realized coefficient mass is an algebraic correction, not a claim of statistical unbiasedness for a gradient-dependent gate.

State preservation means that the stored moments follow the base recurrences on the gradients encountered by AGAM. A separate base-optimizer run generally encounters different gradients and therefore has different numerical states. The applied memory coefficient lies in $[0,\beta]$, while the decay rate of the stored memory remains β. For SGD, $\|\boldsymbol{u}_{\text{eff},t}\| \le \max\{\|\boldsymbol{g}_t\|, \|\boldsymbol{m}_t\|\}$.

With empty memory and a nonzero gradient, the ideal SGD gate equals one. For Adam and Lion the proposed direction need not be parallel to the gradient, so their first-step gates need not equal one. Nevertheless, Adam's coefficient-mass normalization and Lion's sign operation recover their base updates on the first step. Exact cosine scale invariance concerns its two vector arguments. Numerical norm floors and Adam's fixed additive ϵ can break invariance of an entire run to rescaling the loss.

Remark D.2 (Effective gain). For AGAM-SGD, set $\boldsymbol{z}_t = (1-\beta)\boldsymbol{m}_t$. Then

$$\boldsymbol{u}_{\text{eff},t} = k_t\big[(1-\nu_t)\boldsymbol{g}_t + \nu_t \boldsymbol{z}_t\big], \qquad k_t = \frac{1-\beta+\beta\gamma_t}{1-\beta}, \qquad \nu_t = \frac{\gamma_t}{1-\beta+\beta\gamma_t}.$$

The gate changes both the QHM mixture and its gain. Even with a constant gate γ, the historical gradient weights are $\gamma\beta^j$ for $j \ge 1$, not $(\gamma\beta)^j$. A fixed gate therefore gives a fixed QHM mixture with stored decay β, not heavy ball with decay $\gamma\beta$.

D.2 Geometry of the applied direction

Lemma D.3 (Sharp descent certificate for AGAM-SGD)**.** *Let* $\boldsymbol{g}_t \neq 0$*,* $\boldsymbol{m}_t \neq 0$*, and use the ideal cosine gate* $\gamma_t = (1+s_t)/2$*, where* $s_t = \operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{m}_t)$ *and* $\rho_t = \|\boldsymbol{m}_t\|/\|\boldsymbol{g}_t\|$*. Then*

$$\frac{\langle \boldsymbol{g}_t, \boldsymbol{u}_{\mathit{eff},t}\rangle}{\|\boldsymbol{g}_t\|^2} = \frac{1-s_t+\rho_t s_t(1+s_t)}{2}. \tag{14}$$

The score is strictly positive if $s_t \ge 0$ *or* $s_t = -1$*. For* $-1 < s_t < 0$*, it is positive exactly when*

$$\rho_t < \frac{1-s_t}{-s_t(1+s_t)}.$$

Consequently $\rho_t < 3+2\sqrt{2}$ *guarantees strict positivity at every angle. At* $\rho_t = 3+2\sqrt{2}$ *the score is nonnegative and vanishes only at* $s_t = 1-\sqrt{2}$*. The constant is sharp in dimension at least two: every larger norm ratio permits a negative score. At exact antiparallel alignment, the gate is zero and* $\boldsymbol{u}_{\mathit{eff},t} = \boldsymbol{g}_t$*.*

The certificate concerns the observed gradient and the gradient-driven direction. It does not guarantee descent of the population objective for a noisy mini-batch, nor a decrease for an arbitrary learning rate. If $\boldsymbol{g}_t = \nabla f(\boldsymbol{\theta}_t)$, f is L-smooth, and $\langle \boldsymbol{g}_t, \boldsymbol{u}_{\text{eff},t} \rangle > 0$, then

$$0 < \eta_t < \frac{2\langle \boldsymbol{g}_t, \boldsymbol{u}_{\text{eff},t} \rangle}{L\|\boldsymbol{u}_{\text{eff},t}\|^2} \quad \Longrightarrow \quad f(\boldsymbol{\theta}_t - \eta_t \boldsymbol{u}_{\text{eff},t}) < f(\boldsymbol{\theta}_t).$$

For nonzero directions, the SGD interpolation also gives $\operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{u}_{\text{eff},t}) \geq \operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{m}_t)$. The angle can improve while remaining obtuse: $\boldsymbol{g}_t = (1, 0)$ and $\boldsymbol{m}_t = (-10, 10)$ give $\langle \boldsymbol{g}_t, \boldsymbol{u}_{\text{eff},t} \rangle \approx -0.611$. Numerical norm clamping preserves the interpolation but not the angle-specific constant above (Appendix F.1).

For Adam, the same cone argument applies relative to $\mathbf{D}_t^{-1}\boldsymbol{g}_t$, not necessarily to $\boldsymbol{g}_t$. Writing $A = \boldsymbol{g}_t{}^\top \mathbf{D}_t^{-1} \boldsymbol{g}_t > 0$ and $B = \boldsymbol{g}_t{}^\top \mathbf{D}_t^{-1} \hat{\boldsymbol{m}}_t$ gives

$$\langle \boldsymbol{g}_t, \boldsymbol{u}_{\text{eff},t} \rangle = (1 - \nu_t)A + \nu_t B. \tag{15}$$

If $B < 0$, this is positive exactly when $\nu_t < A/(A - B)$, but the cosine gate does not enforce that condition universally.

Proposition D.4 (Signed agreement for AGAM-Lion)**.** *Let* $\boldsymbol{c}_t = \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$ *and* $\widetilde{\boldsymbol{c}}_t = (1 - \gamma_t)\boldsymbol{g}_t + \gamma_t \boldsymbol{c}_t$*, with* $\gamma_t \in [0, 1]$*. Using* $\operatorname{sign}(0) = 0$*,*

$$\langle \boldsymbol{g}_t, \operatorname{sign}(\widetilde{\boldsymbol{c}}_t) \rangle \geq \langle \boldsymbol{g}_t, \operatorname{sign}(\boldsymbol{c}_t) \rangle.$$

Proof. Coordinatewise, mixing $c_{t,i}$ with $g_{t,i}$ cannot change an agreeing sign into a disagreeing sign. A disagreeing sign can remain unchanged, become zero, or become agreeing. Thus $g_{t,i} \operatorname{sign}(\widetilde{c}_{t,i}) \geq g_{t,i} \operatorname{sign}(c_{t,i})$, and summing over i proves the claim. □

The resulting score can still be negative. This comparison neither extends the SGD norm-ratio certificate to Lion nor includes decoupled weight decay. The one-step smoothness interpolation for SGD is proved separately in Appendix F.3. It is an upper bound, not a guarantee that the chosen gate minimizes loss.

D.3 A LOCAL NOISE MODEL FOR THE GATE

The inclusion of the current gradient in the probe creates positive alignment even without population signal. The following calculation makes that effect explicit for AGAM-SGD.

Proposition D.5 (High-dimensional Gaussian reference)**.** *Fix* $0 < \beta < 1$ *and* $\sigma > 0$*. In dimension* d*, let* $\boldsymbol{g}_t = \boldsymbol{\mu}_d + \boldsymbol{\xi}_t$*, with* $\boldsymbol{\xi}_t$ *independent across time and distributed as* $\mathcal{N}(0, \sigma^2 \boldsymbol{I}_d)$*. Use the equilibrated heavy-ball buffer and the ideal cosine. If* $\|\boldsymbol{\mu}_d\|^2/(d\sigma^2) = \kappa$ *is fixed as* $d \to \infty$*, then* $s_t = s_\infty(\kappa) + \mathcal{O}_p(d^{-1/2})$*, where*

$$s_\infty(\kappa) = \frac{\kappa + 1 - \beta}{\sqrt{(\kappa + 1)\{\kappa + (1 - \beta)/(1 + \beta)\}}}, \qquad \gamma_\infty(\kappa) = \frac{1 + s_\infty(\kappa)}{2}. \tag{16}$$

The gate converges in probability and in expectation to $\gamma_\infty(\kappa)$*. The limiting cosine is strictly increasing in* $\kappa \geq 0$ *and tends to one as* $\kappa \to \infty$*. At zero signal,*

$$\gamma_\infty(0) = \frac{1 + \sqrt{1 - \beta^2}}{2} \approx 0.71794 \quad (\beta = 0.9). \tag{17}$$

For this zero-mean Gaussian model only, $\operatorname{Var}(\gamma_t) = \beta^4/(4d) + o(d^{-1})$.

Equation 17 is a reference value for a specified limiting model, not a lower bound on individual gates or on means measured during training. Finite tensors can have gates below $1/2$, and anisotropy, temporal dependence, and a drifting mean can change the reference. Isotropy alone does not imply the concentration used in the proof. A gate near 0.718 therefore does not itself demonstrate useful signal or improved SNR. In the same zero-mean model, the stale-buffer cosine has expectation zero and converges to zero in high dimension. These statements compare statistics and do not assert that TAM gates momentum in the same way as AGAM. The calculation also does not apply directly to the preconditioned Adam or signed Lion probes.

D.4 STATIONARITY WITH FIXED MOMENTUM

For the full parameter vector, both AGAM-SGD and AGAM-Adam admit

$$\boldsymbol{u}_{\text{eff},t} = \boldsymbol{H}_t\big[(\boldsymbol{I} - \boldsymbol{N}_t)\boldsymbol{g}_t + \boldsymbol{N}_t\boldsymbol{z}_t\big], \qquad 0 \preceq \boldsymbol{N}_t \preceq \boldsymbol{I}, \tag{18}$$

where $\boldsymbol{N}_t$ is block-scalar. For SGD, $\boldsymbol{z}_t = (1-\beta)\boldsymbol{m}_t$ and $\boldsymbol{H}_t = \operatorname{blockdiag}(k_t^{(k)}\boldsymbol{I})$, with k_t and ν_t from Remark D.2. For Adam, $\boldsymbol{z}_t = \hat{\boldsymbol{m}}_t$, $\boldsymbol{H}_t = \mathbf{D}_t^{-1}$, and the blocks of $\boldsymbol{N}_t$ are given by equation 8.

Let $\mathcal{F}_{t-1}$ contain the initialization and all randomness before sampling $\boldsymbol{g}_t$. We assume a deterministic initial point and the following.

Assumption 1 (Smooth objective)**.** *f has an L-Lipschitz gradient, with $L > 0$, and is bounded below by $f^\star$.*

Assumption 2 (Unbiased oracle with bounded conditional variance)**.** *$\boldsymbol{g}_t = \nabla f(\boldsymbol{\theta}_t) + \boldsymbol{\xi}_t$, where $\mathbb{E}[\boldsymbol{\xi}_t \mid \mathcal{F}_{t-1}] = 0$ and $\mathbb{E}[\|\boldsymbol{\xi}_t\|^2 \mid \mathcal{F}_{t-1}] \le \sigma_t^2$ for deterministic finite variance bounds σ_t^2.*

Assumption 3 (Moment clocks and spectral bounds)**.** *All blocks receive a gradient at every step, their moment states start at zero, and they share a fixed first-moment coefficient $\beta \in (0,1)$ ($\beta = \beta_1$ for Adam). There are deterministic constants $0 < h_- \le h_+ < \infty$ such that $h_-\boldsymbol{I} \preceq \boldsymbol{H}_t \preceq h_+\boldsymbol{I}$ almost surely.*

For SGD the spectral bounds hold with $h_- = 1$ and $h_+ = (1-\beta)^{-1}$. For Adam, a bound $\|\boldsymbol{g}_t\|_\infty \le G_\infty$ on all gradients suffices, with $h_- = (G_\infty + \epsilon)^{-1}$ and $h_+ = \epsilon^{-1}$. The lower spectral bound is an additional assumption, since a positive ϵ alone supplies only the upper bound. Neither $\boldsymbol{H}_t$ nor $\boldsymbol{N}_t$ is assumed independent of the current gradient.

Theorem D.6 (Stationarity bound)**.** *Under Assumptions 1–3, with no decoupled weight decay, define $\boldsymbol{e}_t = \boldsymbol{z}_t - \nabla f(\boldsymbol{\theta}_t)$, $\kappa_{\boldsymbol{H}} = h_+/h_-$, and $\bar{\sigma}_T^2 = T^{-1}\sum_{t=1}^T \sigma_t^2$. For a constant step size satisfying*

$$0 < \eta \le \min\left\{\frac{h_-}{4Lh_+^2},\ \frac{(1-\beta)h_-}{\sqrt{24}\,\beta L h_+^2}\right\}, \tag{19}$$

the iterates satisfy

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^T \mathbb{E}\|\nabla f(\boldsymbol{\theta}_t)\|^2 \le{}& \frac{8(f(\boldsymbol{\theta}_1) - f^\star)}{\eta h_- T} + \frac{12\kappa_{\boldsymbol{H}}^2\mathbb{E}\|\boldsymbol{e}_1\|^2}{(1-\beta)T} \\ &+ 12\kappa_{\boldsymbol{H}}^2\left(1 + \frac{1}{1-\beta}\right)\bar{\sigma}_T^2.\end{aligned} \tag{20}$$

Appendix G gives a full proof. It controls the gradient-dependent mixture pointwise and uses conditional unbiasedness only for the ordinary stored-moment recurrence. With exact gradients, the theorem gives an $\mathcal{O}(1/T)$ average squared-gradient bound. At fixed nonzero oracle variance it gives a stationarity neighborhood, not a vanishing stochastic rate. Decreasing the step size alone does not remove the noise term in this bound. Vanishing-variance consequences, a deterministic PL result, and weight decay are treated in Appendix G.

This result is proved directly for the stated updates. A QHAdam representation does not automatically transfer a theorem for fixed mixing weights or for recursively varying momentum. Likewise, substituting Adam's time-varying preconditioner into a constant-step heavy-ball auxiliary sequence would introduce additional variation and dependence terms (Reddi et al., 2018; Défossez et al., 2022). The theorem does not cover Lion's nonlinear sign operation.

Remark D.7 (Alignment geometry for Adam). The default cosine compares $\boldsymbol{g}_t$ with $\mathbf{D}_t^{-1}\hat{\boldsymbol{m}}_t$. Its numerator is exactly the observed-gradient descent score of the base direction. The metric cosine $\operatorname{cossim}(\mathbf{D}_t^{-1/2}\boldsymbol{g}_t, \mathbf{D}_t^{-1/2}\hat{\boldsymbol{m}}_t)$ has the same numerator and therefore the same sign. The raw-moment cosine and the cosine after applying $\mathbf{D}_t^{-1}$ to both vectors need not have that sign (Appendix J gives explicit counterexamples).

These results do not establish acceleration, improved generalization, or uniformly improved SNR. The frozen-gate calculation in Appendix H separates variance reduction from tracking a changing mean, and Appendix I shows why an unbiased oracle need not yield an unbiased gated update. Performance comparisons are established by the experiments, not by the stationarity bound.

E GATE & MISALIGNMENT TELEMETRY

This appendix records AGAM's gate for every tensor at every step in three settings: AGAM-SGD on ResNet-18/CIFAR-10 and on ResNet-50/CIFAR-100, each under warmup + cosine decay and under a constant learning rate, and AGAM-AdamW during MAE pre-training. A tensor-step is *misaligned* when $\mathrm{cossim}(\boldsymbol{g}_t, \boldsymbol{u}_t) < 0$, equivalently $\gamma_t^{(k)} < 0.5$.

