# HYBRID MOMENTUM METHODS FOR OPTIMIZING DEEP LEARNING MODELS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Momentum is central to stochastic optimization in deep learning, including the training of large language models. Classical momentum methods typically rely on a fast exponential moving average (EMA) scheme, which discounts past gradients on a fixed timescale and may underweight older gradients that remain informative. This creates a trade-off between retaining long-term history and adapting to recent changes. We introduce Hybrid Momentum (`HyM`), which combines a linearly weighted average of past gradients with a geometrically discounted fast momentum. The former assigns weights proportional to the gradients' iteration indices and retains a growing span of history, while the latter emphasizes recent gradients. This hybrid construction is modular and can be combined with adaptive preconditioning and other update transformations. In particular, we propose `HyMuon`, which applies Newton–Schulz orthogonalization to the hybrid direction. Illustrative examples and controlled experiments demonstrate the benefits of combining gradient information across these two timescales. Across image classification, machine translation, and language modeling, `HyM` and `HyMuon` match or improve upon strong baselines. `HyMuon` improves over Muon on IWSLT14, and in both the 100k- and 600k-step GPT-2 124M OpenWebText regimes. For LLaMA-1.3B pretraining on C4, `HyMuon` achieves a final validation loss of 2.777, compared with 2.903 for Muon.

## 1 INTRODUCTION

Momentum methods improve stochastic optimization by combining the current gradient with information from earlier updates. In deep learning, this history is usually maintained through geometric discounting, as in Heavy-Ball momentum and commonly used Nesterov-style updates (Polyak, 1964; Sutskever et al., 2013). Geometric discounting emphasizes recent gradients, but it also limits how long older gradients can influence training. When useful information arrives intermittently across mini-batches, retaining more of this history may help, provided the update remains responsive to changes in the loss landscape.

By combining momentum with iterate averaging, Defazio et al. (2024) proposed the Schedule-Free method, which achieves strong empirical performance without a prescribed learning-rate decay schedule. The empirical success of Schedule-Free optimization motivates revisiting how this history is accumulated. To see how its averaging mechanism retains past gradients, consider the constant-step-size Schedule-Free SGD recurrence

$$\begin{aligned} \mathbf{y}_t &= (1-\beta)\mathbf{z}_t + \beta\mathbf{x}_t, \\ \mathbf{z}_{t+1} &= \mathbf{z}_t - \eta\mathbf{g}_t, \\ \mathbf{x}_{t+1} &= (1-c_{t+1})\mathbf{x}_t + c_{t+1}\mathbf{z}_{t+1}, \end{aligned} \tag{1}$$

where $\mathbf{z}_t$ is the base iterate, $\mathbf{x}_t$ is the averaged iterate, and $\mathbf{y}_t$ is the training iterate. Here $c_{t+1} = 1/(t+1)$, $\mathbf{z}_1 = \mathbf{x}_1$, and $\mathbf{g}_t$ is a stochastic gradient evaluated at $\mathbf{y}_t$. Eliminating $\mathbf{x}_t$ and $\mathbf{z}_t$ gives a single-sequence form that exposes its gradient memory (Morwani et al., 2025):

$$\begin{aligned} \mathbf{n}_t &= (1-c_t)\mathbf{n}_{t-1} + \mathbf{g}_t, \\ \mathbf{y}_{t+1} &= \mathbf{y}_t - \eta[\beta c_{t+1}\mathbf{n}_t + (1-\beta)\mathbf{g}_t], \end{aligned} \tag{2}$$

with $c_t = 1/t$ and $\mathbf{n}_0 = \mathbf{0}$; Appendix A gives the derivation for completeness. In this form, the update combines a running weighted sum of past gradients with the current gradient. As training



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.

proceeds, this weighted sum spans a longer history, but each new gradient contributes a smaller fraction of its total weight. The historical component therefore responds more slowly to new gradients later in training, suggesting that a separate momentum state emphasizing recent gradients can complement this long-term accumulation.

Building on this perspective, we propose Hybrid Momentum (`HyM`), which combines a linearly weighted gradient average with a geometrically discounted fast momentum. The average assigns weights proportional to the gradients' iteration indices and retains a growing span of history, while the fast momentum emphasizes recent gradients. `HyM` also maintains an averaged iterate formed from its training iterates for evaluation and checkpointing. Our convergence analysis via cumulative gradient weights accommodates this growing gradient memory and establishes an $\mathcal{O}(T^{-1/2})$ bound on the expected suboptimality of the averaged iterate for smooth convex objectives. Moreover, this hybrid momentum scheme is modular and can be combined with adaptive preconditioning and other update transformations. In particular, applying Muon's Newton–Schulz (NS) orthogonalization to the combined momentum (Jordan et al., 2024), we develop Hybrid Momentum Orthogonalized by NS (`HyMuon`), which applies to matrix optimization.

We characterize the weights assigned to past gradients and use controlled experiments to examine why this history can remain useful. These experiments show that stored gradients can still lower a target loss thousands of steps later, and that replacing them with current gradients can lower it further. We relate these observations to uneven learning signals across batches and to batch size. Across the evaluated benchmarks, `HyM` matches or improves upon Schedule-Free on image classification, while `HyMuon` improves upon Muon on IWSLT14 and across the evaluated language-modeling settings.

**Notation.** The training and averaged iterates are $\mathbf{y}_t$ and $\mathbf{x}_t$; uppercase letters such as $\mathbf{Y}_t$ and $\mathbf{X}_t$ denote the matrix variables. Vector norms are Euclidean, while $\|\cdot\|_{\mathrm{op}}$ and $\|\cdot\|_*$ denote the matrix operator and nuclear norms. Probabilistic notation is introduced where it is used.

## 2 Related work

Momentum methods use past information in two standard ways: by accumulating gradients or by introducing inertial terms based on differences between successive iterates (Polyak, 1964; Nesterov, 1983). For smooth convex stochastic optimization, Lan (2012) established an optimal convergence rate for accelerated stochastic approximation, while Yang et al. (2016) and Ghadimi & Lan (2016) analyzed stochastic momentum and accelerated-gradient methods for smooth nonconvex objectives. More recently, Wang & Yurtsever (2026) unified Polyak and Nesterov variants within a generalized SGDM framework and analyzed both convex and nonconvex smooth objectives. In deep learning, EMA-Nesterov replaces the standard lookahead direction with an exponential moving average of parameter updates, stabilizing the extrapolation performed by a base optimizer (Yau et al., 2026).

Some methods combine several momentum components to capture different timescales. Aggregated Heavy-Ball studies the convergence of such multi-component schemes (Danilova, 2022), while AdEMAMix pairs fast and slow EMAs to retain gradients over short and long horizons (Pagliardini et al., 2025). `HyM` adopts the same fast–slow structure but replaces the slow EMA with a linearly weighted gradient average. Its slow momentum therefore retains an increasingly long history throughout training, whereas AdEMAMix increases its slow decay coefficient during warmup toward a prescribed value.

Morwani et al. (2025) show that Schedule-Free optimization (Defazio et al., 2024) can be expressed as accelerated SGD followed by weight averaging, clarifying the distinct roles of gradient memory and the averaged iterate. ScheduleFree+ restores Adam's inner first-moment state and extends this construction with Polyak-style learning-rate adaptation, delayed averaging, and an annealed interpolation between its raw and averaged iterates (Defazio, 2026). Although ScheduleFree+ and `HyM` both use momentum and return an averaged iterate, they retain long-term information differently. ScheduleFree+ uses exponentially weighted Adam moments and feeds its averaged iterate into the next gradient query. By contrast, `HyM` mixes a linearly weighted gradient average with a geometrically discounted fast momentum at each step, while maintaining a separate averaged iterate that does not affect its gradients or momentum. Schedule-Free convergence analysis relies on online-to-batch conversion for convex optimization (Defazio et al., 2024). Our analysis instead studies how gradient

memory that grows with training interacts with iterate averaging to provide a convergence guarantee for the averaged iterate.

Preconditioning is also important for training large language models. Two common approaches are diagonal preconditioning, implemented through a vector of coordinatewise scaling factors as in Adam and AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019), and structured matrix preconditioning, as in Shampoo, K-FAC, and SOAP (Gupta et al., 2018; Martens & Grosse, 2015; Vyas et al., 2025). More recently, Muon has attracted considerable interest in training large models by applying NS orthogonalization to matrix momentum (Jordan et al., 2024), motivating subsequent work on efficiency, stability, and distributed training (Li et al., 2025; Kimi Team, 2025; Ahn et al., 2025).

## 3 The Hybrid Momentum Methods

### 3.1 Hybrid Momentum (HyM)

We consider stochastic optimization problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^d} f(\mathbf{x}), \qquad f(\mathbf{x}) := \mathbb{E}_\xi[f(\mathbf{x};\xi)], \tag{3}$$

where $\xi$ is a random sample. Algorithm 1 maintains two momentum states. The slow momentum $\mathbf{m}_t$ is a linearly weighted gradient average that retains an increasingly long history, while the fast momentum $\mathbf{b}_t$ is a geometrically discounted gradient sum that emphasizes recent gradients. Their weighted combination updates the training iterate from $\mathbf{y}_t$ to $\mathbf{y}_{t+1}$, and the next gradient is evaluated at $\mathbf{y}_{t+1}$. A separate averaged iterate $\mathbf{x}_t$ is formed from the training iterates, and $\mathbf{x}_T$ is returned after training. Keeping the sequences separate allows us to distinguish the effect of the two momentum states on the training trajectory from the effect of iterate averaging on the final averaged iterate.

**Algorithm 1** Hybrid Momentum (HyM)

**Input:** $\mathbf{y}_1 \in \mathbb{R}^d$, $\eta > 0$, $\mu \in (0,1)$, $\beta \in [0,1)$, $B, T \in \mathbb{N}$; initialize $\mathbf{m}_0 = \mathbf{b}_0 = \mathbf{0}$, $\mathbf{x}_0 = \mathbf{y}_1$

1: **for** $t = 1, \dots, T$ **do**
2: Sample a mini-batch $\mathcal{B}_t = (\xi_{t,j})_{j=1}^B$
3: $\mathbf{g}_t = \frac{1}{B}\sum_{j=1}^B \nabla f(\mathbf{y}_t; \xi_{t,j})$
4: $\mathbf{m}_t = \frac{t-1}{t+1}\mathbf{m}_{t-1} + \frac{2}{t+1}\mathbf{g}_t$ ▷ Slow momentum
5: $\mathbf{b}_t = \mu\left(1 - \frac{1}{t}\right)\mathbf{b}_{t-1} + \mathbf{g}_t$ ▷ Fast momentum
6: $\boldsymbol{\Delta}_t = \beta\alpha_t\mathbf{m}_t + (1-\beta)\mathbf{b}_t$, where $\alpha_t = \frac{t+1}{2t}\frac{1-\mu^t}{1-\mu}$
7: $\mathbf{x}_t = (1-a_t)\mathbf{x}_{t-1} + a_t\mathbf{y}_t$, where $a_t = \frac{1}{1+(1-\beta)(t-1)}$
8: $\mathbf{y}_{t+1} = \mathbf{y}_t - \eta\boldsymbol{\Delta}_t$
9: **end for**

**Output:** $\mathbf{x}_T$

**Two momentum states.** Unrolling the two recurrences makes their weighting rules explicit:

$$\mathbf{m}_t = \frac{\sum_{i=1}^t i\,\mathbf{g}_i}{\sum_{i=1}^t i} = \sum_{i=1}^t \frac{2i}{t(t+1)}\mathbf{g}_i, \text{ and } \quad \mathbf{b}_t \quad = \sum_{i=1}^t \frac{i}{t}\mu^{t-i}\mathbf{g}_i. \tag{4}$$

At a fixed iteration $t$, the slow momentum assigns weight proportional to the gradient's iteration index $i$, giving more weight to newer gradients while retaining the entire history. This linearly weighted average is maintained recursively, without storing past gradients. Its newest-gradient weight is $2/(t+1)$. The fast momentum also contains a linear factor in $i$, together with a geometric discount $\mu^{t-i}$ that concentrates its weight on recent gradients. Its retention factor $\mu(1-1/t)$ approaches $\mu$, so it behaves like conventional EMA momentum at large $t$, while differing from a fixed-decay EMA at finite $t$. The rescaling factor $\alpha_t$ places the total gradient weights of the two momentum states on the same order, $\mathcal{O}((1-\mu)^{-1})$. Consequently, $\beta$ controls their relative contribution rather than compensating for a difference in scale. The step size $\eta$ then scales the combined momentum. Section 4.1 describes how weight is distributed over past gradients, and Appendix B.1 derives the coefficient sums and the weight assigned to each past gradient.

**Training and averaged iterates.** The training iterate $\mathbf{y}_t$ and the averaged iterate $\mathbf{x}_t$ satisfy the exact relation

$$\mathbf{y}_t = \mathbf{x}_t + (1-\beta)(t-1)(\mathbf{x}_t - \mathbf{x}_{t-1}). \tag{5}$$

The same parameter $\beta$ determines the averaging coefficient $a_t$: for fixed $t > 1$, a larger $\beta$ places more weight on the current training iterate $\mathbf{y}_t$. Equivalently, $\mathbf{x}_t = \sum_{i=1}^{t} \pi_{t,i}\mathbf{y}_i$, where $\pi_{t,i} = a_i \prod_{j=i+1}^{t}(1-a_j)$, $\pi_{t,i} \geq 0$, and $\sum_{i=1}^{t} \pi_{t,i} = 1$. Since $\mathbf{x}_t$ does not enter the gradient or momentum recursion, iterate averaging does not affect the training trajectory. Thus, $\{\mathbf{y}_t\}$ is the training trajectory, whereas $\{\mathbf{x}_t\}$ is the averaged sequence used for evaluation, and the algorithm returns $\mathbf{x}_T$.

### 3.2 Extension to Matrix Problems: HyMuon

The hybrid momentum scheme is highly modular and can be combined with other update transformations, such as the preconditioned gradient update. We illustrate this with the integration of Muon's update (Jordan et al., 2024). Specifically, we consider the following stochastic optimization problem with matrix-valued variables:

$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} f(\mathbf{X}), \qquad f(\mathbf{X}) := \mathbb{E}_{\xi}[f(\mathbf{X};\xi)]. \tag{6}$$

For hidden weight matrices, `HyMuon` applies the two momentum recurrences of `HyM` in Algorithm 1 to matrix gradients. It then applies Muon's Newton–Schulz (NS) orthogonalization to the combined momentum $\mathbf{D}_t$ (Jordan et al., 2024):

$$\mathbf{Y}_{t+1} = \mathbf{Y}_t - \eta\,\mathrm{NS}(\mathbf{D}_t), \tag{7}$$

where $\mathrm{NS}(\cdot)$ denotes the output of finitely many NS iterations approximating the polar factor. The averaged matrix iterate $\mathbf{X}_t$ follows the same averaging rule as in `HyM`. Appendix D gives the matrix-update pseudocode and the full-model implementation.

## 4 Why Past Gradients Can Remain Useful

`HyM` retains old gradients in its slow momentum and weights recent gradients through its fast momentum. We first characterize the weights assigned to past gradients, then test whether this history has a measurable effect on later training. Finally, we examine how useful contributions vary across batches.

### 4.1 Retaining history while responding to change

`HyM` retains the influence of past gradients without storing them or tracking their source batches. Unrolling its two momentum states gives the resulting weights:

$$\boldsymbol{\Delta}_t = \sum_{i=1}^{t} w_{t,i}\mathbf{g}_i, \qquad w_{t,i} = \frac{i}{t}\left(\frac{\beta(1-\mu^t)}{t(1-\mu)} + (1-\beta)\mu^{t-i}\right). \tag{8}$$

The slow-momentum term distributes weight linearly across the entire history, while the fast-momentum term applies an additional geometric discount to older gradients. These are gradient weights; the separate $\pi_{t,i}$ in Section 3.1 average parameters. For `HyMuon`, the same weights combine matrix gradients into the combined momentum $\mathbf{D}_t$ in Algorithm 2, before NS orthogonalization.

Average lag is the coefficient-weighted mean of $t-i$, the steps since gradient $i$ was computed. After normalizing each component's coefficients to sum to one, the gradients contributing to the slow momentum have an average lag of $(t-1)/3$ steps, whereas those contributing to the fast momentum have an average lag of $\mu/(1-\mu)$ steps as $t$ grows. As training proceeds, the two timescales separate: the slow momentum carries a growing span of history, and the fast momentum continues to respond to recent changes. In Figure 1, `HyM`'s weights increase approximately linearly across older gradients and form a peak on recent gradients. For fixed $\mu, \beta \in (0,1)$, any fixed early fraction of training retains a positive limiting share of the direction's total weight (Appendix B.1).

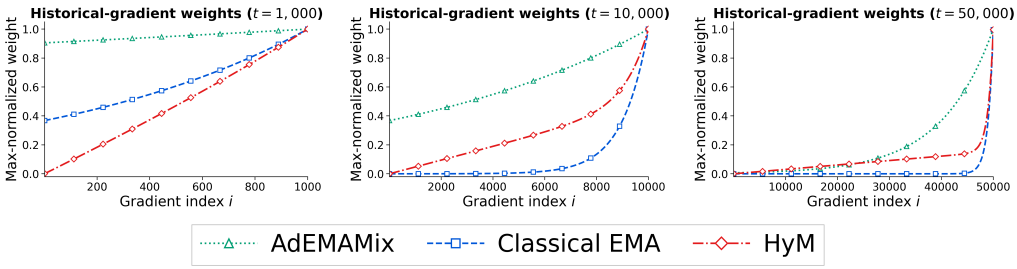


Figure 1: Max-normalized historical-gradient weighting profiles for AdEMAMix ($\beta_3 = 0.9999$), classical EMA ($\mu = 0.999$), and `HyM` ($\mu = 0.999$, $\beta = 0.9$) at $t \in \{1000, 10000, 50000\}$ (left to right). At each horizon, every method's raw coefficients are divided by that method's maximum coefficient, so the panels compare shape rather than overall coefficient scale.

Viewing $t$ as continuous, the slow-momentum recurrence is approximated by

$$\frac{\mathrm{d}\mathbf{m}}{\mathrm{d}\log t} = 2(\mathbf{g} - \mathbf{m}).$$

Once the gradient becomes constant at $\mathbf{g}$, the deviation $\mathbf{m}(t) - \mathbf{g}$ decays as $t^{-2}$. Reducing this deviation by a fixed factor therefore requires multiplying $t$ by a fixed factor, so the number of iterations needed to change the slow momentum grows during training. AdEMAMix also schedules an increasing slow memory (Pagliardini et al., 2025), whereas `HyM` uses the newest-gradient weight $2/(t+1)$, whose effective memory horizon continues to grow with $t$.