E.1 AGAM-SGD ON RESNET-18/CIFAR-10

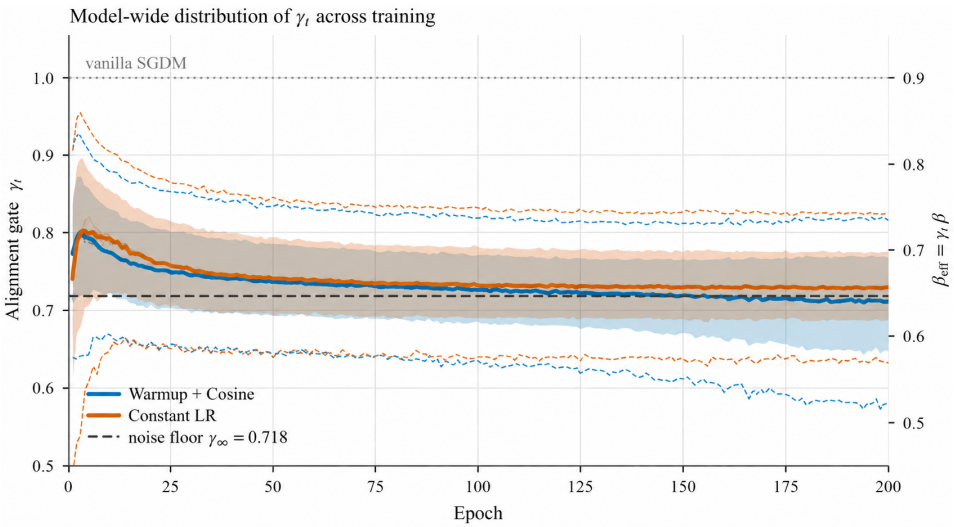
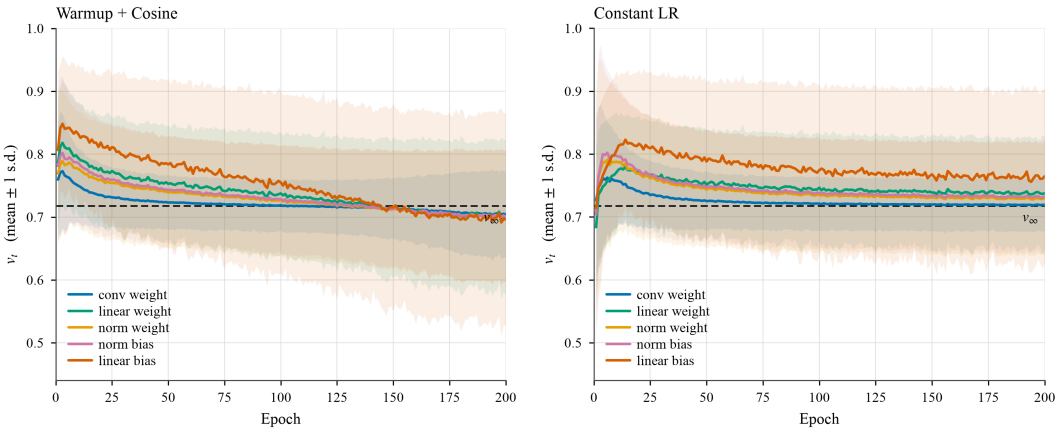


Figure 4: Model-wide distribution of the AGAM-SGD gate on ResNet-18/CIFAR-10 ($\beta = 0.9$) under a warmup + cosine schedule and a constant learning rate. The gate descends from near-saturation early in training and settles near the Gaussian reference value $\gamma_\infty = 0.718$ of equation 17, which depends on β alone.



(a) Warmup + cosine schedule.

(b) Constant learning rate.

Figure 5: Per-parameter-kind gate trajectories of AGAM-SGD on ResNet-18/CIFAR-10 (the plots label the gate v_t), expanding observation (O2). Every kind starts near saturation and relaxes toward the reference value γ_∞, but at markedly different rates and to different steady-state values. Under the decayed schedule all kinds approach γ_∞ and then fall below it, whereas under a constant learning rate they stratify and remain separated, with the classifier bias holding the highest gate throughout.

Proposition D.5 supplies the reference $\frac{1}{2}(1 + \sqrt{1 - \beta^2}) = 0.718$ at $\beta = 0.9$ for an equilibrated high-dimensional tensor under independent zero-mean Gaussian noise. We use it as a comparison

level, without assuming that the training gradients satisfy that model. We instrument AGAM-SGD on ResNet-18/CIFAR-10 and record $\gamma_t^{(k)}$ for every tensor at every step, under a warmup + cosine schedule and a constant learning rate.

(O1) Late-training gates are near the Gaussian reference. The reported late-training model-mean gates are 0.705 under warmup + cosine and 0.728 under a constant learning rate, compared with the Gaussian reference 0.718 (Figure 4). These values are numerically close but do not validate the local model. In particular, the limiting formula is minimized at zero SNR, so a value below 0.718 cannot be explained merely by a lower nonnegative SNR within that formula. Finite dimension, anisotropy, temporal dependence, and non-stationarity can all contribute. The reported correlation with training loss ($r = 0.99$) is descriptive, since training loss is not itself an estimate of gradient SNR. A nearly constant gate would correspond to a nearly fixed QHM mixture, while the stored momentum decay remains $\beta = 0.9$.

(O2) Conflict is strongly heterogeneous across tensors. At steady state the mean gate ranges from 0.700 to 0.764 across parameter kinds, and the fraction of conflicting steps ($s_t < 0$, i.e. $\gamma_t^{(k)} < 0.5$) ranges from 0.3% for convolutional weights to 13.0% for the classifier bias in the same run (Figure 5). Over depth, the share of high-alignment steps rises monotonically from stem to head while the conflict share is U-shaped. A model-wide scalar must therefore either under-correct the conflicting tensors or over-damp the aligned ones, and would in any case be dominated by the convolutional weights that hold most of the parameters and almost never conflict. This is the empirical content of the per-tensor design.

(O3) Observed conflict and population stationarity are distinct. The classifier-bias measurements identify steps on which the proposed update opposes the consumed mini-batch gradient. The descent certificate predicts when gating removes that conflict only after the probe-to-gradient norm ratio is also taken into account. Misalignment percentages alone do not measure the true-gradient tracking error in Theorem D.6, and cannot establish a better convergence rate. Comparing proposed and applied directions at the same step isolates the immediate geometric effect of the gate, whereas comparing different optimizers also includes their different parameter trajectories.

E.2 AGAM-SGD on ResNet-50/CIFAR-100

We repeat the instrumentation on a GroupNorm ResNet-50 trained on CIFAR-100 for 200 epochs with AGAM-SGD ($\beta = 0.9$, one seed), again under warmup + cosine decay and under a constant learning rate without warmup. The 161 parameter tensors are pooled into five groups: convolutional weights, normalization scales, normalization biases, and the weight and bias of the classification head. Unless stated otherwise, statistics are pooled over tensor-steps, that is, over every tensor of a group at every optimizer step of an epoch. Epoch axes use a square-root scale to resolve the early transient, and misalignment rates use a symmetric-log axis that is linear below 0.01%, so a plotted zero is an exact zero. A tensor-step is *misaligned* when $\mathrm{cossim}(\boldsymbol{g}_t, \boldsymbol{u}_t) < 0$, equivalently $\gamma_t^{(k)} < 0.5$. The gate itself is reported in Figures 6–10, misalignment in Figures 11–16, and the granularity ablation in Figure 17.

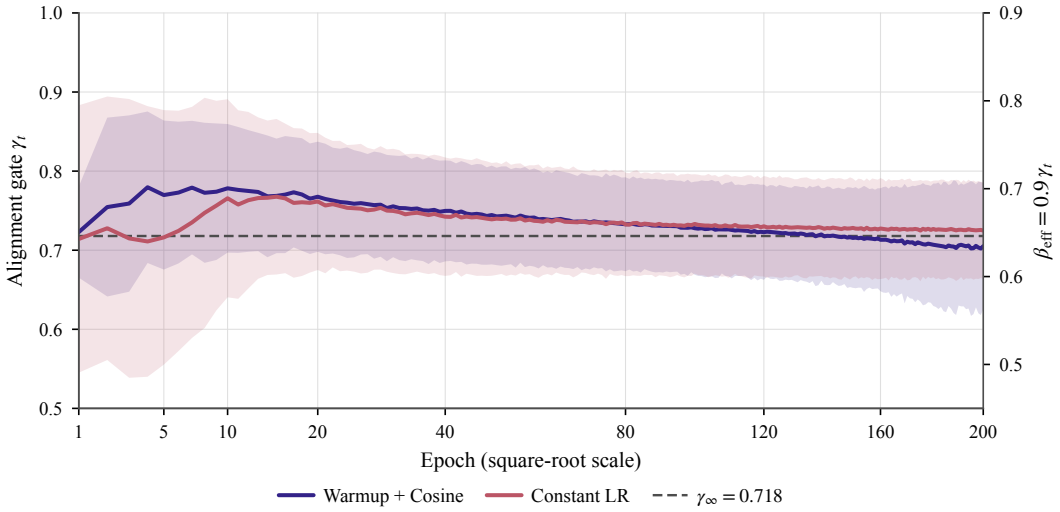


Figure 6: Model-mean gate of AGAM-SGD on ResNet-50/CIFAR-100. Solid lines are the mean of $\gamma_t^{(k)}$ over all 161 tensors and all steps of an epoch, and shaded bands span ± 1 standard deviation of the same tensor-step observations. The dashed line is the Gaussian reference value $\gamma_\infty = 0.718$ of equation 17, and the right axis gives the corresponding effective momentum $\beta_{\text{eff}} = 0.9\,\gamma_t$. With warmup the mean gate rises to about 0.78 within the first ten epochs. Without warmup it stays near γ_∞ for the first five epochs, peaks at 0.77 around epoch 15, and has a much wider spread across tensors early on. Both runs then relax toward the reference level: the decayed run crosses below it after roughly epoch 130 and ends near 0.705, while the constant-rate run ends marginally above it, near 0.725. This is the same ordering observed on ResNet-18 in (O1).

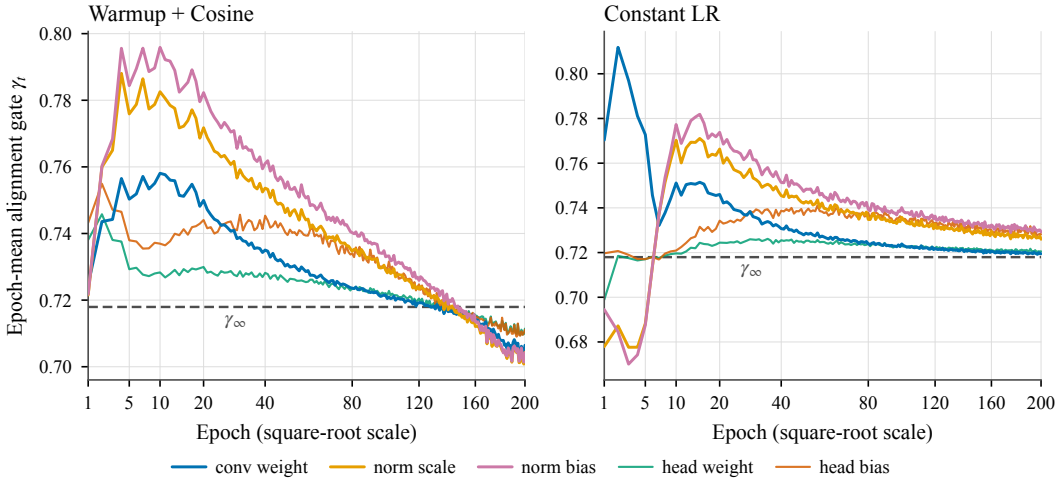


Figure 7: Epoch-mean gate of AGAM-SGD on ResNet-50/CIFAR-100 by parameter group, averaged over the tensors of each group. Under warmup + cosine decay (left), normalization parameters call for the least damping early in training (up to 0.79–0.80), convolutional weights and the head bias sit near 0.75, and the head weight stays close to γ_∞ throughout. The groups converge after roughly epoch 100 and cross below γ_∞ together. Without warmup (right), the first five epochs push the groups in opposite directions: convolutional weights peak at 0.81 while normalization parameters fall to 0.67–0.69. The ordering reverses by epoch 10, and at a constant learning rate the groups never fully merge: normalization parameters and the head bias remain near 0.73, while convolutional and head weights settle onto γ_∞.

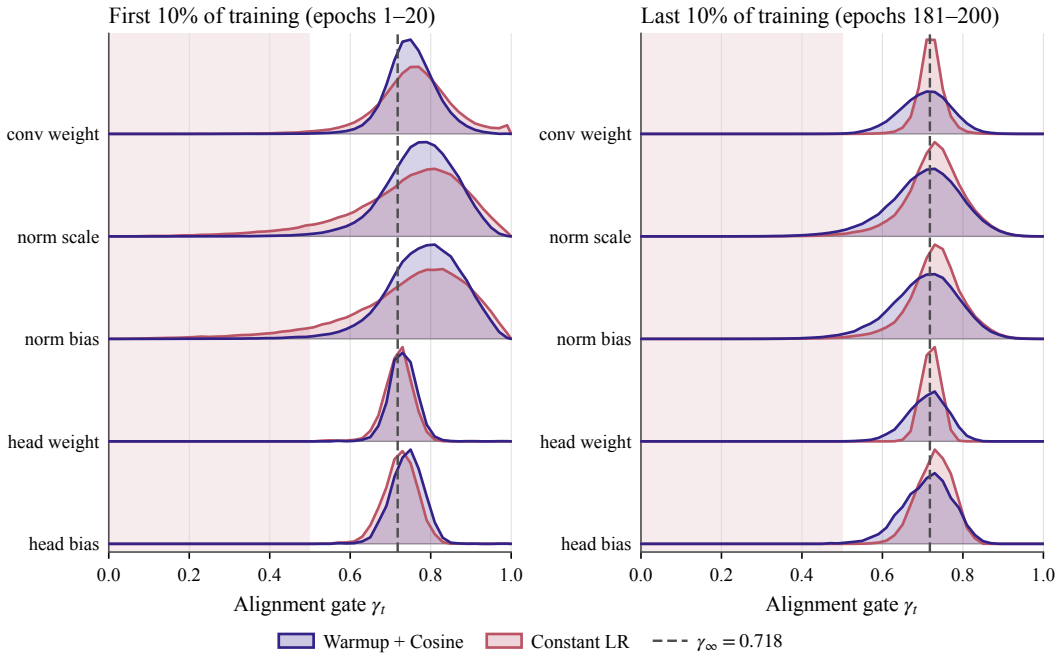


Figure 8: Distribution of the AGAM-SGD gate by parameter group on ResNet-50/CIFAR-100, over all tensor-steps in the first (epochs 1–20) and last (epochs 181–200) 10% of training. The shaded region $\gamma_t^{(k)} < 0.5$ contains the steps on which the gradient opposes the proposed heavy-ball update, and the dashed line marks γ_∞. Early in training the normalization parameters have the widest distributions and, without warmup, a long tail into the shaded region, whereas the head tensors are already concentrated around γ_∞. By the end of training every group is centered on γ_∞, more tightly under a constant learning rate than under cosine decay, where the distributions broaden toward lower gates.