### 4.2 Old gradients can still affect later steps

A batch can affect later training through both its change to the parameters and its stored gradient's contribution to later update directions. To measure their combined lasting effect, we adapt AdEMAMix's held-out-batch forgetting experiment (Pagliardini et al., 2025) to GPT-2 124M on OpenWebText. A fixed batch $\mathcal{B}$ is excluded from ordinary sampling. The paired branches start from the same complete checkpoint immediately before step 5k. At that step, $\mathcal{B}$ replaces the ordinary batch in the inject branch; the control branch never trains on $\mathcal{B}$. Both branches then use the same ordinary-batch sequence. At step 10k, the control-minus-inject loss on $\mathcal{B}$ is about $0.030$ for Muon and $0.016$ for `HyMuon` (Figure 2, left). Thus, for both methods, training once on $\mathcal{B}$ leaves a measurable reduction in its loss 5,000 steps later.

To test whether stored history still contributes, we branch from the same complete checkpoint at step $t = 7500$. The stored contribution is the signed gradient difference $\Delta\mathbf{g}_s = \mathbf{g}_{\mathcal{B}}(s) - \mathbf{g}_{\text{ordinary}}(s)$ introduced at $s = 5000$. At the intervention, its coefficient in `HyMuon`'s slow momentum is $s/t = 2/3$. Keep retains this contribution, Delete removes it, and Refresh replaces it with a norm-matched current gradient on $\mathcal{B}$. Model parameters, the averaged iterate $\mathbf{x}_t$, and all other optimizer states are held fixed across the three branches. Appendix E.3.1 gives the complete construction.

At step 10k, `HyMuon` Delete, Keep, and Refresh give target losses of $3.048$, $3.038$, and $3.025$, respectively. Its stored contribution therefore still lowers the target loss, and refreshing it lowers the loss further. Muon's Keep, Delete, and Refresh values differ by less than $0.0024$ and overlap at the scale of the plot. Together, the two panels distinguish a lasting parameter change from continued use of optimizer memory (Appendix E.3.1).

### 4.3 Useful contributions are uneven across batches

For a particular prediction target, gradient steps based on different batches can yield different reductions in the target loss. A batch that provides a useful direction may be followed by one that contributes little to the same target. We examine this variation across candidate passages at fixed model parameters.

At each of three GPT-2 124M checkpoints, we apply one equal-length, normalized gradient step from each of 2,000 training fragments. Each candidate starts from the same parameters with no

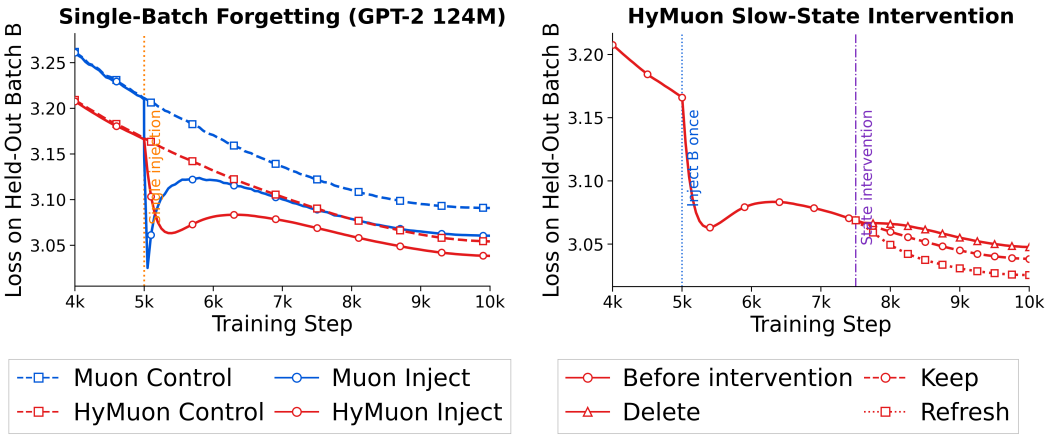


Figure 2: Separating lasting parameter effects from continued use of optimizer state. Left: held-out loss after one exposure versus an excluded-batch control. Right: Keep, Delete, and Refresh applied to the injected batch's contribution stored in `HyMuon`'s slow momentum. At intervention, parameters, the averaged iterate, and all other optimizer states are held fixed; Refresh substitutes a current gradient with matched groupwise norm. `HyMuon` curves are evaluated at the averaged iterate, whereas Muon curves are evaluated at the training iterate; all curves are unsmoothed. Full protocols are in Appendices E.3 and E.3.1.

optimizer history. For each of ten semantic targets, a frozen E5 encoder (Wang et al., 2022) ranks candidates by relevance. Each target consists of semantically related held-out text fragments. We measure loss reduction on separate fragments from the same target. Ten targets formed by randomly mixing fragments serve as controls. Appendix E.2 gives the construction and statistics.

Target-specific improvement subtracts a candidate's mean loss reduction on the other targets. Across all three checkpoints, E5 similarity is positively associated with target-specific improvement for all ten semantic targets. This association strengthens from early to middle training and remains similar late in training. The mixed controls have mean raw improvements near zero at all three checkpoints. The most similar 20% of candidates also account for about one third of the nonnegative target-specific improvement at each checkpoint. Appendix E.2 reports the exact statistics and target-wise distributions. Figure 3 shows how improvement changes across the similarity ranking.

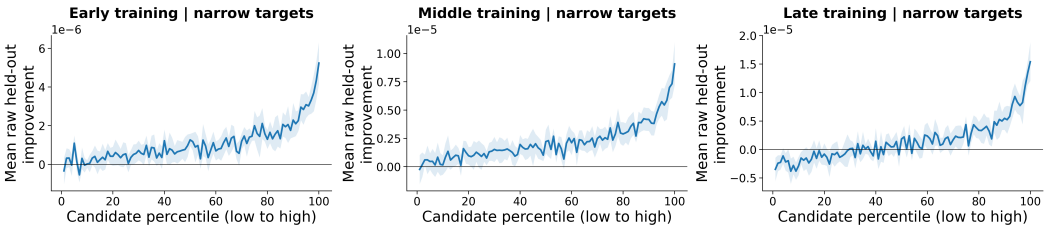


Figure 3: Raw held-out improvement for the semantic targets across 100 E5-similarity percentiles at 10k, 50k, and 100k checkpoints (left to right). Each target contributes 20 candidates per percentile; blue curves average over ten targets, with $\pm 1.96$ standard-error bands across targets. Positive values indicate loss reduction; vertical scales differ. These panels use the same semantic-target measurements from the complete plot in Appendix Figure 9.

Together, the optimizer-state intervention and these measurements suggest why retaining gradients across batches can help: an earlier contribution can remain beneficial after many subsequent steps, while useful contributions arrive unevenly. The weights in Section 4.1 provide a mechanism: aligned gradients can accumulate in the slow momentum, while the fast momentum remains responsive when the useful direction changes.

## 5 CONVERGENCE ANALYSIS

As shown in Section 4.1, the gradients in HyM's slow momentum have an average lag of $(t-1)/3$ iterations, so the convergence analysis must allow the memory horizon to grow with training. We therefore study a general representation in which each iterate depends on all earlier stochastic gradients through deterministic, possibly time-varying cumulative gradient weights. Theorem 1 gives a convergence bound for the averaged iterate when the sum of the squared deviations of the gradient weights from a constant reference is uniformly bounded. We verify this condition for HyM and obtain Corollary 1.

Let us denote $\mathbf{v}_t = \frac{\mathbf{y}_t - \beta \mathbf{x}_t}{1-\beta}$. Relation (5) gives $\mathbf{v}_t = t\mathbf{x}_t - (t-1)\mathbf{x}_{t-1}$, implying that $\mathbf{x}_t$ is the average of $\mathbf{v}_1, \dots, \mathbf{v}_t$. Expanding the recurrences gives $\mathbf{v}_t = \mathbf{v}_1 - \eta \sum_{k=1}^{t-1} K_{t,k}\mathbf{g}_k$. Here $K_{t,k}$ is the cumulative weight of gradient $\mathbf{g}_k$ in the expansion of the reconstructed iterate $\mathbf{v}_t$, with the common factor $-\eta$ removed. It combines that gradient's contributions across updates, whereas $w_{t,k}$ in (8) is its weight in the single step-$t$ update direction. For a fixed reference weight $\Gamma > 0$, define the deviation $e_{t,k} = K_{t,k} - \Gamma$. We consider the general form

$$\begin{aligned} \mathbf{x}_t &= \frac{1}{t}\sum_{i=1}^{t} \mathbf{v}_i, \qquad \mathbf{y}_t = (1-\zeta_t)\mathbf{v}_t + \zeta_t \mathbf{x}_t, \\ \mathbf{v}_t &= \mathbf{v}_1 - \eta \sum_{k=1}^{t-1} (\Gamma + e_{t,k})\mathbf{g}_k. \end{aligned} \tag{9}$$

The coefficients $e_{t,k}$ are deterministic, and $\zeta_t \in [0,1]$ is chosen using information available before the mini-batch at iteration $t$ is sampled. Setting $e_{t,k} = 0$, $\Gamma = 1$, and $\zeta_t = \beta$ reduces (9) to Schedule-Free SGD (Defazio et al., 2024), with gradients evaluated at $\mathbf{y}_t$.

**Assumption 1.** *The objective $f : \mathbb{R}^d \to \mathbb{R}$ is convex, admits a minimizer $\mathbf{x}^*$, and is $L$-smooth for some $L > 0$: for all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, $\|\nabla f(\mathbf{u}) - \nabla f(\mathbf{v})\| \le L\|\mathbf{u} - \mathbf{v}\|$.*

**Assumption 2.** *Let $(\mathcal{F}_t)_{t\ge 0}$ be the filtration of the training history, where $\mathcal{F}_{t-1}$ contains the information available before sampling $\mathcal{B}_t$. The training iterate $\mathbf{y}_t$ is $\mathcal{F}_{t-1}$-measurable, and $\mathbf{g}_t$ is $\mathcal{F}_t$-measurable. Conditional on $\mathcal{F}_{t-1}$, the batch contains $B$ independent, identically distributed samples. The stochastic gradient from each sample has mean $\nabla f(\mathbf{y}_t)$ and expected squared distance from that mean at most $\sigma^2$ for some finite $\sigma \ge 0$. Their average satisfies*

$$\mathbb{E}[\mathbf{g}_t \mid \mathcal{F}_{t-1}] = \nabla f(\mathbf{y}_t), \qquad \mathbb{E}[\|\mathbf{g}_t - \nabla f(\mathbf{y}_t)\|^2 \mid \mathcal{F}_{t-1}] \le \varsigma^2 := \sigma^2/B. \tag{10}$$

We now state the main convergence guarantee for the cumulative-gradient framework under these assumptions.

**Theorem 1.** *Under Assumptions 1 and 2, suppose* (9) *holds with deterministic $\mathbf{v}_1$ and $e_{t,k}$, and with each $\zeta_t \in [0,1]$ determined by $\mathcal{F}_{t-1}$. Assume that $\sup_{t\ge 1}\sum_{k=1}^{t-1} e_{t,k}^2 \le C_E < \infty$ for some $C_E > 0$. For a run of $T$ iterations, set $\eta = \eta_0/\sqrt{T}$, where $\eta_0 \in (0, [2L(\Gamma + 2\sqrt{C_E})]^{-1}]$. Then*

$$\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] \le \frac{\|\mathbf{v}_1 - \mathbf{x}^*\|^2/(2\eta_0\Gamma) + \eta_0\Gamma\varsigma^2/2}{\sqrt{T}} + \frac{2L\eta_0^2 C_E \varsigma^2}{T}. \tag{11}$$

For fixed momentum parameters, HyM satisfies the uniform square-sum condition in Theorem 1, yielding the following convergence guarantee.

**Corollary 1.** *Under Assumptions 1 and 2, consider Algorithm 1 with deterministic $\mathbf{y}_1$ and fixed $\mu \in (0,1)$ and $\beta \in [0,1)$. Choose $\eta = \eta_0/\sqrt{T}$, where $\eta_0 > 0$ is sufficiently small, depends only on $L$, $\mu$, and $\beta$, and is independent of $T$. Then $\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] = \mathcal{O}(T^{-1/2})$.*

For fixed $\eta_0$, $\mu$, and $\beta$, the constants are independent of $T$. They may increase as $\mu$ or $\beta$ approaches one. The result requires neither strong convexity nor uniformly bounded gradients.

## 6 EMPIRICAL STUDIES

We evaluate HyM and HyMuon on image classification, translation, and language modeling. In the experiments, we refer to the step size $\eta$ as the learning rate. Figures 4 and 5 show the main learning

curves. Full configurations, evaluation protocols, and a numerical summary (Table 1) are given in Appendix F. CIFAR, SVHN, and IWSLT report test metrics; ImageNet reports accuracy on its official validation split. Vision curves show means over repeated runs with one-SE bands and sample SDs in the legends; IWSLT curves show repeated-run means with one-SE bands.

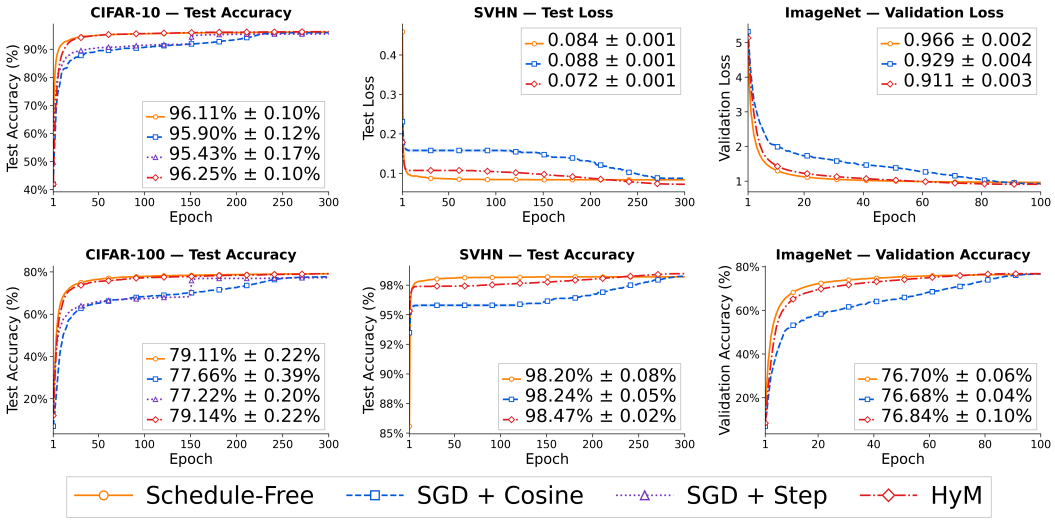


Figure 4: Image classification. Left: CIFAR-10 and CIFAR-100 test accuracy (top/bottom). Middle: SVHN test loss and accuracy. Right: ImageNet-1K validation loss and accuracy. Curves show run means with SE bands; legends report SD. CIFAR test-loss curves are in Appendix Figure 11.

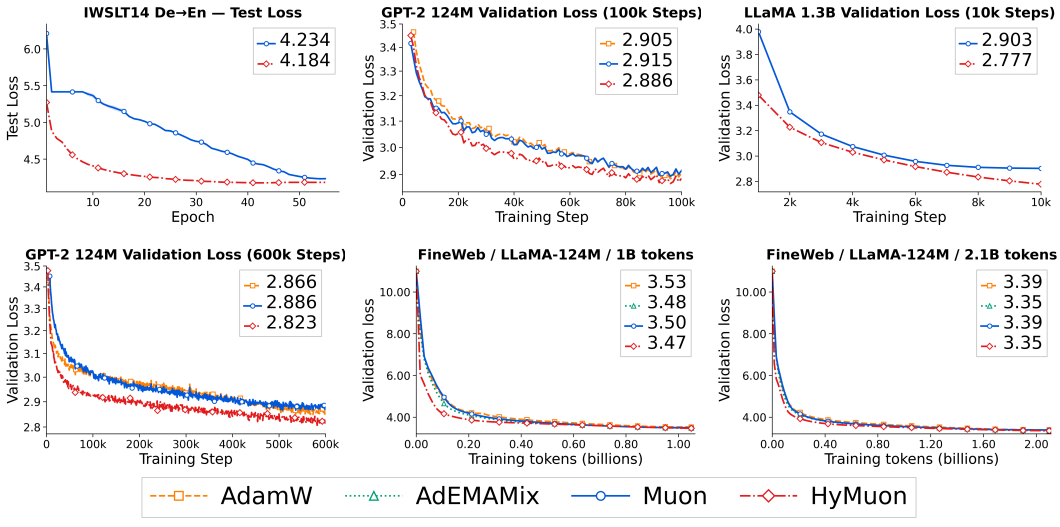


Figure 5: Language tasks. Top: IWSLT14 German–English test loss, OpenWebText validation loss at 100k steps, and packed-C4 validation loss at 10k steps in a comparison against the reproduced Muon baseline. Bottom: OpenWebText validation loss at 600k steps and FineWeb validation loss at the 1B- and 2.1B-token budgets. IWSLT curves show repeated-run means with SE bands. BLEU and C4 perplexity are in Appendix Figures 14 and 16.

### 6.1 Image classification

We compare `HyM` with momentum SGD (Cosine/Step schedules) and Schedule-Free on CIFAR-10 (WRN-16-8), CIFAR-100 (DenseNet-BC), SVHN (ResNet-3-96), and ImageNet-1K (ResNet-50)

(Krizhevsky, 2009; Netzer et al., 2011; Russakovsky et al., 2015). HyM is evaluated at its averaged iterate, as described in Appendix F.

HyM reaches mean test accuracy of $96.25\%$ on CIFAR-10 and $98.47\%$ on SVHN, compared with $96.11\%$ and $98.24\%$ for the strongest corresponding baseline means (Table 1). On CIFAR-100, HyM and Schedule-Free are close at $79.14\%$ and $79.11\%$. ImageNet mean validation top-1 accuracies are $76.84\%$ for HyM, $76.70\%$ for Schedule-Free, and $76.68\%$ for cosine SGD. Schedule-Free makes faster early progress on ImageNet (Figure 4, ImageNet panels).

### 6.2 Language tasks

**IWSLT14 machine translation.** On IWSLT14 German–English (Cettolo et al., 2014), we train Fairseq's attention-based LSTM preset (Ott et al., 2019) for 55 epochs. We search the hidden-matrix and auxiliary learning rates, weight decay, and optimizer-specific momentum, mixing, and auxiliary second-moment coefficients, then evaluate the selected configurations over repeated runs. Test metrics are reported at the checkpoint with the highest validation BLEU in each run. Across these runs, HyMuon attains mean test loss $4.184 \pm 0.004$ and BLEU $33.874 \pm 0.072$, compared with $4.234 \pm 0.006$ and $33.424 \pm 0.048$ for Muon (Table 1; Figure 5, top left; Figure 14 in Appendix F.4).

**GPT-2 124M on OpenWebText.** We pretrain GPT-2 124M (Radford et al., 2019) from scratch on OpenWebText (Gokaslan & Cohen, 2019) for 100k and 600k steps. HyMuon reaches final validation losses of $2.886$ and $2.823$, respectively, lower than AdamW and Muon at both training lengths (Figure 5, top middle and bottom left; full results in Table 1).