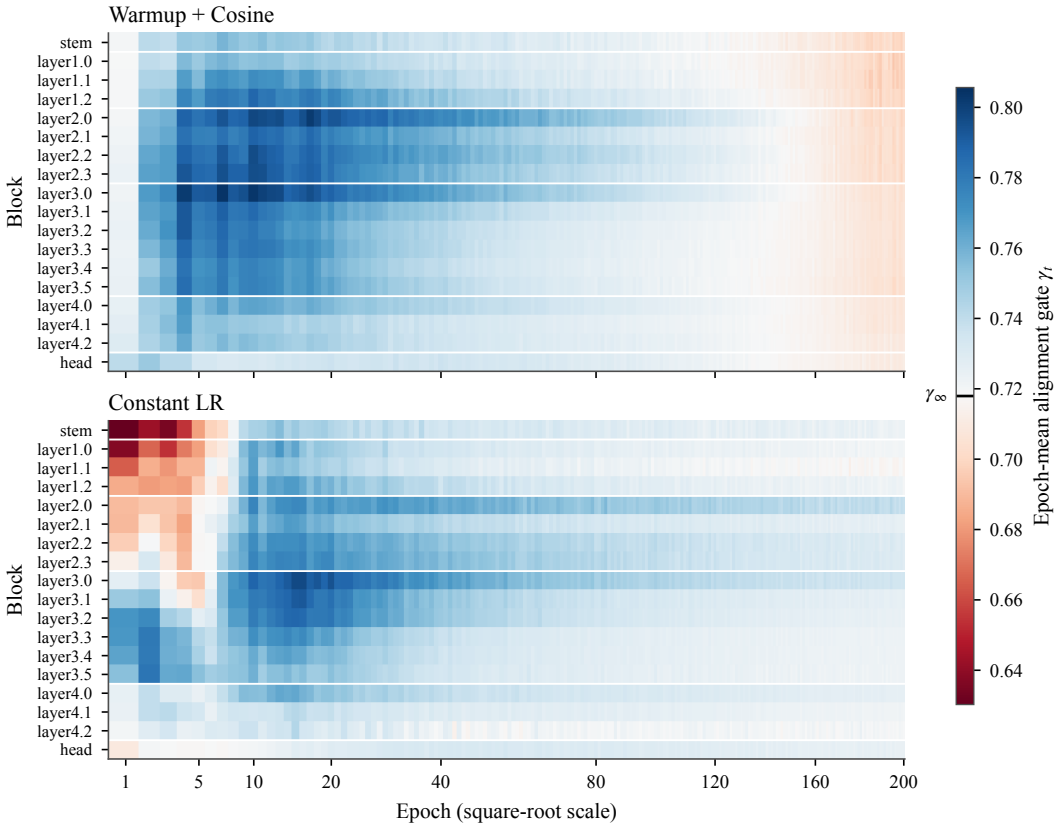


Figure 9: Epoch-mean AGAM-SGD gate per residual block on ResNet-50/CIFAR-100, averaged over the tensors of each block, on a diverging scale centered at γ_∞ (blue for $\gamma_t > \gamma_\infty$ and red for $\gamma_t < \gamma_\infty$). White lines separate the four stages. Under warmup + cosine decay (top), the whole network sits above the reference level for roughly the first forty epochs, most strongly in the first block of stages 2 and 3 (`layer2.0`, `layer3.0`), and drifts below it in the final third of training, starting from the stem and first stage. Without warmup (bottom), the stem and first stage start well below the reference level ($\gamma_t \approx 0.63$–0.67) while the third stage starts above it. This pattern reverses within about eight epochs, after which `layer2.0` and `layer3.0` stay above the reference level for the rest of training.

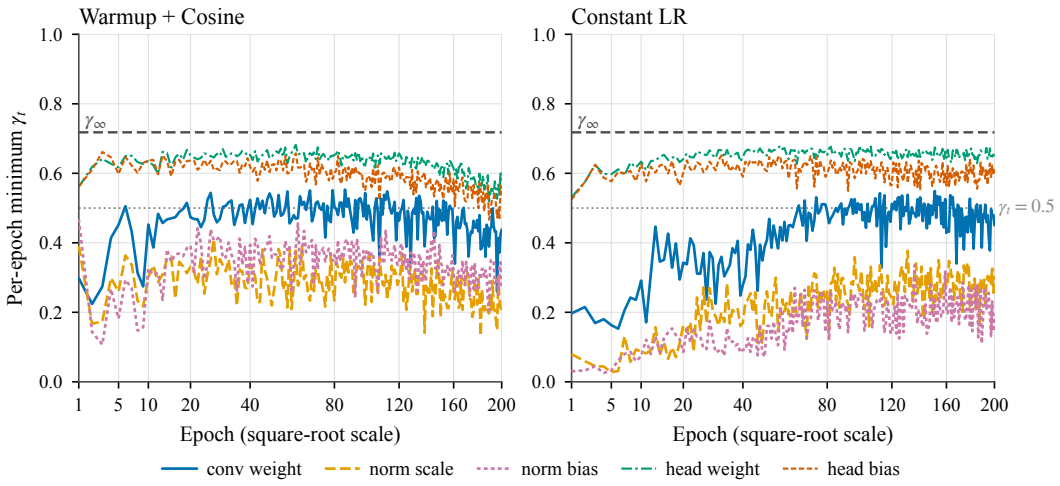


Figure 10: Per-epoch minimum of $\gamma_t^{(k)}$ over the tensor-steps of each parameter group on ResNet-50/CIFAR-100. The dashed line is γ_∞, and the dotted line marks $\gamma_t = 0.5$, below which the gradient opposes the proposed update and the gate removes more than half of the momentum. The head weight never falls below 0.5, and the head bias does so only at the end of the decayed run. For convolutional weights, the minimum falls below 0.5 in most epochs, and normalization parameters are damped hardest, with minima near 0.03 in the first epochs without warmup and between 0.1 and 0.4 for most of training. Even where the mean gate sits on the reference level (Figure 6), individual tensor-steps are therefore gated close to plain SGD.

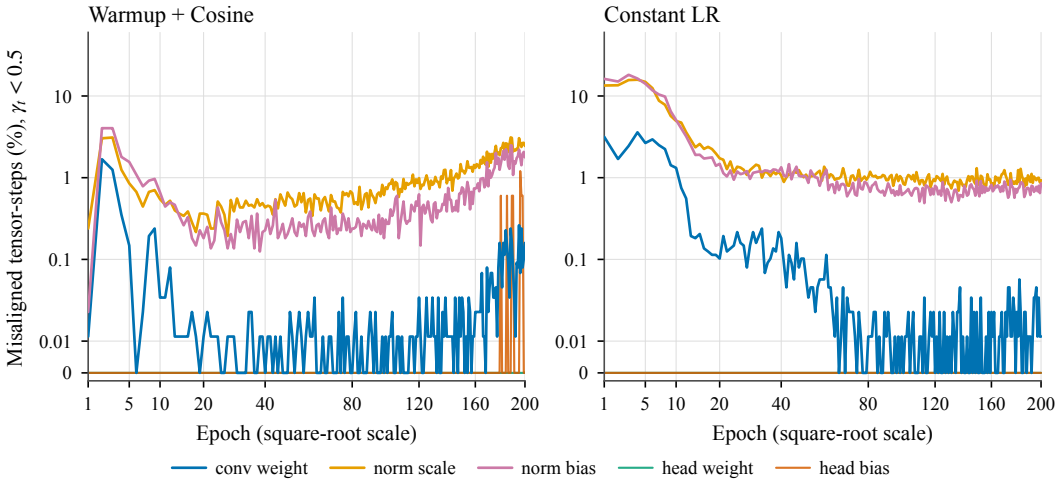


Figure 11: Share of misaligned tensor-steps ($\gamma_t^{(k)} < 0.5$) of AGAM-SGD on ResNet-50/CIFAR-100 per epoch and parameter group, on a symmetric-log axis that is linear below 0.01%, so a plotted zero is exact. Normalization scales and biases carry most of the misalignment. Without warmup they start at 13–18% and settle near 1%. With warmup they stay below 5%, reach their minimum (about 0.2%) around epochs 20–30, and then rise to about 2% as the learning rate anneals, the late rise anticipated in Section 3.5. Convolutional weights stay at or below about 0.2% after the first ten epochs, and the head weight never misaligns.

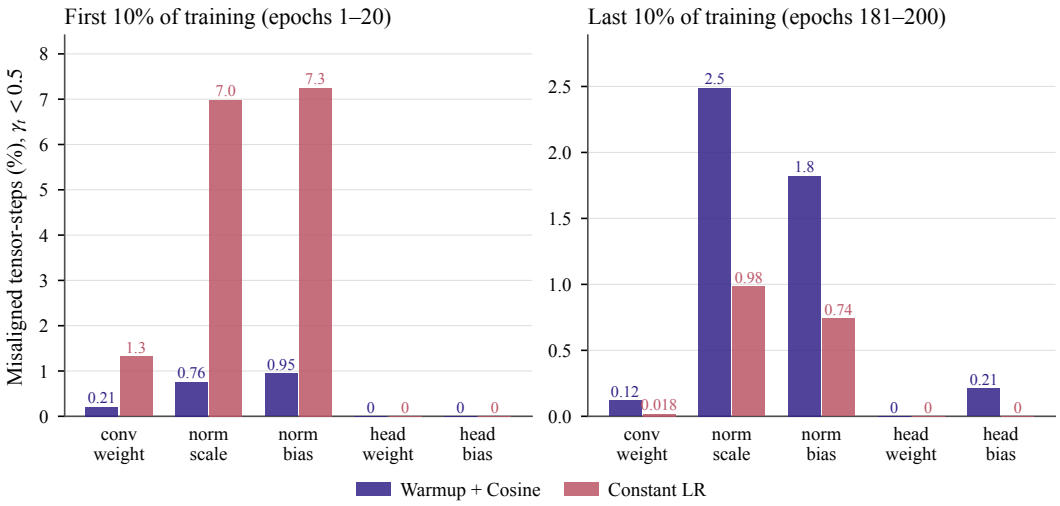


Figure 12: Share of misaligned tensor-steps ($\gamma_t^{(k)} < 0.5$) per parameter group on ResNet-50/CIFAR-100 in the first (epochs 1–20) and last (epochs 181–200) 10% of training. Early on, removing warmup raises misalignment six- to nine-fold, most of it in the normalization parameters (7.0% and 7.3%, against 0.76% and 0.95% with warmup). Late in training the ordering reverses: cosine decay produces more misalignment than a constant learning rate in every group except the head weight, with normalization scales reaching 2.5%. The head weight is never misaligned in either window. The late rise under the decayed schedule is the regime anticipated in Section 3.5: as gradients shrink relative to the buffer, $\beta\|\boldsymbol{m}_{t-1}\|$ can exceed $\|\boldsymbol{g}_t\|$ and misalignment becomes reachable.

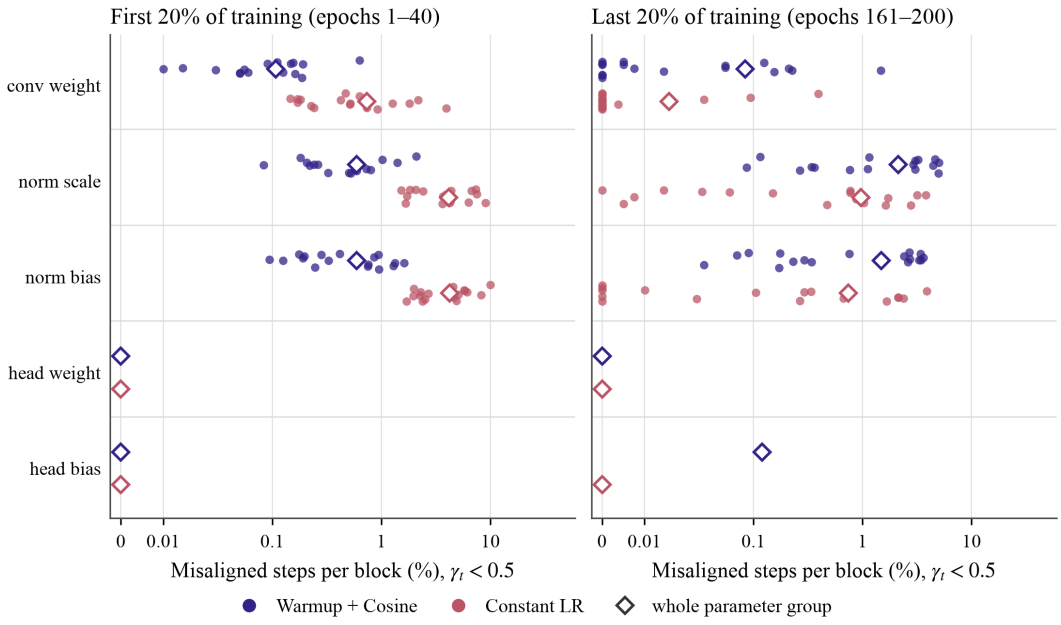


Figure 13: Per-block view of Figure 12 over wider windows, the first (epochs 1–40) and last (epochs 161–200) 20% of training. Each dot is one residual block within a parameter group (convolution dots pool the three or four convolutions of a bottleneck block). Diamonds give the whole group, and the head weight and bias are single tensors. The horizontal axis is linear below 0.01%. Early in training the constant-rate run lies about an order of magnitude to the right of the decayed run in every multi-block group. Late in training, blocks of the same group range from zero to several percent: under a constant learning rate most convolutional blocks are never misaligned, while under cosine decay several normalization layers misalign on 3–5% of steps.

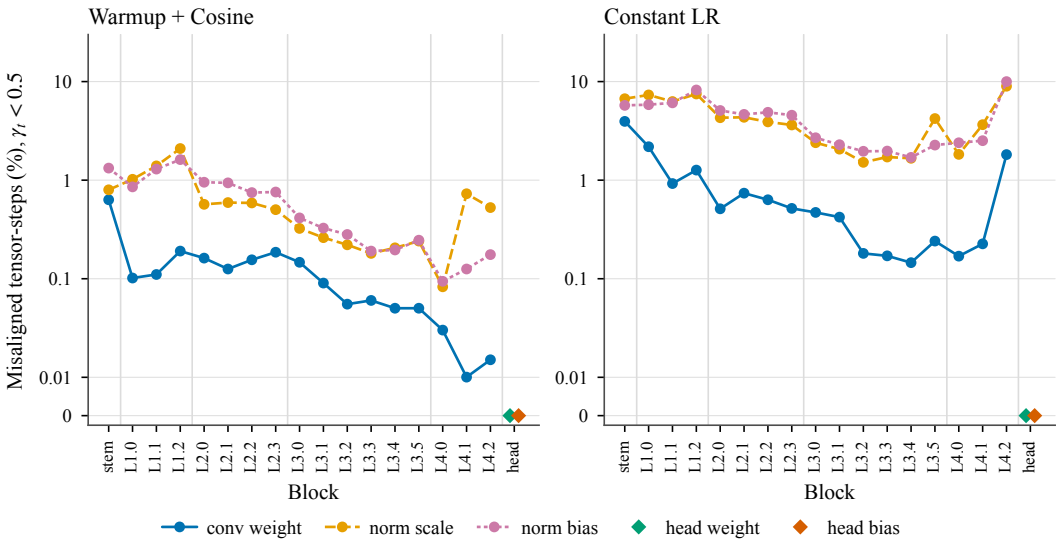


Figure 14: Misaligned tensor-steps ($\gamma_t^{(k)} < 0.5$) per residual block and parameter group on ResNet-50/CIFAR-100 in the first 20% of training (epochs 1–40). Without warmup (right), misalignment concentrates at both ends of the network: convolutional misalignment is highest in the stem (3.9%), lowest in the third stage (down to 0.15%), and rises again to 1.8% in the last block, while normalization parameters reach 10% in `layer4.2`. Warmup (left) lowers every rate, to 0.01–0.63% for convolutional weights and 0.08–2.1% for normalization parameters, and the convolutional rate then decreases steadily with depth. Neither head tensor misaligns in this window.