**LLaMA-124M on FineWeb: comparison with AdEMAMix.** We compare AdamW, AdEMAMix, Muon, and HyMuon on a common FineWeb subset using the 124M large-batch protocol of Semenov et al. (2025), with separate 1B- and 2.1B-token runs. At 1B tokens, HyMuon reaches final validation loss $3.47$, compared with $3.48$ for AdEMAMix. At 2.1B tokens, HyMuon and AdEMAMix are close, both rounding to $3.35$, versus $3.39$ for Muon and AdamW (Figure 5). Protocol details are in Appendix F.7.

**LLaMA-1.3B on C4.** We compare HyMuon against Muon reproduced with the hyperparameters selected by Gu & Xie (2026) for LLaMA-1.3B/C4. We search the HyMuon optimizer hyperparameters and report the selected configuration. Training uses document-packed C4/en for 10k steps. HyMuon reaches validation loss $2.777$, compared with $2.903$ for Muon (Figure 5, top right). The corresponding perplexity curves are reported in Figure 16 in Appendix F.6.

## 7 Conclusion

HyM combines slow and fast momentum and returns the final averaged iterate. In the slow-momentum intervention, keeping a batch's contribution in HyMuon's slow momentum gives a lower loss on that batch than removing it; replacing it with a current gradient of the same norm lowers the loss further. Under the conditions in Section 5, HyM achieves an $\mathcal{O}(T^{-1/2})$ bound on the expected function-value gap at its final averaged iterate. HyMuon applies Newton–Schulz orthogonalization to the combined momentum before updating matrix parameters. Appendix C.5 also analyzes mixtures of fixed-decay EMAs. The same analysis and intervention protocol can be used to study combinations of more than two momentum states with distinct horizons.

### AI use statement

Generative AI tools were used to assist with manuscript editing, LaTeX organization, mathematical and code review, and research-code development. All AI-assisted material was reviewed and revised by the authors. The authors take responsibility for the final text, claims, proofs, experiments, code, and associated materials.

### Reproducibility statement

The algorithm definitions and implementation variants are specified in the main text and Appendix. The Appendix gives the theoretical assumptions and proofs, experimental configurations, and evaluation protocols.

## References


Kwangjun Ahn, Byron Xu, Natalie Abreu, Ying Fan, Gagik Magakyan, Pratyusha Sharma, Zheng Zhan, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign. In *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, pp. 2–17, 2014.

Marina Danilova. On the convergence analysis of aggregated heavy-ball method. In *Mathematical Optimization Theory and Operations Research (MOTOR 2022)*, volume 13367 of *Lecture Notes in Computer Science*, pp. 3–17. Springer, 2022. doi: 10.1007/978-3-031-09607-5_1.

Aaron Defazio. ScheduleFree+: Scaling learning-rate-free and schedule-free learning to large language models. *arXiv preprint arXiv:2605.19095*, 2026.

Aaron Defazio, Xingyu Alice Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Saeed Ghadimi and Guanghui Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1–2):59–99, 2016. doi: 10.1007/s10107-015-0871-8.

Aaron Gokaslan and Vanya Cohen. OpenWebText corpus. `https://skylion007.github.io/OpenWebTextCorpus/`, 2019.

Yufei Gu and Zeke Xie. Mano: Restriking manifold optimization for LLM training. *arXiv preprint arXiv:2601.23000*, 2026.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1842–1850. PMLR, 2018. URL `https://proceedings.mlr.press/v80/gupta18a.html`.

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.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. Error feedback fixes SignSGD and other gradient compression schemes. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3252–3261, 2019.

Andrej Karpathy. nanoGPT. `https://github.com/karpathy/nanoGPT`, 2022.

Kimi Team. Kimi K2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

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

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

Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1–2):365–397, 2012. doi: 10.1007/s10107-010-0434-y.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. NorMuon: Making Muon more efficient and scalable. *arXiv preprint arXiv:2510.05491*, 2025.

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

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

Paul-Aymeric McRae, Prasanna Parthasarathi, Mahmoud Assran, and Sarath Chandar. Memory augmented optimizers for deep learning. In *International Conference on Learning Representations*, 2022.

Depen Morwani, Nikhil Vyas, Hanlin Zhang, and Sham Kakade. Connections between Schedule-Free optimizers, AdEMAMix, and accelerated SGD variants. *arXiv preprint arXiv:2502.02431*, 2025.

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

Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103 (1):127–152, 2005. doi: 10.1007/s10107-004-0552-5.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, pp. 48–53, 2019. doi: 10.18653/v1/N19-4009.

Matteo Pagliardini, Pierre Ablin, and David Grangier. The AdEMAMix optimizer: Better, faster, older. In *International Conference on Learning Representations*, 2025.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. doi: 10.1016/0041-5553(64)90137-5.

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

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. Benchmarking optimizers for large language model pretraining. *arXiv preprint arXiv:2509.01440*, 2025.

Minhak Song, Beomhan Baek, Kwangjun Ahn, and Chulhee Yun. Through the river: Understanding the benefit of Schedule-Free methods for language model training. In *Advances in Neural Information Processing Systems*, 2025.

Sebastian U. Stich and Sai Praneeth Karimireddy. The error-feedback framework: SGD with delayed gradients. *Journal of Machine Learning Research*, 21(237):1–36, 2020.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In Sanjoy Dasgupta and David McAllester (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IDxZhXrpNf.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

Zimeng Wang and Alp Yurtsever. Generalized stochastic gradient descent with momentum methods for smooth optimization. *arXiv preprint arXiv:2602.23444*, 2026.

Tianbao Yang, Qihang Lin, and Zhe Li. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *arXiv preprint arXiv:1604.03257*, 2016.

Chung-Yiu Yau, Dawei Li, Athanasios Glentis, Valentyn Boreiko, Hoi-To Wai, and Mingyi Hong. EMA-Nesterov: Stabilizing Nesterov's lookahead for accelerated deep learning optimization. *arXiv preprint arXiv:2605.25395*, 2026.

# Appendix


**A Reduction of Schedule-Free to a single-sequence form** . . . . . 14

**B Historical-gradient weighting and averaging geometry** . . . . . 14

B.1 Weights on early and recent gradients . . . . . 14
B.2 Stationary-oracle filtering comparison . . . . . 16
B.3 Response to a persistent change in gradient mean . . . . . 17
B.4 Geometry of averaging along a curved valley . . . . . 17

**C Convergence analysis via cumulative gradient weights** . . . . . 19

C.1 Geometry of the averaged iterate and the Bregman term . . . . . 19
C.2 A finite-horizon theorem for cumulative gradient weights . . . . . 20
C.3 Bounding cumulative gradient weight deviations for HyM . . . . . 21
C.4 Convergence of the averaged iterate for HyM . . . . . 22
C.5 Finite-EMA recovery and relation to other analyses . . . . . 23

**D HyMuon matrix updates and implementation** . . . . . 23

**E Diagnostic experiment details** . . . . . 24

E.1 Batch-size dependence of the HyM advantage . . . . . 24
E.2 Semantic relevance and target-specific improvement . . . . . 25
E.3 Single-batch forgetting diagnostic . . . . . 27

**F Benchmark experiment details** . . . . . 28

F.1 Image-classification runs . . . . . 29
F.2 HyM at the training and averaged iterates . . . . . 31
F.3 Component ablation of HyM on CIFAR-10 . . . . . 33
F.4 IWSLT14 German–English . . . . . 35
F.5 NanoGPT on OpenWebText . . . . . 36
F.6 LLaMA-1.3B pretraining on C4 . . . . . 37
F.7 LLaMA-124M FineWeb benchmark . . . . . 37


**Structure of the Appendix.** Appendix A records the Schedule-Free identity used throughout the analysis. Appendix B collects the historical-gradient weighting, filtering, and curved-valley calculations that support the discussion of gradient-history effects. Appendix C develops the convergence analysis via cumulative gradient weights and its specialization to HyM. Appendix D presents HyMuon's matrix updates and full-model implementation. Finally, Appendix E documents the diagnostic experiments, while Appendix F records benchmark configurations and implementation details.

# A Reduction of Schedule-Free to a single-sequence form

The Schedule-Free method of Defazio et al. (2024) is originally stated with three sequences: the averaged iterate $\mathbf{x}_t$, the base iterate $\mathbf{z}_t$, and the training iterate $\mathbf{y}_t$,

$$\mathbf{y}_t = (1-\beta)\mathbf{z}_t + \beta\mathbf{x}_t, \tag{12}$$

$$\mathbf{z}_{t+1} = \mathbf{z}_t - \eta\,\mathbf{g}_t, \quad \mathbf{g}_t = \frac{1}{B}\sum_{j=1}^{B}\nabla f(\mathbf{y}_t;\xi_{t,j}), \tag{13}$$

$$\mathbf{x}_{t+1} = (1-c_{t+1})\mathbf{x}_t + c_{t+1}\mathbf{z}_{t+1}, \tag{14}$$

initialized at $\mathbf{z}_1 = \mathbf{x}_1$, with averaging weights $c_{t+1}$ (for a constant step size, $c_{t+1} = 1/(t+1)$). The single-sequence form (2) follows from the change of variable

$$\mathbf{n}_t := \frac{\mathbf{x}_t - \mathbf{z}_{t+1}}{\eta}, \tag{15}$$

which measures the (rescaled) gap between the averaged and the base sequences.

Using $\mathbf{x}_t = (1-c_t)\mathbf{x}_{t-1} + c_t\mathbf{z}_t$ and $\mathbf{z}_{t+1} = \mathbf{z}_t - \eta\,\mathbf{g}_t$, we have

$$\mathbf{n}_t = \frac{\mathbf{x}_t - \mathbf{z}_{t+1}}{\eta} = \frac{(1-c_t)(\mathbf{x}_{t-1} - \mathbf{z}_t) + \eta\,\mathbf{g}_t}{\eta} = (1-c_t)\mathbf{n}_{t-1} + \mathbf{g}_t. \tag{16}$$

Since $\mathbf{z}_1 = \mathbf{x}_1$, (15) gives $\mathbf{n}_1 = (\mathbf{x}_1 - \mathbf{z}_2)/\eta = \mathbf{g}_1$, consistent with $c_1 = 1$.

Differencing (12) and using $\mathbf{z}_{t+1} - \mathbf{z}_t = -\eta\,\mathbf{g}_t$, $\mathbf{x}_{t+1} - \mathbf{x}_t = c_{t+1}(\mathbf{z}_{t+1} - \mathbf{x}_t)$, and $\mathbf{z}_{t+1} - \mathbf{x}_t = -\eta\,\mathbf{n}_t$ from (15), we obtain

$$\begin{aligned}\mathbf{y}_{t+1} - \mathbf{y}_t &= (1-\beta)(\mathbf{z}_{t+1} - \mathbf{z}_t) + \beta(\mathbf{x}_{t+1} - \mathbf{x}_t)\\ &= -(1-\beta)\eta\,\mathbf{g}_t + \beta\,c_{t+1}(\mathbf{z}_{t+1} - \mathbf{x}_t)\\ &= -\eta\big[(1-\beta)\mathbf{g}_t + \beta\,c_{t+1}\mathbf{n}_t\big],\end{aligned} \tag{17}$$

which is exactly (2).

For $c_t = 1/t$, the recurrence has the closed form $\mathbf{n}_t = \frac{t-1}{t}\mathbf{n}_{t-1} + \mathbf{g}_t = \sum_{i=1}^{t}\frac{i}{t}\mathbf{g}_i$, whose weights grow linearly with $i$. This is the same linear weighting pattern used in the estimate-sequence construction of Nesterov's accelerated methods (Nesterov, 1983; 2005). `HyM` uses the same recurrence after the rescaling $\mathbf{n}_t = (t+1)\mathbf{m}_t/2$, but scales its contribution by the geometric sum $(1-\mu^t)/(1-\mu)$ and replaces the contribution of the current gradient with its fast momentum. Its direction is $\beta(1-\mu^t)\mathbf{n}_t/[t(1-\mu)] + (1-\beta)\mathbf{b}_t$, where $\mathbf{m}_t$ denotes the slow momentum in Algorithm 1.

# B Historical-gradient weighting and averaging geometry

## B.1 Weights on early and recent gradients

We give the calculations underlying Section 4.1, for $c_t = 1/t$, fixed $\mu \in (0,1)$, and $\beta \in [0,1)$. In this appendix we use $q_t = (1-\mu^t)/(1-\mu)$ for the recurring geometric sum.

**Kernel decomposition and normalization.** Unrolling the two momentum states gives

$$\mathbf{m}_t = \sum_{i=1}^{t}\frac{2i}{t(t+1)}\mathbf{g}_i, \qquad \mathbf{b}_t = \sum_{i=1}^{t}\frac{i}{t}\mu^{t-i}\mathbf{g}_i. \tag{18}$$

Substitution into Algorithm 1, line 6, yields the direction weights (8). Define the fast mass and the two component masses by

$$Z_t = \sum_{i=1}^{t}\frac{i}{t}\mu^{t-i}, \qquad A_t = \beta\alpha_t, \qquad C_t = (1-\beta)Z_t. \tag{19}$$

The total coefficient sum is $S_t = \sum_{i=1}^{t} w_{t,i} = A_t + C_t$, with

$$S_1 = 1, \qquad S_t \longrightarrow \frac{1-\beta/2}{1-\mu}. \tag{20}$$

To separate the distribution over past gradients from this total coefficient sum, use sum normalization,

$$p_{t,i} = \frac{w_{t,i}}{S_t} = \frac{w_{t,i}}{\sum_{j=1}^{t} w_{t,j}}, \qquad \sum_{i=1}^{t} p_{t,i} = 1. \tag{21}$$

Thus $p_{t,i}$ is the fraction of the direction's total scalar gradient weight assigned to $\mathbf{g}_i$. It has the exact mixture representation

$$p_{t,i} = \lambda_t \underbrace{\frac{2i}{t(t+1)}}_{U_{t,i}} + (1-\lambda_t) \underbrace{\frac{(i/t)\mu^{t-i}}{Z_t}}_{V_{t,i}}, \qquad \lambda_t = \frac{A_t}{A_t + C_t}. \tag{22}$$

Both $U_{t,i}$ and $V_{t,i}$ sum to one. The proposition below quantifies the aggregate weight on an early fraction of history and a fixed recent window. It concerns the distribution of weight only and says nothing about the usefulness of the retained directions.

**Proposition B.1.** *Fix $\mu \in (0,1)$ and $\beta \in (0,1)$. Then*

$$\lambda_t \longrightarrow \frac{\beta}{2-\beta}, \qquad 1-\lambda_t \longrightarrow \rho := \frac{2(1-\beta)}{2-\beta}.$$

*For every fixed $a \in (0,1)$ and positive integer $K$,*

$$\lim_{t\to\infty} \sum_{i=1}^{\lfloor at \rfloor} p_{t,i} = \frac{\beta}{2-\beta} a^2, \tag{23}$$

$$\lim_{t\to\infty} \sum_{i=t-K+1}^{t} p_{t,i} = \rho(1-\mu^K). \tag{24}$$

*Proof.* Recall that

$$\alpha_t = \frac{t+1}{2t} q_t, \qquad Z_t = \sum_{i=1}^{t} \frac{i}{t} \mu^{t-i}.$$

Since $q_t \to (1-\mu)^{-1}$ and, after setting $k = t - i$,

$$Z_t = \sum_{k=0}^{t-1} \left(1 - \frac{k}{t}\right) \mu^k \longrightarrow \frac{1}{1-\mu},$$

we obtain

$$A_t \to \frac{\beta}{2(1-\mu)}, \qquad C_t \to \frac{1-\beta}{1-\mu},$$

and hence $\lambda_t \to \beta/(2-\beta)$ and $1-\lambda_t \to \rho = 2(1-\beta)/(2-\beta)$.

Let $k_t = \lfloor at \rfloor$. The slow-momentum component satisfies

$$\sum_{i=1}^{k_t} U_{t,i} = \frac{k_t(k_t+1)}{t(t+1)} \longrightarrow a^2.$$

For the fast-momentum component, $Z_t \geq 1$ and

$$0 \leq \sum_{i=1}^{k_t} V_{t,i} \leq \sum_{k=t-k_t}^{\infty} \mu^k = \frac{\mu^{t-k_t}}{1-\mu} \longrightarrow 0.$$

Combining these limits with the mixture representation (22) proves (23).

For a fixed positive integer $K$,

$$\sum_{i=t-K+1}^{t} U_{t,i} \longrightarrow 0, \qquad \sum_{i=t-K+1}^{t} V_{t,i} = \frac{\sum_{k=0}^{K-1}(1-k/t)\mu^k}{Z_t} \longrightarrow 1-\mu^K.$$

The same mixture representation therefore gives (24). □

**Emergence of the recent-gradient peak.** At short horizons, the component shapes nearly coincide when $t(1-\mu) \ll 1$, and the combined profile is approximately a ramp. At longer horizons, the fast-momentum weights become concentrated within $\mathcal{O}((1-\mu)^{-1})$ steps, while the slow-momentum weights span $\mathcal{O}(t)$ steps. For $0 < \beta < 1$, the ratio of the fast to slow contribution at lag $k = t - i$ is $(1-\beta)t\mu^k/(\beta q_t)$. A ramp can remain prominent beyond the overlapping-memory regime when the fast contribution is still small relative to it. Once $q_t$ is near its limit, the newest-index ratio reaches one around

$$t_{\text{head}} \simeq \frac{\beta}{(1-\beta)(1-\mu)}. \tag{25}$$

This gradual separation produces a recent-gradient peak above the broad ramp. A larger relative peak need not mean a larger absolute normalized weight on the newest gradient. Moreover, the slow momentum retains mass $\beta/(2-\beta)$ even at long horizons; its contribution remains substantial whenever $\beta$ is large.

For a finite mixture of EMAs with fixed decay coefficients below one, the mass on $i \le at$, for fixed $a \in (0,1)$, vanishes as $t$ grows. Each EMA tail is bounded by a geometrically decreasing term, whereas `HyM` retains the positive aggregate limit in (23).