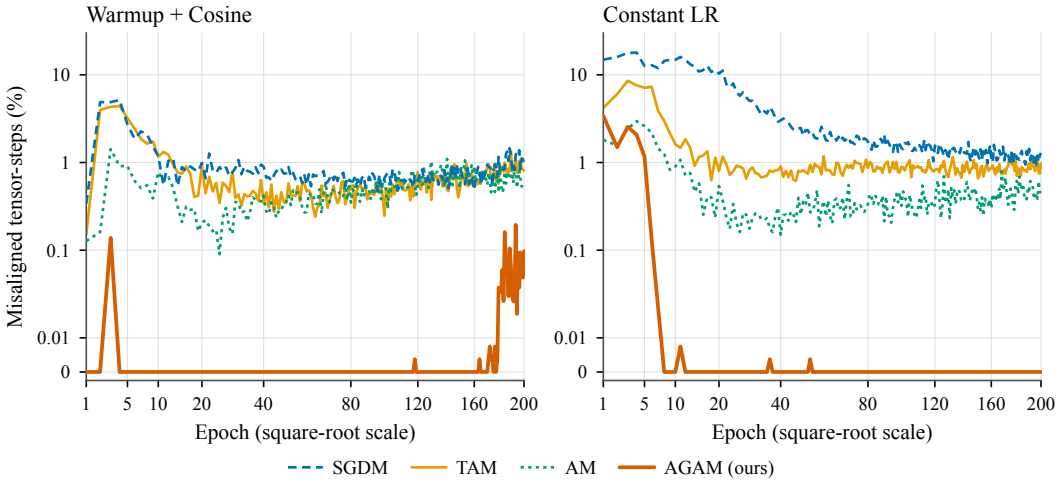


Figure 15: Misalignment between the gradient and the update each optimizer applies on ResNet-50/CIFAR-100: the share of tensor-steps with $\langle \boldsymbol{g}_t, \boldsymbol{u}_t \rangle < 0$, where $\boldsymbol{u}_t$ is the momentum buffer for SGDM, the modified momentum directions for TAM and AM, and the gated update $\boldsymbol{m}_{\text{eff},t}$ for AGAM-SGD. Each optimizer follows its own trajectory under the same schedule. SGDM applies an update that opposes the current gradient on up to 5% of tensor-steps with warmup and up to 18% without it. TAM and AM reduce this rate but do not remove it. AGAM's rate is exactly zero in most epochs of both runs. Its nonzero values are confined to the first eight epochs without warmup (up to 3.5%) and to isolated epochs, mostly near the end of cosine decay. When the alignment norm floor is inactive, Lemma D.3 states that $\langle \boldsymbol{g}_t, \boldsymbol{m}_{\text{eff},t} \rangle < 0$ requires both $s_t < 0$ and $\|\boldsymbol{m}_t\| > (3 + 2\sqrt{2})\,\|\boldsymbol{g}_t\|$.

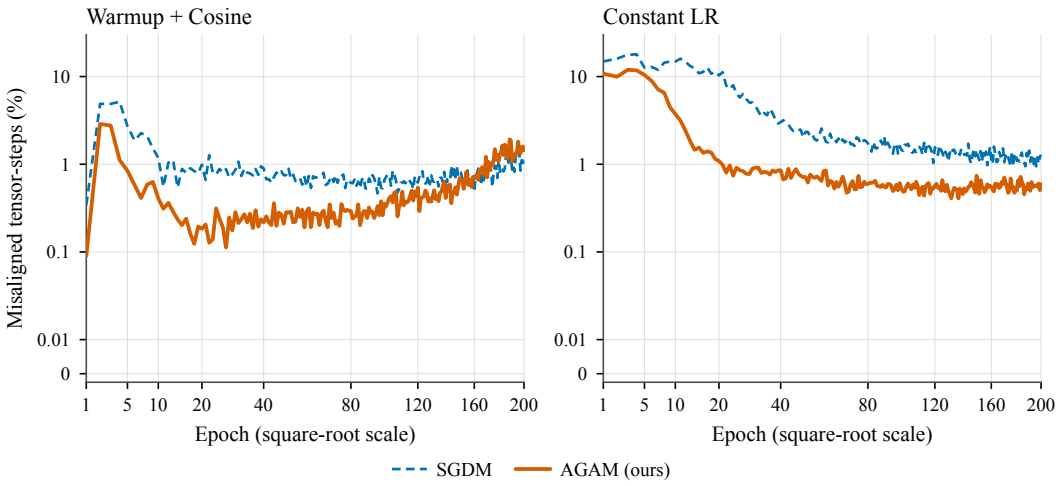


Figure 16: Misalignment between the gradient and the heavy-ball momentum buffer $\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \boldsymbol{g}_t$ for SGDM and AGAM-SGD on ResNet-50/CIFAR-100. Both optimizers maintain this buffer by the same recursion, because AGAM gates only the transient update and never writes the gated momentum back into the optimizer state. The comparison therefore reflects the trajectory AGAM follows, not a change to the stored momentum. For AGAM-SGD, $\boldsymbol{m}_t$ is also the proposed update, so its curve is the share of tensor-steps with $\gamma_t^{(k)} < 0.5$. Together with Figure 15, this isolates the effect of the gate: on the steps where AGAM's proposed update conflicts with the gradient, the gated update it applies almost never does. AGAM's buffer conflicts with the gradient less often than SGDM's for most of training: about 0.2–0.5% against 0.6–1% in the middle of the decayed run, and roughly half of SGDM's rate throughout the constant-rate run. The exception is the last $\sim$40 epochs of cosine decay, where AGAM's rate rises above SGDM's.

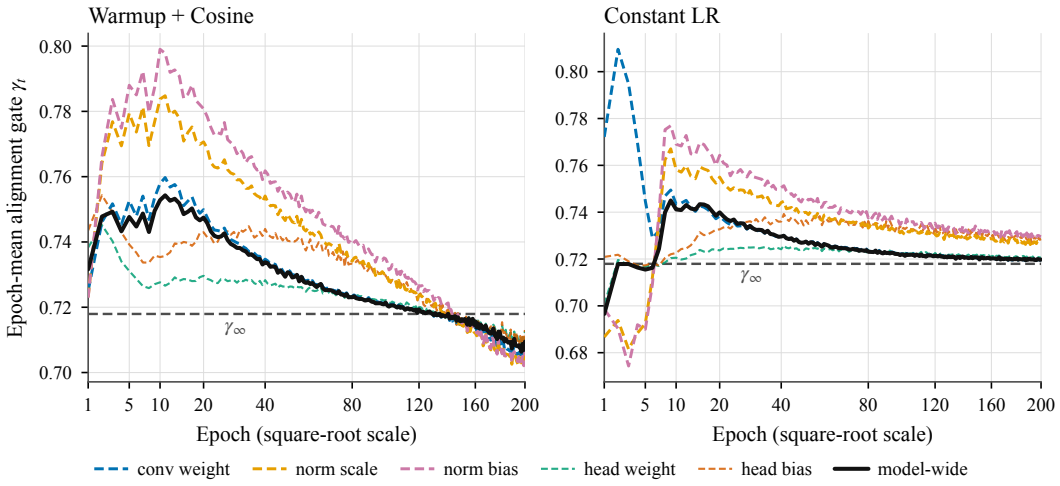


Figure 17: Ablation of gate granularity on ResNet-50/CIFAR-100. The network is trained with a single model-wide gate, computed from the cosine between the concatenated gradient and the concatenated proposed update of all tensors and applied to every tensor (black). At the same steps of the same run we also compute, without applying them, the per-tensor gates that AGAM would have used, and plot their epoch means for each parameter group (dashed). The concatenated cosine is dominated by the tensors with the largest gradient and update norms, so the model-wide gate follows the convolutional weights and largely ignores the other groups: under warmup + cosine decay, normalization parameters call for gates up to 0.78–0.80 while the model-wide gate stays near 0.75. Without warmup the mismatch is sharper. In the first four epochs, convolutional weights call for 0.77–0.81 and normalization parameters for 0.67–0.70, on opposite sides of γ_∞, and the model-wide gate (0.70–0.72) matches neither. The per-group curves closely resemble those of the per-tensor run in Figure 7.

E.3 AGAM-AdamW on MAE ViT pre-training

We record the same statistics for the AGAM-AdamW MAE pre-training run of Section 4.2: 300 epochs, of which the first 15 are warmup (shaded in the plots), with a 12-block encoder and a 4-block decoder. Tensors are grouped by module (attention, MLP, normalization, patch embedding together with the decoder head, and the learned special tokens), and each module is further split into weights, including normalization scales, and biases. Here $\boldsymbol{u}_t$ is the preconditioned update $\mathbf{D}_t^{-1}\hat{\boldsymbol{m}}_t$, so equation 17 does not apply directly, and we draw $\gamma_\infty = 0.718$ only as a reference level. Figures 18 and 19 report the gate, Figures 20 and 21 misalignment, and Figure 22 compares per-tensor and model-wide gating.

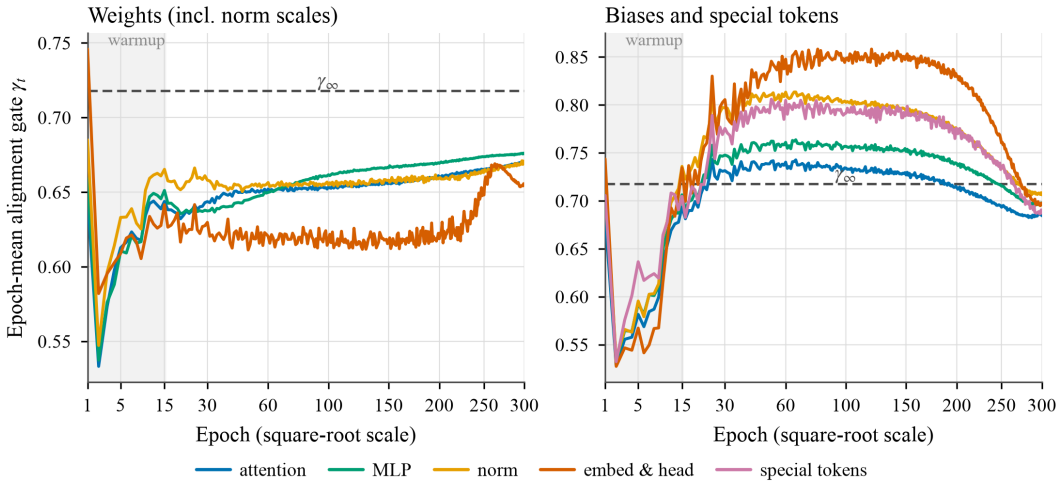


Figure 18: Epoch-mean AGAM-AdamW gate by module during MAE pre-training, for weights (left, normalization scales included) and for biases and special tokens (right). Every group drops to 0.53–0.58 in the second epoch and recovers over the warmup. Weights then stay below the reference level for the entire run: attention, MLP and normalization weights rise slowly from about 0.64 to 0.67, and the embedding and head weights stay near 0.62 until about epoch 220. Biases and special tokens behave differently. After warmup they rise well above the reference level, from about 0.74 for attention biases to 0.85 for the embedding and head biases, and all of them fall below it again over the last hundred epochs. The same optimizer thus applies systematically different amounts of momentum to the weights and biases of a single module, which a shared gate could not represent.

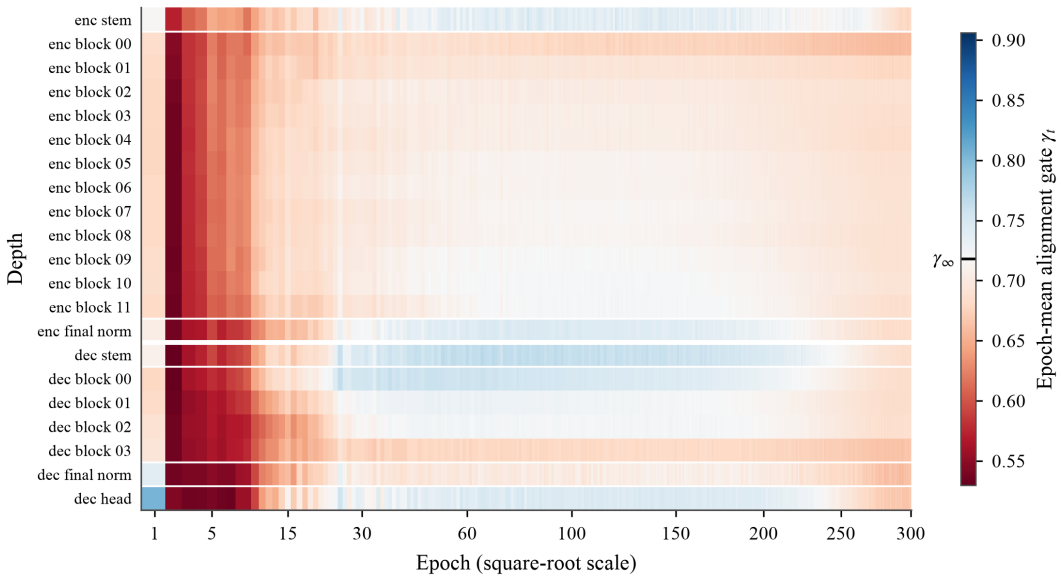


Figure 19: Epoch-mean AGAM-AdamW gate per depth position during MAE pre-training, on a diverging scale centered at γ_∞. White lines separate the encoder stem, encoder blocks, encoder final normalization, decoder stem, decoder blocks, decoder final normalization, and prediction head. Every position is strongly damped during warmup ($\gamma_t \approx 0.53$–0.62 for epochs 2–10). Afterwards the encoder blocks remain just below the reference level for the rest of training, whereas the entry and exit points of each stack (encoder stem, encoder final normalization, decoder stem, first decoder block, and prediction head) sit above it from roughly epoch 30 to epoch 200. The first encoder block and the last decoder block are the most damped transformer blocks throughout.

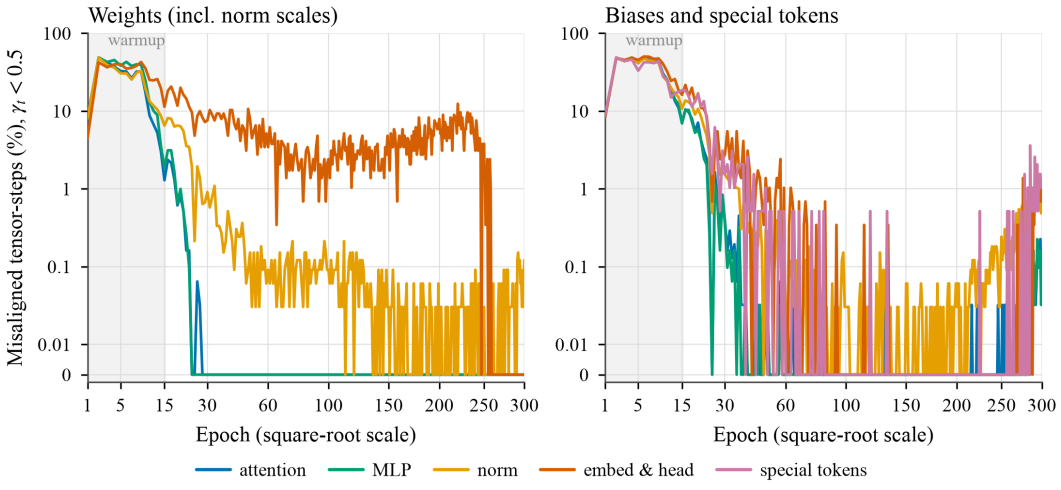


Figure 20: Share of misaligned tensor-steps ($\gamma_t^{(k)} < 0.5$) by module during MAE pre-training, for weights (left) and for biases and special tokens (right). Between epochs 2 and 10 the proposed update opposes the gradient on 30–50% of tensor-steps in every group. After warmup, attention and MLP weights stop misaligning altogether by epoch 30, and normalization weights settle at or below 0.2%. The embedding and head weights are the exception: they remain misaligned on 1–10% of steps until about epoch 250 and then drop to zero. Biases and special tokens fall to sporadic, mostly zero rates by epoch 60 and rise again to 1–3% over the final fifty epochs.