### B.2 Stationary-oracle filtering comparison

**Proposition B.2.** *Fix $\mu \in (0,1)$ and $\beta \in [0,1)$. Suppose $\mathbf{g}_i = \mathbf{g} + \boldsymbol{\varepsilon}_i$, where the noises are independent, zero mean, and have covariance $\Sigma/B$. Let $\widetilde{\boldsymbol{\Delta}}_t = \sum_{i=1}^t p_{t,i}\mathbf{g}_i$, $\rho = 2(1-\beta)/(2-\beta)$, and $\chi = \rho(1-\mu)$. Then $\widetilde{\boldsymbol{\Delta}}_t$ is unbiased, $p_{t,t} \to \chi$, and*

$$\mathrm{Cov}(\widetilde{\boldsymbol{\Delta}}_t) \longrightarrow \rho^2 \frac{1-\mu}{1+\mu}\frac{\Sigma}{B}. \tag{26}$$

*For $0 < \beta < 1$, its limiting scalar covariance factor is strictly below that of a normalized EMA with newest-gradient weight $\chi$:*

$$\rho^2 \frac{1-\mu}{1+\mu} < \frac{\chi}{2-\chi}.$$

*Proof.* Define

$$Z_t := \sum_{i=1}^t \frac{i}{t}\mu^{t-i}, \qquad S_t := \sum_{i=1}^t w_{t,i} = \frac{\beta q_t(t+1)}{2t} + (1-\beta)Z_t.$$

The normalized weights can be decomposed as

$$\frac{w_{t,i}}{S_t} = \lambda_t U_{t,i} + (1-\lambda_t)V_{t,i},$$

where

$$U_{t,i} = \frac{2i}{t(t+1)}, \qquad V_{t,i} = \frac{(i/t)\mu^{t-i}}{Z_t}, \qquad \lambda_t = \frac{\beta q_t(t+1)/(2t)}{S_t}, \qquad 1-\lambda_t = \frac{(1-\beta)Z_t}{S_t}.$$

Both $U_t$ and $V_t$ are probability vectors. Since $q_t \to 1/(1-\mu)$ and $Z_t \to 1/(1-\mu)$,

$$1-\lambda_t \longrightarrow \rho = \frac{2(1-\beta)}{2-\beta}, \qquad \lambda_t \longrightarrow 1-\rho.$$

Furthermore,

$$\sum_{i=1}^t U_{t,i}^2 = \frac{2(2t+1)}{3t(t+1)} \longrightarrow 0, \qquad \sum_{i=1}^t V_{t,i}^2 \longrightarrow \frac{1-\mu}{1+\mu}.$$

The cross term $\sum_{i=1}^t U_{t,i}V_{t,i}$ tends to zero by Cauchy–Schwarz. Independence of the noise therefore gives

$$\mathrm{Cov}(\widetilde{\boldsymbol{\Delta}}_t) = \left(\sum_{i=1}^t \frac{w_{t,i}^2}{S_t^2}\right)\frac{\Sigma}{B} \longrightarrow \rho^2 \frac{1-\mu}{1+\mu}\frac{\Sigma}{B}.$$

The newest normalized weight is

$$\frac{\beta q_t/t + 1 - \beta}{S_t} \longrightarrow \rho(1-\mu) = \chi.$$

A normalized EMA with newest weight $\chi$ has weights $\chi(1-\chi)^k$. Its scalar covariance factor is

$$\chi^2 \sum_{k=0}^{\infty} (1-\chi)^{2k} = \frac{\chi}{2-\chi}.$$

Writing $a = 1-\mu$, the difference between the matched-EMA and HyM factors is

$$\frac{\rho a}{2-\rho a} - \frac{\rho^2 a}{2-a} = \frac{\rho a(1-\rho)\,[2-a(1+\rho)]}{(2-\rho a)(2-a)} > 0$$

for $0 < \rho < 1$ and $0 < a < 1$, proving the proposition. □

### B.3 Response to a persistent change in gradient mean

The matched-newest-weight comparison in Proposition B.2 describes the immediate response to a change in mean. To make the distinction for a persistent change explicit, suppose the deterministic gradient mean is $\mathbf{g}^-$ through step $\tau$ and $\mathbf{g}^+ = \mathbf{g}^- + \mathbf{d}$ afterward. At $t = \tau + k$, the fraction of the normalized HyM direction that responds to $\mathbf{d}$ is

$$H_{t,\tau} = \sum_{i=\tau+1}^{t} p_{t,i} = \lambda_t \left(1 - \frac{\tau(\tau+1)}{t(t+1)}\right) + (1-\lambda_t)\frac{\sum_{j=0}^{k-1}(1-j/t)\mu^j}{Z_t}. \tag{27}$$

For fixed $k$ and $\tau \to \infty$, the slow term has not yet accumulated appreciable mass, and

$$H_{\tau+k,\tau} \longrightarrow \rho(1-\mu^k).$$

A normalized EMA with constant newest weight $\chi = \rho(1-\mu)$, whose initialization bias has vanished, responds by the fraction $1-(1-\chi)^k$. Its immediate response therefore matches HyM's, while for $k > 1$, $0 < \rho < 1$, and $0 < \mu < 1$,

$$1 - \big(1 - \rho(1-\mu)\big)^k > \rho(1-\mu^k).$$

The inequality follows from strict concavity of $h(a) = 1-(1-a)^k$ on $(0,1)$. Matching the newest-gradient weight therefore does not imply matching adaptation to a lasting change: the EMA reacts faster over a fixed recent window, and HyM assigns the remaining mass to its growing-history component. Under the matched-newest-weight comparison, HyM has lower limiting covariance and the EMA adapts faster to a persistent change.

### B.4 Geometry of averaging along a curved valley

#### B.4.1 Rosenbrock illustration and the role of the averaged iterate

Accumulation can reveal a weak direction that appears repeatedly within gradients dominated by larger, oscillating components. For the Rosenbrock function $f_{\mathrm{RB}}$ defined below, the component of $\nabla f_{\mathrm{RB}}$ normal to the curve $v = u^2$ changes sign across iterates, while the tangential component can keep its sign. This geometric motivation also appears in AdEMAMix and Schedule-Free analyses (Pagliardini et al., 2025; Song et al., 2025).

An average preserves the persistent component while cancelling opposite normal contributions. HyM's slow momentum accumulates the former over a growing history, while its fast momentum responds when the direction of the curve changes. Thus uneven usefulness across batches and sign changes of the normal component are two ways in which the most recent batch alone can obscure a useful direction.

We visualize this behavior on $f_{\mathrm{RB}}(u,v) = (1-u)^2 + 100(v-u^2)^2$. Figure 6 shows the training iterates; Figure 7 shows the corresponding averaged iterates.

Gradient memory changes the training trajectory, whereas iterate averaging produces a separate sequence by averaging the training iterates. In HyMuon, the two momentum states are combined before the nonlinear NS transformation, so averaging gradients is also distinct from averaging the resulting directions.

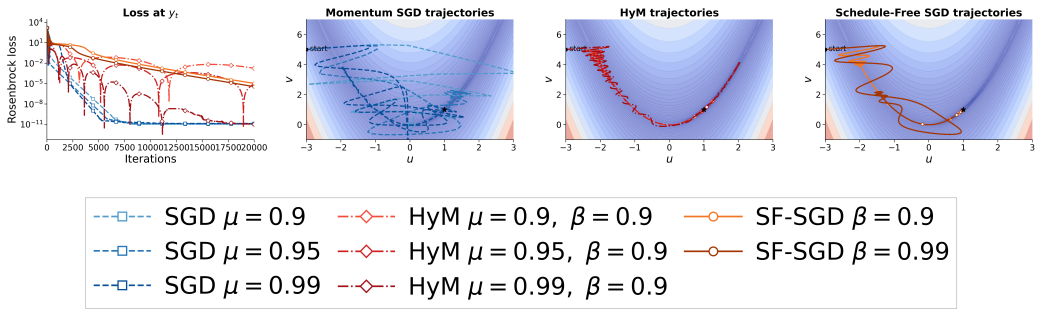


Figure 6: Behavior of the Rosenbrock training iterates from $(-3, 5)$ over 20,000 exact-gradient steps. Momentum SGD uses $\eta = 4 \times 10^{-3}$ and $\mu \in \{0.9, 0.95, 0.99\}$; HyM uses $\eta = 3 \times 10^{-4}$, $\beta = 0.9$, and $\mu \in \{0.9, 0.95, 0.99\}$; Schedule-Free SGD uses $\eta = 7 \times 10^{-4}$ and $\beta \in \{0.9, 0.99\}$. The left panel shows objective values at $\mathbf{y}_t$; the remaining panels show the trajectories over objective contours. The star marks the minimizer $(1, 1)$.

**Why an average can leave a curved valley.** Let $v = \phi(u)$ denote a valley centerline and let $\{(u_i, v_i)\}_{i=1}^N$ be $N$ training iterates with $\pi_i \geq 0$ and $\sum_{i=1}^N \pi_i = 1$. Write

$$\bar{u} = \sum_{i=1}^N \pi_i u_i, \qquad \bar{v} = \sum_{i=1}^N \pi_i v_i.$$

The averaged iterate has the exact centerline residual

$$\bar{v} - \phi(\bar{u}) = \underbrace{\sum_{i=1}^N \pi_i \big(v_i - \phi(u_i)\big)}_{\text{averaged tracking residual}} + \underbrace{\left(\sum_{i=1}^N \pi_i \phi(u_i) - \phi\left(\sum_{i=1}^N \pi_i u_i\right)\right)}_{\text{curvature-induced averaging gap}}. \tag{28}$$

Thus even points lying exactly on the centerline can average to a point away from it. If $\phi$ is twice differentiable and $|\phi''(u)| \leq M$ on the convex hull of the $u_i$, Taylor's theorem gives

$$\left|\sum_{i=1}^N \pi_i \phi(u_i) - \phi(\bar{u})\right| \leq \frac{M}{2} \sum_{i=1}^N \pi_i (u_i - \bar{u})^2. \tag{29}$$

For the Rosenbrock valley centerline $\phi(u) = u^2$, the bound holds with $M = 2$.