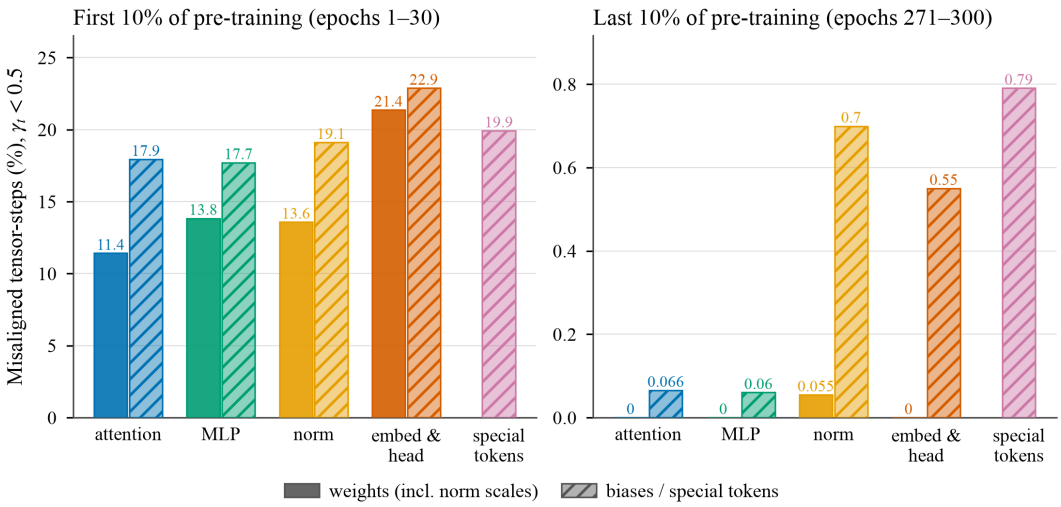


Figure 21: Share of misaligned tensor-steps by module in the first (epochs 1–30) and last (epochs 271–300) 10% of MAE pre-training. Solid bars are weights (including normalization scales) and hatched bars are biases and special tokens. Early in training every group is misaligned on 11–23% of steps, and in every module the biases are misaligned more often than the weights. By the end, attention, MLP, and embedding/head weights are never misaligned and normalization weights only on 0.055% of steps, whereas biases and special tokens still misalign on 0.06–0.79%.

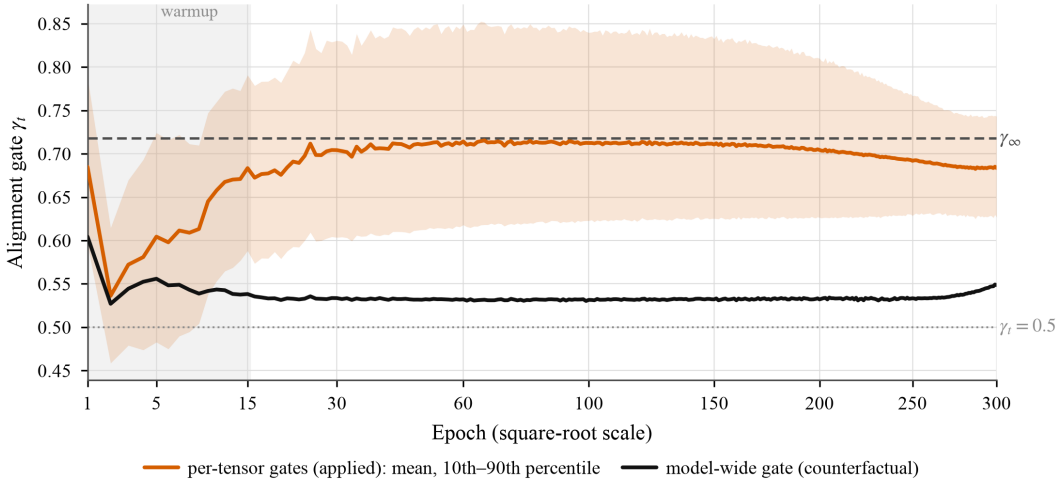


Figure 22: Per-tensor versus model-wide gating in AGAM-AdamW MAE pre-training. The orange line is the mean of the per-tensor gates applied during training, and the band covers their 10th to 90th percentiles. The black line is the model-wide gate that a single cosine between the concatenated gradient and proposed update would have produced at the same steps. It is computed but not applied. After warmup the per-tensor mean stays near 0.71 and declines to 0.68 by the end of training, while the model-wide gate remains at 0.53–0.55, below the 10th percentile of the per-tensor gates from about epoch 12 onward. A model-wide gate would therefore have held every tensor at $\beta_{\text{eff}} \approx 0.48$ after warmup, damping most tensors far more than their own alignment calls for.

F Proofs and Exact Representations

We suppress the tensor index in blockwise identities. The convergence proof uses the full parameter vector and the filtration defined in Appendix D. The gate may depend on the current gradient, whereas the pre-step state is measurable before that gradient is sampled.

F.1 Exact representations and initialization

AGAM-SGD. With the undampened buffer $\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \boldsymbol{g}_t$, the identity $\boldsymbol{u}_{\text{eff},t} = (1-\gamma_t)\boldsymbol{g}_t + \gamma_t \boldsymbol{m}_t$ follows directly by substituting $\boldsymbol{m}_t = \beta \boldsymbol{m}_{t-1} + \boldsymbol{g}_t$ into the right-hand side. For the normalized state $\boldsymbol{z}_t = (1-\beta)\boldsymbol{m}_t$, matching the coefficients of $\boldsymbol{g}_t$ and $\boldsymbol{z}_t$ in $\boldsymbol{u}_{\text{eff},t} = k_t\{(1-\nu_t)\boldsymbol{g}_t + \nu_t \boldsymbol{z}_t\}$ gives $k_t(1-\nu_t) = 1-\gamma_t$ and $k_t\nu_t = \gamma_t/(1-\beta)$, hence $k_t = (1-\beta+\beta\gamma_t)/(1-\beta)$ and $\nu_t = \gamma_t/(1-\beta+\beta\gamma_t)$, which is Remark D.2. The normalized state obeys $\boldsymbol{z}_t = \beta \boldsymbol{z}_{t-1} + (1-\beta)\boldsymbol{g}_t$ with $\boldsymbol{z}_0 = \boldsymbol{0}$.

AGAM-Adam. Before step t the stored first-moment mass is $1-\beta_1^{t-1}$. With $a_t = \beta_1\gamma_t$ and complementary gradient weight $1-a_t$, the applied mass is

$$a_t\big(1-\beta_1^{t-1}\big) + 1 - a_t = 1 - \gamma_t\beta_1^t > 0,$$

which is equation 5. Substituting $\boldsymbol{m}_t = (1-\beta_1^t)\hat{\boldsymbol{m}}_t$ into $\gamma_t \boldsymbol{m}_t + (1-\gamma_t)\boldsymbol{g}_t$ and dividing by that mass proves equation 7 with $\nu_t = \gamma_t(1-\beta_1^t)/(1-\gamma_t\beta_1^t)$. The normalized Adam state $\boldsymbol{z}_t = \hat{\boldsymbol{m}}_t$ obeys

$$\boldsymbol{z}_t = \rho_t \boldsymbol{z}_{t-1} + (1-\rho_t)\boldsymbol{g}_t, \qquad \rho_t = \frac{\beta_1(1-\beta_1^{t-1})}{1-\beta_1^t} \le \beta_1, \tag{21}$$

with $\rho_1 = 0$, so $\boldsymbol{z}_1 = \boldsymbol{g}_1$. Direct subtraction gives

$$\nu_t - \gamma_t = -\frac{\gamma_t(1-\gamma_t)\beta_1^t}{1-\gamma_t\beta_1^t},$$

so $0 \le \nu_t \le \gamma_t \le 1$ and $\nu_t - \gamma_t \to 0$. Thus $\boldsymbol{e}_1 = \boldsymbol{\xi}_1$. For SGD, $\boldsymbol{e}_1 = -\beta\nabla f(\boldsymbol{\theta}_1) + (1-\beta)\boldsymbol{\xi}_1$, hence

$$\mathbb{E}\|\boldsymbol{e}_1\|^2 \le \beta^2\|\nabla f(\boldsymbol{\theta}_1)\|^2 + (1-\beta)^2\sigma_1^2.$$

Both identities assume a shared per-step clock and zero initialization. Missing gradients, Nesterov updates, norm matching, recursive gating, and alternative fresh-gradient weights are outside the stated configuration.

Degenerate cases. The implementation clamps each alignment norm at 10^{-8} and clamps the reported cosine to $[-1,1]$. The algebra and the convergence result require only $\gamma_t \in [0,1]$ and are unaffected by this floor in exact arithmetic. The angle-specific score of Lemma D.3 and the Gaussian limits of Proposition D.5 use the ideal cosine, for which the floor is inactive. For SGD and Adam, a zero probe with a nonzero gradient gives gate $1/2$, and a zero gradient uses the base memory coefficient, represented here by $\gamma_t = 1$. Lion uses the base coefficient if either vector is zero. For example, with the SGD norm floor active, $g = 10^{-12}$ and $m = -2\cdot 10^{-12}$ give a gate close to $1/2$ and a negative score despite a norm ratio of two. Thus the sharp ideal-cosine threshold is not asserted in this numerical regime.

F.2 Proof of Lemma D.3

By Proposition D.1, $\langle \boldsymbol{g}_t, \boldsymbol{u}_{\text{eff},t}\rangle = (1-\gamma_t)\|\boldsymbol{g}_t\|^2 + \gamma_t\langle \boldsymbol{g}_t, \boldsymbol{m}_t\rangle = (1-\gamma_t)\|\boldsymbol{g}_t\|^2 + \gamma_t s_t\|\boldsymbol{g}_t\|\|\boldsymbol{m}_t\|$. Substituting $\gamma_t = (1+s_t)/2$ and $\rho_t = \|\boldsymbol{m}_t\|/\|\boldsymbol{g}_t\|$ gives equation 14.

For $s_t \ge 0$ both bracketed terms are non-negative. At $s_t = -1$ the score equals $\|\boldsymbol{g}_t\|^2$. For $-1 < s_t < 0$ the bracket is non-negative iff $\rho_t \le h(s_t) := (1-s_t)/\{-s_t(1+s_t)\}$. Writing $a = -s_t \in (0,1)$, $h = (1+a)/\{a(1-a)\}$ and $h'(a) = (a^2+2a-1)/\{a(1-a)\}^2$, whose unique root in $(0,1)$ is $a = \sqrt{2}-1$. This is a minimum, with $h = 3+2\sqrt{2}$. Hence $\rho_t \le 3+2\sqrt{2}$ guarantees a non-negative score at *every* angle, and at $s_t = 1-\sqrt{2}$ equality gives a zero score while any larger ρ_t gives a negative one, so the constant is sharp as a uniform condition. At $s_t = -1$, $\gamma_t = 0$ and $\boldsymbol{u}_{\text{eff},t} = \boldsymbol{g}_t$.

Finally $\gamma_t s_t = s_t(1+s_t)/2$ is a parabola in s_t minimized at $s_t = -\frac{1}{2}$, where it equals $-\frac{1}{8}$.

A related geometric statement, for nonzero $\boldsymbol{g}_t, \boldsymbol{m}_t, \boldsymbol{u}_{\text{eff},t}$, is that $\operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{u}_{\text{eff},t}) \geq \operatorname{cossim}(\boldsymbol{g}_t, \boldsymbol{m}_t)$: a positive combination $(1-\gamma_t)\boldsymbol{g}_t + \gamma_t \boldsymbol{m}_t$ lies in the cone spanned by $\boldsymbol{g}_t$ and $\boldsymbol{m}_t$, so adding a non-negative multiple of $\boldsymbol{g}_t$ to a positive multiple of $\boldsymbol{m}_t$ cannot increase the polar angle from $\boldsymbol{g}_t$ in $[0, \pi]$. Gating therefore never worsens the angle to the current gradient. *Angle improvement is not descent*: an obtuse angle can remain obtuse, which is exactly what the threshold quantifies. For AGAM-Adam, linearity of $\mathbf{D}_t^{-1}$ gives $\boldsymbol{u}_{\text{eff},t} = (1-\nu_t)\mathbf{D}_t^{-1}\boldsymbol{g}_t + \nu_t \mathbf{D}_t^{-1}\hat{\boldsymbol{m}}_t$ and hence the same cone property relative to $\mathbf{D}_t^{-1}\boldsymbol{g}_t$. The directional score and its positivity condition are given by equation 15.

F.3 One-step interpolation

Lemma F.1. *Consider a single parameter block with all other blocks held fixed. Let f denote the resulting L-smooth objective, let $\Phi_t^{\text{SGD}} = f(\boldsymbol{\theta}_t) - \eta\langle \nabla f(\boldsymbol{\theta}_t), \boldsymbol{g}_t\rangle + \frac{L\eta^2}{2}\|\boldsymbol{g}_t\|^2$, and let Φ_t^{SHB} be the same expression with $\boldsymbol{m}_t$ in place of $\boldsymbol{g}_t$ (these are the one-step descent-lemma guarantees of SGD and of stochastic heavy ball). Then*

$$f(\boldsymbol{\theta}_t - \eta \boldsymbol{u}_{\mathit{eff},t}) \leq (1-\gamma_t)\,\Phi_t^{\text{SGD}} + \gamma_t\,\Phi_t^{\text{SHB}}.$$

Proof. By L-smoothness, $f(\boldsymbol{\theta}_t - \eta \boldsymbol{d}) \leq f(\boldsymbol{\theta}_t) - \eta\langle \nabla f(\boldsymbol{\theta}_t), \boldsymbol{d}\rangle + \frac{L\eta^2}{2}\|\boldsymbol{d}\|^2$ for any $\boldsymbol{d}$. Apply this with $\boldsymbol{d} = \boldsymbol{u}_{\text{eff},t} = (1-\gamma_t)\boldsymbol{g}_t + \gamma_t \boldsymbol{m}_t$. The inner product is linear in $\boldsymbol{d}$ and $\|\cdot\|^2$ is convex, so $\|\boldsymbol{u}_{\text{eff},t}\|^2 \leq (1-\gamma_t)\|\boldsymbol{g}_t\|^2 + \gamma_t\|\boldsymbol{m}_t\|^2$. Collecting terms gives the claim. □

This is a pathwise upper bound on the next objective value. It need not be below either endpoint bound, and it does not identify the cosine gate as the minimizer of that bound. The result is stated for one block, and the full-vector convergence analysis separately accounts for simultaneous block updates.

F.4 Proof of Proposition D.5 and the finite-dimensional spread

In the stationary local model $\boldsymbol{g}_t = \boldsymbol{\mu} + \boldsymbol{\xi}_t$ with $\boldsymbol{\xi}_t$ i.i.d., zero-mean, covariance $\boldsymbol{\Sigma}$, the equilibrated buffer is $\boldsymbol{m}_t = \boldsymbol{\mu}/(1-\beta) + \sum_{j\geq 0}\beta^j \boldsymbol{\xi}_{t-j}$. Independence and zero mean give $\operatorname{Cov}(\boldsymbol{g}_t, \boldsymbol{m}_t) = \boldsymbol{\Sigma}$ and $\operatorname{Cov}(\boldsymbol{m}_t) = \boldsymbol{\Sigma}/(1-\beta^2)$, hence

$$\mathbb{E}\langle \boldsymbol{g}_t, \boldsymbol{m}_t\rangle = \frac{\|\boldsymbol{\mu}\|^2}{1-\beta} + \operatorname{tr}\boldsymbol{\Sigma}, \quad \mathbb{E}\|\boldsymbol{g}_t\|^2 = \|\boldsymbol{\mu}\|^2 + \operatorname{tr}\boldsymbol{\Sigma}, \quad \mathbb{E}\|\boldsymbol{m}_t\|^2 = \frac{\|\boldsymbol{\mu}\|^2}{(1-\beta)^2} + \frac{\operatorname{tr}\boldsymbol{\Sigma}}{1-\beta^2}. \tag{22}$$

For isotropic Gaussian noise in a sequence of dimensions $d \to \infty$ with $\kappa = \|\boldsymbol{\mu}\|^2/(d\sigma^2)$ held fixed, Gaussian quadratic-form concentration and concentration of $\boldsymbol{\mu}^\top \boldsymbol{\xi}_t/d$ imply that the inner product and both squared norms, divided by d, converge in probability to their normalized means. Taking the ratio and writing $\kappa = \|\boldsymbol{\mu}\|^2/\operatorname{tr}\boldsymbol{\Sigma}$,

$$s_\infty = \frac{\kappa/(1-\beta)+1}{\sqrt{(\kappa+1)\,\{\kappa/(1-\beta)^2 + 1/(1-\beta^2)\}}} = \frac{\kappa+1-\beta}{\sqrt{(\kappa+1)\{\kappa + (1-\beta)/(1+\beta)\}}},$$

which is equation 16. Boundedness of the cosine upgrades convergence in probability to convergence of the expectation. At $\kappa = 0$ this is $(1-\beta)\sqrt{(1+\beta)/(1-\beta)} = \sqrt{1-\beta^2}$, giving equation 17, and as $\kappa \to \infty$ it tends to 1. Writing $a = 1-\beta$ and $c = (1-\beta)/(1+\beta)$, differentiation gives

$$s'_\infty(\kappa) = \frac{\beta^2 \kappa}{(1+\beta)[(\kappa+1)(\kappa+c)]^{3/2}},$$

which is positive for $\kappa > 0$. The same Gaussian variance estimates and the smoothness of the ratio near its nonzero limiting norms give the stated $\mathcal{O}_p(d^{-1/2})$ error. For zero-initialized memory at a fixed step t the corresponding limit at $\kappa = 0$ is $\gamma_t \to \frac{1}{2}\{1 + \sqrt{(1-\beta^2)/(1-\beta^{2t})}\}$, which starts at 1 and decreases to equation 17. This is a startup calculation for the frozen Gaussian model, not a prediction of the whole training transient.

Spread at zero signal. For $\boldsymbol{\mu} = 0$, standardize $\boldsymbol{g}_t$ and $\boldsymbol{m}_t$ to X, Y whose coordinate pairs are i.i.d. centered normal with correlation $\varrho = \sqrt{1-\beta^2}$. Applying the multivariate delta method to $(d^{-1}\sum X_i Y_i,\ d^{-1}\sum X_i^2,\ d^{-1}\sum Y_i^2)$ and the map $a/\sqrt{bc}$, Gaussian fourth moments give asymptotic cosine variance $(1-\varrho^2)^2/d$. Since $\gamma_t = (1+s_t)/2$, $\mathrm{Var}(\gamma_t) = \beta^4/(4d) + o(d^{-1})$.

Finite-sample check. For $\beta = 0.9$ and 8,000 independent draws per dimension, the following simulated means approach the limiting value 0.71794 and illustrate the leading-order spread. A finite-dimensional mean can lie below that limit:

d	mean gate	gate s.d.	leading-order s.d. $\beta^2/(2\sqrt{d})$
16	0.71100	0.10446	0.10125
128	0.71672	0.03588	0.03580
1024	0.71798	0.01270	0.01266

Correlated coordinates, anisotropy, a non-stationary mean, and Adam's gradient-dependent preconditioner can all invalidate the isotropic prediction. The formula is not a calibration of observed tensor dimensionality, and we do not use it to predict AGAM-Adam gates.

G Proof and Consequences of the Stationarity Bound

Proof of Theorem D.6. Write $\boldsymbol{F}_t = \nabla f(\boldsymbol{\theta}_t)$ and $\Delta = f(\boldsymbol{\theta}_1) - f^\star$. The exact common representation gives

$$\boldsymbol{u}_{\mathrm{eff},t} = \boldsymbol{H}_t(\boldsymbol{F}_t + \boldsymbol{w}_t), \qquad \boldsymbol{w}_t = (\boldsymbol{I} - \boldsymbol{N}_t)\boldsymbol{\xi}_t + \boldsymbol{N}_t \boldsymbol{e}_t.$$

Since each block of $\boldsymbol{N}_t$ is a scalar in $[0,1]$, convexity of the squared norm within each block gives the pointwise inequality

$$\|\boldsymbol{w}_t\|^2 \le \|\boldsymbol{\xi}_t\|^2 + \|\boldsymbol{e}_t\|^2. \tag{23}$$

No expectation is factored across the gate and the noise.

Let $b = 3h_+^2/(4h_-)$. Smoothness, the spectral bounds, and Young's inequality imply

$$\begin{aligned} f(\boldsymbol{\theta}_{t+1}) - f(\boldsymbol{\theta}_t) &\le -\eta h_- \|\boldsymbol{F}_t\|^2 + \eta h_+ \|\boldsymbol{F}_t\|\|\boldsymbol{w}_t\| + L\eta^2 h_+^2(\|\boldsymbol{F}_t\|^2 + \|\boldsymbol{w}_t\|^2) \\ &\le -\eta(h_-/2 - L\eta h_+^2)\|\boldsymbol{F}_t\|^2 + \eta\{h_+^2/(2h_-) + L\eta h_+^2\}\|\boldsymbol{w}_t\|^2 \\ &\le -\frac{\eta h_-}{4}\|\boldsymbol{F}_t\|^2 + \eta b \|\boldsymbol{w}_t\|^2, \end{aligned} \tag{24}$$

where the last line uses the first restriction in equation 19. Also,

$$\|\boldsymbol{u}_{\mathrm{eff},t}\|^2 \le 2h_+^2(\|\boldsymbol{F}_t\|^2 + \|\boldsymbol{w}_t\|^2). \tag{25}$$

For $t \ge 2$, the normalized stored state satisfies $\boldsymbol{z}_t = \rho_t \boldsymbol{z}_{t-1} + (1-\rho_t)\boldsymbol{g}_t$ with deterministic $0 \le \rho_t \le \beta$. For SGD, $\rho_t = \beta$, and for Adam it is given by equation 21. Hence

$$\boldsymbol{e}_t = \rho_t(\boldsymbol{e}_{t-1} + \boldsymbol{F}_{t-1} - \boldsymbol{F}_t) + (1-\rho_t)\boldsymbol{\xi}_t.$$

The first term is $\mathcal{F}_{t-1}$-measurable, so conditional unbiasedness eliminates its cross term with $\boldsymbol{\xi}_t$. Using $\rho^2\|a+c\|^2 \le \rho\|a\|^2 + \rho^2\|c\|^2/(1-\rho)$ and $\|\boldsymbol{F}_t - \boldsymbol{F}_{t-1}\| \le L\eta\|\boldsymbol{u}_{\mathrm{eff},t-1}\|$ gives

$$\mathbb{E}\|\boldsymbol{e}_t\|^2 \le \beta\mathbb{E}\|\boldsymbol{e}_{t-1}\|^2 + \frac{\beta^2 L^2\eta^2}{1-\beta}\mathbb{E}\|\boldsymbol{u}_{\mathrm{eff},t-1}\|^2 + \sigma_t^2. \tag{26}$$

The noise coefficient $(1-\rho_t)^2$ has been bounded by one. This cancellation uses the fixed stored recurrence, not independence of the gate. Predictability of past state also holds for ordinary recursively gated optimizers, so it is not a property exclusive to state preservation.

Define the finite-horizon sums

$$S_T = \sum_{t=1}^{T}\mathbb{E}\|\boldsymbol{F}_t\|^2, \quad E_T = \sum_{t=1}^{T}\mathbb{E}\|\boldsymbol{e}_t\|^2, \quad W_T = \sum_{t=1}^{T}\mathbb{E}\|\boldsymbol{w}_t\|^2, \quad V_T = \sum_{t=1}^{T}\sigma_t^2.$$

Summing equation 26 and dropping nonnegative terminal terms gives

$$E_T \le C_0 + C\eta^2 \sum_{t=1}^{T} \mathbb{E}\|\boldsymbol{u}_{\text{eff},t}\|^2 + \frac{V_T}{1-\beta}, \qquad C_0 = \frac{\mathbb{E}\|\boldsymbol{e}_1\|^2}{1-\beta}, \quad C = \frac{\beta^2 L^2}{(1-\beta)^2}.$$

Combining equation 23 and equation 25, and setting $a = 2C\eta^2 h_+^2$, yields

$$(1-a)W_T \le C_0 + aS_T + \left(1 + \frac{1}{1-\beta}\right) V_T.$$

The second step-size restriction implies $a \le 1/(12\kappa_{\boldsymbol{H}}^2) \le 1/12$. Therefore

$$W_T \le 2C_0 + 2aS_T + 2\left(1 + \frac{1}{1-\beta}\right) V_T.$$

Telescoping equation 24 gives $S_T \le 4\Delta/(\eta h_-) + 3\kappa_{\boldsymbol{H}}^2 W_T$. Substituting the preceding bound and absorbing $6\kappa_{\boldsymbol{H}}^2 a S_T \le S_T/2$ proves equation 20. All expectations are finite at finite horizons under the variance, smoothness, and spectral assumptions: the moment and parameter recurrences give finite second moments inductively. □

Corollary G.1 (Consequences)**.** *Under the assumptions of Theorem D.6, exact gradients give an $\mathcal{O}(1/T)$ average squared-gradient bound and $\|\nabla f(\boldsymbol{\theta}_t)\| \to 0$. If $\bar{\sigma}_T^2 \to 0$, the average expected squared gradient also tends to zero. In particular, $\sigma_t^2 \le C/t$ gives $\mathcal{O}(\log T/T)$ in optimizer iterations. If $\sum_t \sigma_t^2 < \infty$, then $\|\nabla f(\boldsymbol{\theta}_t)\| \to 0$ almost surely. For a fixed batch with $\sigma_t^2 \le \sigma^2/B$, the bound is instead $\mathcal{O}(1/T) + \mathcal{O}(\sigma^2/B)$.*

Proof. For exact gradients, the bound on S_T is uniform in T, so the squared gradient norms are summable. The averaged claims follow by dividing the bound by T. With summable variances, the sum of the expected squared gradient norms is finite, and Tonelli's theorem then gives almost-sure summability. None of these statements asserts convergence of the parameters or convergence to a local minimum. □

Reducing variance by increasing the batch size changes the oracle cost. If independent averaging gives variance at most σ^2/B and all other constants are uniform in B, choosing $B = T$ gives an $\mathcal{O}(1/T)$ bound with $K = T^2$ individual gradient evaluations, equivalently $\mathcal{O}(K^{-1/2})$. This is not a fixed-batch $\mathcal{O}(T^{-1/2})$ guarantee.

Corollary G.2 (Deterministic PL convergence)**.** *Suppose additionally that $\|\nabla f(\boldsymbol{x})\|^2 \ge 2\mu(f(\boldsymbol{x}) - f^\star)$ for $\mu > 0$. Under equation 19 and exact gradients, define*

$$\Phi_t = f(\boldsymbol{\theta}_t) - f^\star + \ell\|\boldsymbol{e}_t\|^2, \qquad \ell = \frac{3\eta h_+^2}{2h_-(1-\beta)}.$$

Then $\Phi_{t+1} \le (1-\chi)\Phi_t$, where $\chi = \min\{\eta h_- \mu/4, (1-\beta)/4\}$.

Proof. Here $\boldsymbol{w}_t = \boldsymbol{N}_t \boldsymbol{e}_t$, so $\|\boldsymbol{w}_t\| \le \|\boldsymbol{e}_t\|$. equation 26 and equation 25 imply

$$\|\boldsymbol{e}_{t+1}\|^2 \le (\beta + d_0)\|\boldsymbol{e}_t\|^2 + d_0\|\boldsymbol{F}_t\|^2, \qquad d_0 = \frac{2\beta^2 L^2 \eta^2 h_+^2}{1-\beta}.$$

The step-size bound gives $d_0 \le (1-\beta)/(12\kappa_{\boldsymbol{H}}^2)$. With $b = 3h_+^2/(4h_-)$ and $\ell = 2\eta b/(1-\beta)$, it follows that $\ell d_0 \le \eta h_-/8$ and $\eta b - \ell(1-\beta-d_0) \le -\eta b/2$. Adding the tracking inequality times ℓ to equation 24 gives

$$\Phi_{t+1} \le \Phi_t - \frac{\eta h_-}{8}\|\boldsymbol{F}_t\|^2 - \frac{\eta b}{2}\|\boldsymbol{e}_t\|^2.$$

The PL inequality and $\eta b/(2\ell) = (1-\beta)/4$ prove the claim. □

The PL assumption is additional structure. The result establishes linear convergence under that assumption, not accelerated complexity for general deep-network training.

G.1 WEIGHT DECAY

Coupled SGD decay on a fixed coordinate selector $\boldsymbol{J} = \boldsymbol{J}^\top = \boldsymbol{J}^2$ supplies the oracle $\boldsymbol{g}_t + \lambda \boldsymbol{J}\boldsymbol{\theta}_t$, with $\lambda \geq 0$. Applying Theorem D.6 to $f_\lambda(\boldsymbol{x}) = f(\boldsymbol{x}) + \lambda \|\boldsymbol{J}\boldsymbol{x}\|^2/2$ gives a stationarity claim for this regularized objective, whose smoothness constant is at most $L + \lambda$. The geometric certificate then concerns the consumed gradient of the regularized mini-batch loss.

Decoupled AdamW decay applies $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\{\boldsymbol{u}_{\text{eff},t} + \lambda \boldsymbol{J}\boldsymbol{\theta}_t\}$. Let $\boldsymbol{\delta}_t = \lambda \boldsymbol{J}\boldsymbol{\theta}_t$ and $D_T = \sum_{t=1}^T \mathbb{E}\|\boldsymbol{H}_t^{-1}\boldsymbol{\delta}_t\|^2$. Under the stricter restriction

$$0 < \eta \leq \min\left\{ \frac{h_-}{4Lh_+^2},\ \frac{(1-\beta)h_-}{\sqrt{48}\,\beta L h_+^2} \right\},$$

the same argument gives

$$\begin{aligned} \frac{1}{T}\sum_{t=1}^T \mathbb{E}\|\nabla f(\boldsymbol{\theta}_t)\|^2 \leq \frac{8\Delta}{\eta h_- T} + \frac{24\kappa_{\boldsymbol{H}}^2 \mathbb{E}\|\boldsymbol{e}_1\|^2}{(1-\beta)T} \\ + 24\kappa_{\boldsymbol{H}}^2 \left[\left(1 + \frac{1}{1-\beta}\right)\bar{\sigma}_T^2 + \frac{D_T}{T} \right]. \end{aligned} \tag{27}$$

To verify the constants, replace $\boldsymbol{w}_t$ by $\boldsymbol{w}_t + \boldsymbol{H}_t^{-1}\boldsymbol{\delta}_t$ and let W_T denote its squared-norm sum. Then $W_T \leq 2E_T + 2V_T + 2D_T$, while the tracking bound uses the actual parameter displacement. With the same C_0, C, a as in the proof above,

$$(1-2a)W_T \leq 2C_0 + 2aS_T + 2\left(1 + \frac{1}{1-\beta}\right)V_T + 2D_T.$$

Now $a \leq 1/(24\kappa_{\boldsymbol{H}}^2)$, so using $(1-2a)^{-1} \leq 2$ and absorbing $12\kappa_{\boldsymbol{H}}^2 a S_T \leq S_T/2$ proves equation 27.

If $\|\boldsymbol{J}\boldsymbol{\theta}_t\| \leq R$ almost surely, then $D_T/T \leq \lambda^2 R^2/h_-^2$. This is a neighborhood bound for the unregularized f, with decay treated as a perturbation. A changing diagonal preconditioner generally prevents identifying decoupled decay with one fixed Euclidean L_2-regularized objective.

H FROZEN-GATE NOISE–LAG TRADE-OFF

To separate smoothing from lag, consider a local model with independent zero-mean noise of covariance $\boldsymbol{\Sigma}$, $\operatorname{tr}\boldsymbol{\Sigma} > 0$, and $0 < \beta < 1$. Freeze the gate at a constant $\gamma \in [0,1]$ and let the local mean drift linearly, $\boldsymbol{\mu}_{t-j} = \boldsymbol{\mu}_t - j\boldsymbol{a}$, with $\boldsymbol{u}_{\text{eff},t} = \boldsymbol{g}_t + \gamma\sum_{j\geq 1}\beta^j \boldsymbol{g}_{t-j}$ and $k(\gamma) = 1 + \beta\gamma/(1-\beta)$. Writing $S = \operatorname{tr}\boldsymbol{\Sigma}$, the scale-calibrated direction $\boldsymbol{u}_{\text{eff},t}/k(\gamma)$ has bias $-\gamma\beta\boldsymbol{a}/\{k(\gamma)(1-\beta)^2\}$ and covariance $\{1 + \gamma^2\beta^2/(1-\beta^2)\}\boldsymbol{\Sigma}/k(\gamma)^2$, so

$$\mathbb{E}\left\| \frac{\boldsymbol{u}_{\text{eff},t}}{k(\gamma)} - \boldsymbol{\mu}_t \right\|^2 = \frac{S + \gamma^2\left\{ \dfrac{\beta^2 S}{1-\beta^2} + \dfrac{\beta^2\|\boldsymbol{a}\|^2}{(1-\beta)^4} \right\}}{\{1 + \beta\gamma/(1-\beta)\}^2}. \tag{28}$$

Differentiating with $b_0 = \beta/(1-\beta)$ and $C_1 = \beta^2 S/(1-\beta^2) + \beta^2\|\boldsymbol{a}\|^2/(1-\beta)^4$ gives derivative $2(C_1\gamma - b_0 S)/(1 + b_0\gamma)^3$, so the MSE-optimal constant gate is

$$\gamma^\star = \min\left\{1,\ \frac{(1+\beta)/\beta}{1 + (1+\beta)\|\boldsymbol{a}\|^2/\{(1-\beta)^3 S\}}\right\}.$$

This calculation has two limiting cases. With a *stationary* mean ($\boldsymbol{a} = \boldsymbol{0}$) the unconstrained optimum is $(1+\beta)/\beta > 1$, so $\gamma^\star = 1$: ordinary momentum is best within this family, the MSE decreases throughout $\gamma \in [0,1]$, and the squared SNR $k(\gamma)^2\|\boldsymbol{\mu}\|^2/[\{1+\gamma^2\beta^2/(1-\beta^2)\}S]$ is nondecreasing on $[0,1]$, and strictly increasing when $\|\boldsymbol{\mu}\| > 0$. *Attenuation therefore has no universal stationary-noise SNR advantage.* With sufficient drift, an interior gate minimizes MSE by trading smoothing for reduced lag. This motivates adaptive mixing under non-stationarity. It does not assert that the cosine rule equals the MSE-optimal gate, and it requires a frozen gate, so it does not factor expectations through the implemented random gate.

I GATE-INDUCED BIAS UNDER BOUNDED NOISE

The following example shows that an unbiased gradient oracle does not make the AGAM update unbiased, even at a stationary mean. The proof of Theorem D.6 retains the gate–noise dependence explicitly.

Consider a frozen one-dimensional oracle at a population stationary point, $g_t = \xi_t$ with $\xi_t = 1$ with probability 0.9 and $\xi_t = -9$ with probability 0.1, which is bounded and has mean zero. For $\beta = 0.9$ the stationary heavy-ball state $m_{t-1} = \sum_{j\geq 0} \beta^j \xi_{t-1-j}$ has mean zero, lies in $[-90, 10]$, and is independent of g_t. In one dimension, away from a zero probe, the ideal gate is 1 when g_t and $g_t + \beta m_{t-1}$ share a sign and 0 otherwise. For $g_t = -9$ it is 1 almost surely, and for $g_t = 1$ it is 1 when $m_{t-1} > -1/\beta$ and 0 below. Hence

$$\mathbb{E}[u_{\text{eff},t}] = 0.9\beta\,\mathbb{E}\big[m_{t-1}\mathbf{1}\{m_{t-1} > -1/\beta\}\big] + 0.45\beta\,\mathbb{E}\big[m_{t-1}\mathbf{1}\{m_{t-1} = -1/\beta\}\big] > 0,$$

strictly, because two consecutive negative samples place the state below $-1/\beta$ with positive probability. This is a frozen-oracle bias example, not a nonconvergence counterexample on a changing objective: it does not prove that every noisy AGAM trajectory has a nonzero limiting error.

J ALIGNMENT GEOMETRY

Let $\boldsymbol{P} = \mathbf{D}_t^{-1} \succ 0$ and $\boldsymbol{m} = \hat{\boldsymbol{m}}_t$. The default update cosine and the metric cosine have numerators $\boldsymbol{g}_t{}^\top \boldsymbol{P}\boldsymbol{m}$ and $(\boldsymbol{P}^{1/2}\boldsymbol{g}_t)^\top(\boldsymbol{P}^{1/2}\boldsymbol{m})$, respectively, which are identical. Their magnitudes can differ, so they can still produce different gates.

The other two geometries need not detect the sign of the applied base direction. For $\boldsymbol{g}_t = (1, 1)$, $\boldsymbol{m} = (2, -1)$, and $\boldsymbol{P} = \operatorname{diag}(0.1, 1)$, $\boldsymbol{g}_t{}^\top \boldsymbol{m} = 1$ but $\boldsymbol{g}_t{}^\top \boldsymbol{P}\boldsymbol{m} = -0.8$. For $\boldsymbol{g}_t = (1, 1)$, $\boldsymbol{m} = (-3, 1)$, and $\boldsymbol{P} = \operatorname{diag}(1, 2)$, the actual score is $\boldsymbol{g}_t{}^\top \boldsymbol{P}\boldsymbol{m} = -1$, whereas the numerator after preconditioning both vectors is $\boldsymbol{g}_t{}^\top \boldsymbol{P}^2\boldsymbol{m} = 1$. These examples concern gradient-driven directions. Adding weight decay introduces a separate directional-derivative term.

K VERIFICATION OF THE DERIVATIONS

The accompanying numerical checks supplement the written proofs. Using the optimizer directly with explicit paper configurations, float64 arithmetic, and a fixed seed, we checked three momentum values, three parameter tensors, and 32 steps. The largest discrepancy between an actual parameter update and the derived mixture of Proposition D.1 was below 1.8×10^{-14}. These cases include startup, zero gradients, and gradients below the alignment norm floor. Symbolic checks confirm the sharp threshold of Lemma D.3, the MSE derivative of equation 28, and the delta-method variance of Appendix F.4. Additional numerical checks cover $100{,}000$ random geometric configurations and the descent, tracking, and PL inequalities along two 400-step quadratic trajectories. For Lion, 288 tensor-steps check the state recurrence and the pre-sign QHM identity, whose maximum discrepancy is below 3.4×10^{-16}, and $100{,}000$ random mixtures also check the signed-agreement inequality. The symbolic checks include the derivative of $s_\infty(\kappa)$ and Adam's exact normalized-mixture coefficient. These checks expose algebra and implementation mismatches, but they do not replace the written proofs, establish training performance, or validate the Gaussian assumptions for the recorded telemetry.

L IMPLEMENTATION

All three variants compute one inner product and two norms per tensor per step and keep no state beyond the base optimizer's. In every variant, the norms in the cosine are clamped at 10^{-8}, and a zero gradient falls back to the base coefficient. With a nonzero gradient and a zero proposed update, SGD and Adam use gate $1/2$, while Lion uses the base coefficient (Appendix F.1). All experiments use the gate of equation 1 without any exponent, the transient (non-persistent) update, and alignment with the proposed update. For AGAM-AdamW, the first moment is normalized by the gated mass of equation 5. TAM and AM use the published defaults: a smoothing rate of 0.9 and a gradient floor of 10^{-8} for TAM and AdaTAMW, and a proximal weight $\lambda = 0.1$ and a momentum cap $\beta_{\max} = 0.9$ for AM-MSGD and $\beta_{1,\max} = 0.89$ for AM-AdamW.

Algorithm 2 AGAM in PyTorch. Differences are in **blue**. The gate never writes to the momentum buffer.

```
# AGAM-SGD  (base: heavy-ball momentum SGD)
# g: grad; m: momentum; cos(.,.): cosine similarity
m.mul_(beta).add_(g)                          # SGDM buffer, unchanged
gate = (1 + cos(g, m)) / 2                    # alignment gate
p.add_(m.lerp(g, 1 - gate), alpha=-lr)

# AGAM-Adam  (base: Adam)
# m, v: 1st and 2nd moments; bc_i = (1 - beta_i**t)
# denom = sqrt(v / bc_2) + eps
m.lerp_(g, 1 - beta_1)                          # Adam buffer, unchanged
u = m / (bc_1 * denom)                          # Adam's direction
gate = (1 + cos(g, u)) / 2                      # alignment gate
nu   = gate * bc_1 / (1 - gate * beta_1**t)     # normalized gate
p.add_(u * nu + (1 - nu) * g / denom, alpha=-lr)

# AGAM-Lion  (base: Lion)
u = m.lerp(g, 1 - beta_1).sign()                # Lion's direction
gate = (1 + cos(g, u)) / 2                      # alignment gate
p.add_(m.lerp(g, 1 - gate * beta_1).sign(), alpha=-lr)
m.lerp_(g, 1 - beta_2)                          # Lion buffer, unchanged
```

M Experimental Details

Every configuration of every method is trained with seeds 42, 1337, and 2026. AGAM variants inherit their base optimizer's momentum coefficients and are swept over exactly the base optimizer's grid. Table 4 summarizes the grids, and the paragraphs below give the full recipes.

Image classification. Both networks are torchvision ResNets (He et al., 2016) trained from scratch and adapted to 32×32 inputs: the 7×7, stride-2 stem is replaced by a 3×3, stride-1 convolution with padding 1 and Kaiming-normal initialization, the initial max-pooling layer is removed, every BatchNorm layer is replaced by GroupNorm (Wu & He, 2018) with $\min(32, C/4)$ groups for C channels, and the classifier outputs 10 or 100 classes. Training images are randomly cropped to 32×32 after four-pixel reflective padding, flipped horizontally with probability 0.5, augmented with RandAugment (Cubuk et al., 2020) (two operations, magnitude 9), normalized per dataset, and randomly erased (Zhong et al., 2020) with probability 0.1, area scale $[0.02, 0.33]$, and aspect ratio $[0.3, 3.3]$. We train for 200 epochs with label smoothing 0 on CIFAR-10 and 0.1 on CIFAR-100. The learning rate warms up linearly over five epochs from 1% of its nominal value and then follows a step-wise cosine decay to 10^{-6}. Weight decay 5×10^{-4} is applied as coupled L_2 regularization and excludes biases and one-dimensional normalization parameters. Batches larger than 512 are formed by gradient accumulation over micro-batches of at most 512. We use two protocols. *Final:* train on all 50,000 training images and report the test accuracy after the last epoch on the official 10,000-image test set. *Validation-selected:* split the training set once into a stratified $85\%/15\%$ partition (42,500 training and 7,500 validation images, i.e., 750 per class on CIFAR-10 and 75 on CIFAR-100), evaluate validation accuracy after every epoch, and report the test accuracy of the checkpoint with the highest validation accuracy. The epochs-to-target diagnostic (Figure 3) records the first epoch at which validation accuracy reaches 90% on CIFAR-10 or 70% on CIFAR-100.

MAE pre-training. We pre-train an MAE (He et al., 2022) with a ViT-Tiny encoder (Dosovitskiy et al., 2021) and a 4-block decoder on Tiny-ImageNet (Le & Yang, 2015) at 64×64 resolution, with patch size 8, 75% masking, random resized crops, and horizontal flips. Training lasts 300 epochs at batch size 1024 (micro-batches of 256 with gradient accumulation), with a 15-epoch linear warmup followed by cosine decay, BF16 autocast, and TF32. The peak learning rate is the base learning rate times $B/256$. The AdamW family uses $(\beta_1, \beta_2) = (0.9, 0.95)$ and sweeps base learning rates $\{1.5 \times 10^{-4}, 10^{-3}, 1.5 \times 10^{-3}, 2 \times 10^{-3}\}$ and weight decays $\{10^{-3}, 0.05, 0.15\}$. The Lion family uses $(\beta_1, \beta_2) = (0.9, 0.99)$ and sweeps base learning rates $\{5 \times 10^{-5}, 10^{-4}, 1.5 \times 10^{-4}, 2 \times 10^{-4}\}$ and weight decays $\{0.15, 0.3, 0.5\}$. We split the 100,000 training images once into a stratified development-training set of 90,000 images and a development-validation set of 10,000 images (50

Table 4: Training recipes and hyperparameter grids. Every configuration is run with seeds 42, 1337, and 2026. AGAM variants use exactly their base optimizer's grid and coefficients, and TAM and AM use the same grid as their base optimizer except where noted, with their method-specific hyperparameters at the defaults of Appendix L.

	CIFAR-10	**CIFAR-100**	**MAE (Tiny-ImageNet)**	**SmolLM2 (WikiText-2)**	**Qwen2.5 (GSM8K)**
Model	ResNet-18 (GN)	ResNet-50 (GN)	ViT-Tiny MAE, patch 8	SmolLM2-135M, full	Qwen2.5-3B, LoRA r=32
Base optimizer	SGDM	SGDM	AdamW; Lion	AdamW	AdamW; Lion
Momentum	$\beta = 0.9$	$\beta = 0.9$	AdamW $(0.9, 0.95)$; Lion $(0.9, 0.99)$	$(0.9, 0.999)$	AdamW $(0.9, 0.999)$; Lion $(0.9, 0.99)$
Learning rate	$0.025 \cdot 2^i$, $i = 0, \ldots, 5$	$0.025 \cdot 2^i$, $i = 0, \ldots, 5$	base LR $\times B/256$; AdamW {1.5e-4, 1e-3, 1.5e-3, 2e-3}; Lion {5e-5, 1e-4, 1.5e-4, 2e-4}	{0.3, 1, 3} $\times$ 5e-5	AdamW family {5e-5, 1e-4, 2e-4, 8e-4}; AdaTAMW {1e-5, 4e-5, 5e-5, 1e-4}; Lion {1e-5, 1e-4, 1.5e-4}
Batch size	$64 \cdot 2^j$, $j = 0, \ldots, 6$	$128 \cdot 2^j$, $j = 0, \ldots, 5$	1024	32 blocks	32 examples
Weight decay	5e-4 (coupled)	5e-4 (coupled)	AdamW {1e-3, 0.05, 0.15}; Lion {0.15, 0.3, 0.5}	0	AdamW family {0, 0.1}; Lion {0, 0.1, 0.5}
Schedule	5-epoch warmup, cosine to 10^{-6}	5-epoch warmup, cosine to 10^{-6}	15-epoch warmup, cosine	constant; 10% warmup, cosine to 0	5% warmup, cosine to 10% of peak
Budget	200 epochs	200 epochs	300 epochs	5 epochs (780 steps)	3 epochs (633 steps)
Selection	final; best validation	final; best validation	probe accuracy (LR, WD)	validation loss	development loss

per class). Every configuration is pre-trained on the development-training set for the full 300 epochs without early stopping, and each optimizer's learning rate and weight decay are selected by the linear-probe accuracy of the epoch-300 encoder on the development-validation set.

Linear probe. The encoder is frozen and the unmasked class-token features are extracted once, from images resized deterministically to 64×64 with bicubic interpolation and ImageNet normalization, without augmentation. The head is a non-affine BatchNorm ($\epsilon = 10^{-6}$) followed by a linear layer initialized from a truncated normal with standard deviation 0.01 and zero bias. It is trained on the development-training set with LARS (You et al., 2017) (momentum 0.9, no weight decay, trust coefficient 0.001) at batch size 4096 and base learning rate 0.1 (peak $0.1 \times 4096/256 = 1.6$) for 90 epochs, with a 10-epoch linear warmup and cosine decay to zero, and evaluated on the development-validation set. The same probe also monitors every 50th pre-training epoch.

Fine-tuning. The selected epoch-300 encoder of each optimizer is fine-tuned end to end with one fixed recipe on all 100,000 training images, and evaluated on the official 10,000-image validation set after epoch 100 (top-1 and top-5). The recipe uses AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, base learning rate 5×10^{-4} (peak 2×10^{-3} at effective batch size 1024, micro-batches of 256 with 4 accumulation steps), a 5-epoch linear warmup followed by cosine decay to 10^{-6}, weight decay 0.05, layer-wise learning-rate decay 0.65, drop path 0.1, Mixup 0.8, CutMix 1.0, label smoothing 0.1, global average pooling over the patch tokens, BF16 autocast, and TF32.

SmolLM2 fine-tuning. We fine-tune all 134,515,008 parameters of the pre-trained `HuggingFaceTB/SmolLM2-135M` (Ben Allal et al., 2025) on the official splits of WikiText-2 (`wikitext-2-raw-v1`) (Merity et al., 2017). Each non-empty document is tokenized without

added special tokens and followed by the EOS token, and the resulting stream is cut into non-overlapping 512-token blocks, giving 5,013 training blocks (4,992 used per epoch, since the incomplete effective batch is dropped), 523 validation blocks, and 595 test blocks. Each block contributes 511 next-token targets, about 2.55M per epoch. Training lasts 5 epochs at an effective batch size of 32 blocks (micro-batch 8, 4 accumulation steps), i.e., 156 updates per epoch and 780 in total, with FP32 parameters and optimizer states, BF16 autocast, TF32, PyTorch SDPA attention, global gradient-norm clipping at 1.0, no weight decay, and $(\beta_1, \beta_2) = (0.9, 0.999)$. The learning rate is swept over multipliers $\{0.3, 1, 3\}$ of the base learning rate 5×10^{-5}, under both a constant learning rate and a 10% linear warmup (78 updates) followed by cosine decay to zero. Validation loss is measured before fine-tuning and after every epoch, and the checkpoint with the lowest validation loss is restored and evaluated once on the test split. Since perplexity is the exponential of the loss, this is equivalent to selecting by validation perplexity.

Qwen2.5-3B on GSM8K. We fine-tune the pre-trained base (not instruction-tuned) `Qwen/Qwen2.5-3B` (Qwen Team, 2024) on GSM8K (Cobbe et al., 2021) with LoRA (Hu et al., 2022) of rank 32, scaling $\alpha = 64$, and no dropout, applied to the attention projections (`q_proj`, `k_proj`, `v_proj`, `o_proj`) and the MLP projections (`gate_proj`, `up_proj`, `down_proj`), for 59,867,136 trainable adapter parameters. The base weights, embeddings, and biases stay frozen. The official 7,473 training examples are split once, by a question hash with seed 20260918 shared across optimizers and seeds, into 6,726 training and 747 development examples, and all 1,319 official test examples are held out. Training lasts 3 epochs (633 updates) at an effective batch size of 32 examples (micro-batch 2, normally 16 accumulation steps, with final partial batches kept), on variable-length sequences of at most 1,024 tokens with dynamic padding and neither packing nor truncation. The objective is teacher-forced, token-level cross-entropy on the response tokens including EOS, normalized over response tokens across accumulated micro-batches, and prompt and padding tokens are excluded. Each run processes 3,762,045 non-padding input tokens over the three epochs, 2,481,381 of them supervised. The schedule is a 5% linear warmup (32 updates) followed by cosine decay to 10% of the peak learning rate, with global gradient-norm clipping at 1.0 after accumulation. Frozen base weights and the forward pass use BF16, while adapter parameters and optimizer states use FP32, without quantization. Each run uses one NVIDIA A10G GPU. The checkpoint with the lowest development response cross-entropy is evaluated on the test set with one greedy completion per question, without in-context examples, sampling, voting, or a verifier, allowing 512 new tokens and stopping at EOS or at a subsequent `Question:` delimiter. Exact match compares the normalized number after `####` with the reference answer, and the reasoning itself need not match.

N Ablations

Each variant changes one design choice relative to default AGAM and is otherwise trained with the same recipe and three seeds. *Persistent* stores the gated momentum in the buffer, as TAM and AM do. *Stale reference* measures alignment against the stored buffer $\boldsymbol{m}_{t-1}$ instead of the proposed update $\boldsymbol{u}_t$. *Hard reset* keeps the soft gate for $s_t \geq 0$ but sets $\gamma_t = 0$ whenever $s_t < 0$. Like the default gate, it acts only on the current step and never clears the buffer. *Model-wide gate* computes a single gate from the concatenated gradients and updates of all tensors. Table 5 reports AGAM-SGD on ResNet-50/CIFAR-100 at $\eta = 0.1$ and $B = 256$, and Table 6 reports AGAM-AdamW during MAE pre-training with the configuration of Table 1. In both tables, AUC is the area under the accuracy curve over training, normalized by the training length.

Every variant is worse than default AGAM with both base optimizers, in final accuracy and in AUC. The two properties that separate AGAM from TAM and AM matter most. Storing the gated momentum costs 1.18 points on CIFAR-100 and 1.79 on the MAE probe, the largest drop in both settings, and measuring alignment against the stale buffer costs 1.08 and 1.22 points. On CIFAR-100 these two variants also have the lowest AUC, so their deficit builds up during training and is not confined to the final epochs. The hard reset costs only 0.46 points on CIFAR-100 but 1.67 on MAE, where it also has the largest seed variance and the highest validation loss. This favors rescaling the memory continuously over switching it off. The model-wide gate costs the least on MAE and little on CIFAR-100, even though it misses most per-tensor conflicts (Figures 17 and 22).

Table 5: Ablations of AGAM-SGD on ResNet-50/CIFAR-100 at $\eta = 0.1$, $B = 256$. Final test accuracy is reported as mean $\pm$ s.d. over three seeds, and Δ is its change relative to default AGAM. Higher accuracy and AUC are better.

Variant	**Final test acc. (%)** $\uparrow$	Δ **(pp)**	**Test AUC** $\uparrow$
Default AGAM	$\mathbf{79.02 \pm 0.12}$	–	$\mathbf{67.20}$
Hard reset	78.56 ± 0.30	-0.46	67.08
Model-wide gate	78.47 ± 0.39	-0.55	67.06
Stale reference	77.94 ± 0.24	-1.08	65.85
Persistent	77.84 ± 0.16	-1.18	64.55

Table 6: Ablations of AGAM-AdamW during MAE pre-training of ViT-Tiny on Tiny-ImageNet. Final probe accuracy is the linear-probe accuracy of the epoch-300 encoder, reported as mean $\pm$ s.e. over three seeds, and Δ is its change relative to default AGAM. Higher accuracy and AUC are better, and lower validation loss is better.

Variant	**Final probe acc. (%)** $\uparrow$	Δ **(pp)**	**Probe AUC** $\uparrow$	**Final val. loss** $\downarrow$
Default AGAM	$\mathbf{29.77 \pm 0.16}$	–	$\mathbf{26.80}$	$\mathbf{0.58362}$
Model-wide gate	29.53 ± 0.55	-0.24	26.15	0.58506
Stale reference	28.55 ± 0.79	-1.22	25.33	0.58714
Hard reset	28.10 ± 1.50	-1.67	24.19	0.59231
Persistent	27.98 ± 1.00	-1.79	24.56	0.58701

O Additional Results

Figures 23–25 complete Figure 2, and Figures 3 and 26 give the epochs-to-target counts discussed in Section 4.1.

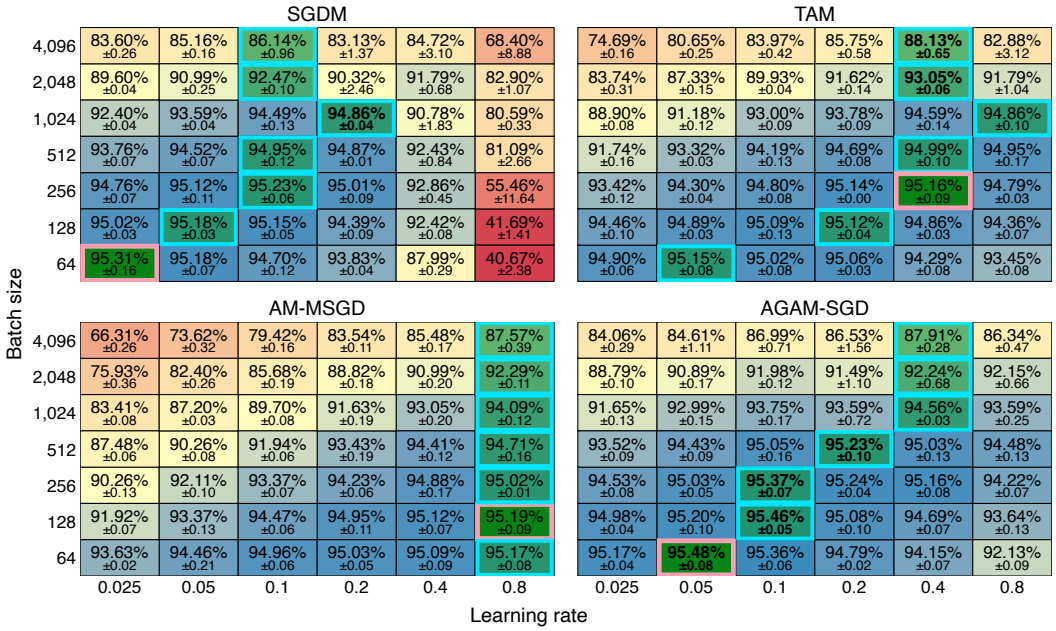


Figure 23: Validation-selected test accuracy (%) on ResNet-18/CIFAR-10 over the full grid (mean $\pm$ s.e. over three seeds). AGAM attains the best cell overall, 95.48% at $\eta = 0.05$, $B = 64$. The clearest structure is the $\eta = 0.8$ column: SGDM degrades sharply there at every batch size below 1024 (to 40.67% at $B = 64$ and 41.69% at $B = 128$), while AGAM stays between 92.13% and 94.48%. TAM and AM are stable in that column but each pays for it elsewhere, TAM at small η and AM across the whole low-η region.

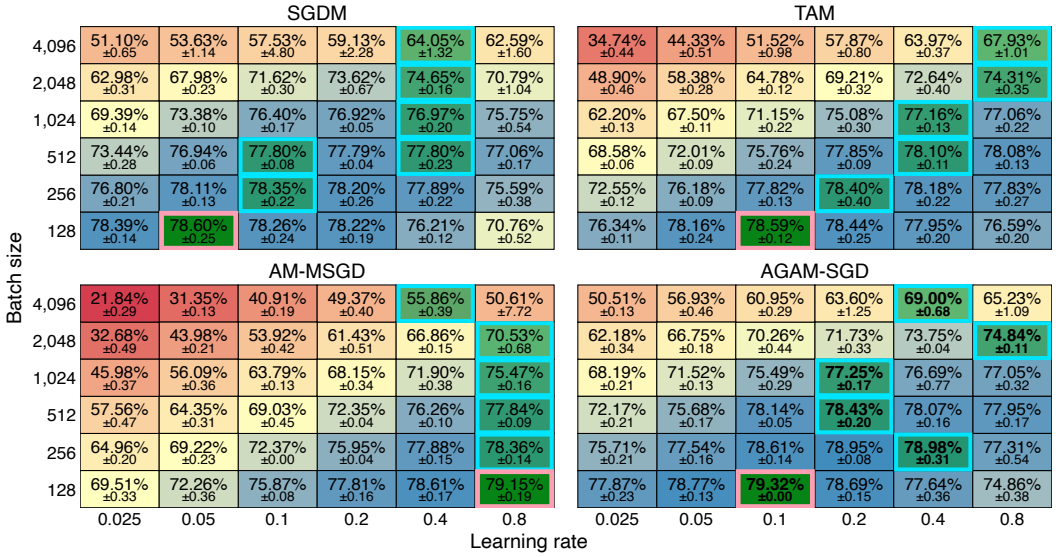


Figure 24: Validation-selected test accuracy (%) on ResNet-50/CIFAR-100 over the full grid (mean $\pm$ s.e. over three seeds). AGAM attains the best cell overall (79.32% at $\eta = 0.1$, $B = 128$) and the best cell in every batch-size row. At $B = 4096$ its best reaches 69.00%, against 64.05% for SGDM, 67.93% for TAM, and 55.86% for AM.

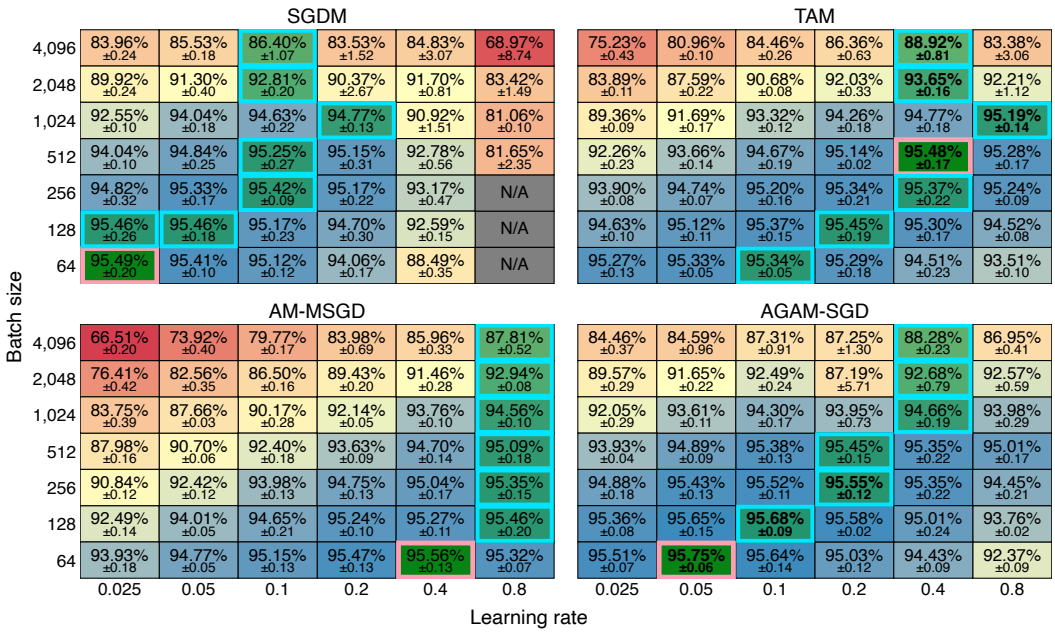
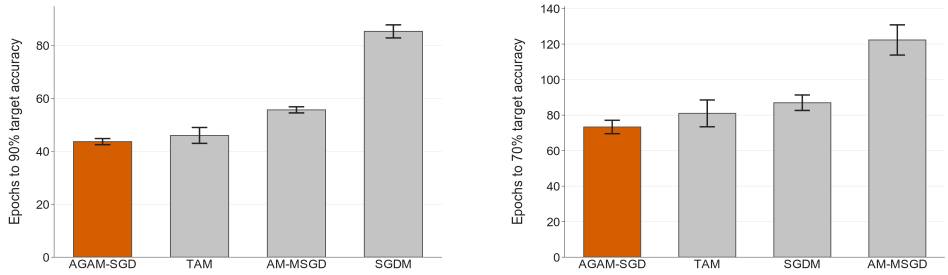


Figure 25: Final-epoch test accuracy (%) on ResNet-18/CIFAR-10 over the full grid (mean ± s.e. over three seeds). The CIFAR-100 counterpart is Figure 2. Cells marked N/A indicate optimizer divergence. The conclusions of Section 4.1 do not depend on the selection rule. The best final-epoch cells are 95.75% (CIFAR-10) and 79.41% (CIFAR-100) for AGAM, against 95.49% and 78.82% for the base optimizer SGDM.



(a) ResNet-18/CIFAR-10, target 90%.

(b) ResNet-50/CIFAR-100, target 70%.

Figure 26: Epochs until the validation accuracy first reaches a fixed target at $\eta = 0.05$, $B = 128$ (mean ± s.d. over three seeds). AGAM-SGD needs 43.5 epochs on CIFAR-10, against 46.0 for TAM, 56.7 for AM, and 83.2 for SGDM. On CIFAR-100, AGAM-SGD needs 73.3 epochs, against 81.0, 122.3, and 87.0, respectively.