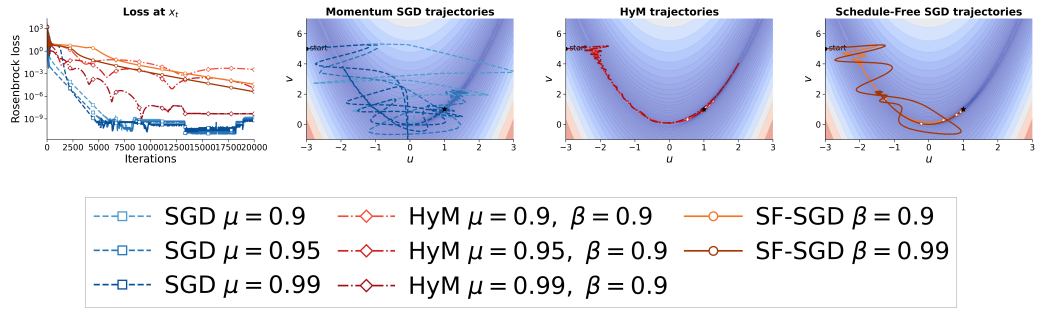


Figure 7: Averaged iterates for the Rosenbrock runs in Figure 6, using the same initialization, optimizer settings, and 20,000 exact-gradient steps. HyM uses $\beta = 0.9$ for iterate averaging; Momentum SGD uses passive iterate averaging with $\beta = 0.99$; Schedule-Free SGD uses its respective $\beta \in \{0.9, 0.99\}$. The left panel shows objective values at $\mathbf{x}_t$; the remaining panels show the averaged trajectories over objective contours. The star marks the minimizer $(1, 1)$.

The CIFAR-10 component ablation separates these roles directly (Appendix F.3). Combining the fast and slow momenta improves performance at the training iterate $\mathbf{y}_t$ relative to using either momentum state alone at all three tested learning rates. Along the same full-HyM trajectories, evaluating

the averaged iterate $\mathbf{x}_t$ adds 2.28–3.60 percentage points of final validation accuracy. Thus, the two momentum states shape the training trajectory, while the averaged iterate $\mathbf{x}_t$ is used for evaluation.

# C Convergence analysis via cumulative gradient weights

Sections C.1–C.4 develop the convex analysis of the averaged iterate, from the Bregman geometry and the generic finite-horizon theorem to the bound on HyM's cumulative gradient weight deviations and convergence result. Section C.5 then shows that the same uniform bound holds for finite EMA mixtures and relates the result to other analyses.

We prove the general result of Theorem 1 and specialize it to HyM. Throughout this appendix, $\eta > 0$ is constant within a run, $\Gamma > 0$, and $\varsigma = \sigma/\sqrt{B} \geq 0$. The initial point is deterministic. All iterates and interpolation coefficients are $\mathcal{F}_{t-1}$-measurable before sampling $\mathbf{g}_t$ at iteration $t$; we call such sequences predictable. We use the conditional mean and variance bounds in (10), implied by Assumption 2, at the training iterate $\mathbf{y}_t$. We write $\boldsymbol{\varepsilon}_t = \mathbf{g}_t - \nabla f(\mathbf{y}_t)$ and $Q_T = \sum_{t=1}^{T} \mathbb{E}\|\nabla f(\mathbf{y}_t)\|^2$. For the dynamics in (9), all iterates and gradients have finite second moments at each finite horizon. This follows by induction from the deterministic initial point and cumulative gradient weights, the bounded interpolation coefficients, the Lipschitz gradient, and the oracle variance bound. The representation of the reconstructed iterates through cumulative gradient weights is exact; weight decay, nonlinear preconditioning, and NS orthogonalization are outside this analysis with deterministic scalar weights.

## C.1 Geometry of the averaged iterate and the Bregman term

For HyM, let $\lambda = 1 - \beta > 0$ and reconstruct

$$\mathbf{v}_t = \frac{\mathbf{y}_t - \beta \mathbf{x}_t}{\lambda}. \tag{30}$$

Equation (5) gives

$$\mathbf{v}_t = t\mathbf{x}_t - (t-1)\mathbf{x}_{t-1}, \qquad \mathbf{x}_t = \frac{1}{t}\sum_{i=1}^{t} \mathbf{v}_i, \qquad \mathbf{y}_t = \lambda \mathbf{v}_t + \beta \mathbf{x}_t. \tag{31}$$

Thus the algorithm's final averaged iterate is the ordinary average of its reconstructed base sequence. The following conversion also permits predictable, time-varying interpolation.

**Lemma C.1.** *Let $f$ be convex and $L$-smooth on $\mathbb{R}^d$, $L > 0$, with minimizer $\mathbf{x}^*$. Suppose $\mathbf{x}_t = t^{-1}\sum_{i=1}^{t} \mathbf{v}_i$ and $\mathbf{y}_t = (1-\zeta_t)\mathbf{v}_t + \zeta_t \mathbf{x}_t$ with predictable $\zeta_t \in [0,1]$ and predictable, square-integrable $\mathbf{v}_t$. Under* (10)*,*

$$T\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] + \frac{Q_T}{2L} \leq \sum_{t=1}^{T} \mathbb{E}\langle \mathbf{g}_t, \mathbf{v}_t - \mathbf{x}^* \rangle. \tag{32}$$

*Proof.* Set $\mathcal{D}_t(\mathbf{u}) = f(\mathbf{u}) - f(\mathbf{y}_t) - \langle \nabla f(\mathbf{y}_t), \mathbf{u} - \mathbf{y}_t \rangle$, $\Psi_t = t(f(\mathbf{x}_t) - f(\mathbf{x}^*))$, and $\Psi_0 = 0$. For $t \geq 2$, the identity $\mathbf{v}_t = t\mathbf{x}_t - (t-1)\mathbf{x}_{t-1}$ yields

$$\langle \nabla f(\mathbf{y}_t), \mathbf{v}_t - \mathbf{x}^* \rangle - (\Psi_t - \Psi_{t-1}) = \mathcal{D}_t(\mathbf{x}^*) + (t-1)\mathcal{D}_t(\mathbf{x}_{t-1}) - t\mathcal{D}_t(\mathbf{x}_t). \tag{33}$$

Moreover,

$$\mathbf{x}_t = (1-\nu_t)\mathbf{y}_t + \nu_t \mathbf{x}_{t-1}, \qquad \nu_t = \frac{(1-\zeta_t)(t-1)}{1 + (1-\zeta_t)(t-1)}, \qquad t\nu_t \leq t-1.$$

Convexity of $\mathcal{D}_t$, with $\mathcal{D}_t(\mathbf{y}_t) = 0$, implies $t\mathcal{D}_t(\mathbf{x}_t) \leq (t-1)\mathcal{D}_t(\mathbf{x}_{t-1})$. At $t = 1$, $\mathbf{v}_1 = \mathbf{x}_1 = \mathbf{y}_1$, so the same conclusion holds with the zero-weighted previous-point term omitted. For an unconstrained minimizer $\nabla f(\mathbf{x}^*) = 0$; the smooth-convex inequality

$$f(\mathbf{u}) \geq f(\mathbf{v}) + \langle \nabla f(\mathbf{v}), \mathbf{u} - \mathbf{v} \rangle + \frac{1}{2L}\|\nabla f(\mathbf{u}) - \nabla f(\mathbf{v})\|^2$$

therefore gives $\mathcal{D}_t(\mathbf{x}^*) \geq \|\nabla f(\mathbf{y}_t)\|^2/(2L)$. Sum (33) and take expectations. Predictability of $\mathbf{v}_t$ permits replacing $\nabla f(\mathbf{y}_t)$ by $\mathbf{g}_t$ in the inner product. □

This retains the negative Bregman term in the equal-weight online-to-batch argument of Schedule-Free (Defazio et al., 2024).

### C.2 A FINITE-HORIZON THEOREM FOR CUMULATIVE GRADIENT WEIGHTS

Suppose the base dynamics in (9) have deterministic weight deviations $e_{t,k} = K_{t,k} - \Gamma$. Define $E_T \in \mathbb{R}^{T\times T}$ by $(E_T)_{t,k} = e_{t,k}$ for $1 \le k < t \le T$, and zero otherwise. Set

$$\mathcal{E}_T = \|E_T\|_{\mathrm{F}}^2 = \sum_{t=1}^{T}\sum_{k=1}^{t-1} e_{t,k}^2, \qquad \chi_T = \left[-\lambda_{\min}\left(\frac{E_T + E_T^\top}{2}\right)\right]_+. \tag{34}$$

Here $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm, $\lambda_{\min}$ denotes the smallest eigenvalue of a symmetric matrix, and $[r]_+ = \max\{r, 0\}$. Acting on a stacked vector sequence, $E_T$ means $E_T \otimes I_d$, where $\otimes$ is the Kronecker product and $I_d$ is the $d \times d$ identity matrix. Its norms are those of the scalar time matrix.

**Theorem C.1.** *Under the objective, geometry, and oracle assumptions of Lemma C.1, suppose* (9) *holds with deterministic* $\mathbf{v}_1$ *and* $0 < \eta \le [2L(\Gamma + 2\chi_T)]^{-1}$. *Then*

$$T\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] + \frac{Q_T}{8L} \le \frac{\|\mathbf{v}_1 - \mathbf{x}^*\|^2}{2\eta\Gamma} + \frac{\eta\Gamma\varsigma^2 T}{2} + 2L\eta^2\varsigma^2\mathcal{E}_T. \tag{35}$$

*Proof.* Define $\mathbf{u}_1 = \mathbf{v}_1$ and $\mathbf{u}_{t+1} = \mathbf{u}_t - \eta\Gamma\mathbf{g}_t$. The standard distance expansion of online gradient descent gives

$$\sum_{t=1}^{T}\mathbb{E}\langle \mathbf{g}_t, \mathbf{u}_t - \mathbf{x}^*\rangle \le \frac{\|\mathbf{v}_1 - \mathbf{x}^*\|^2}{2\eta\Gamma} + \frac{\eta\Gamma}{2}(Q_T + \varsigma^2 T). \tag{36}$$

Here we used $\mathbb{E}\|\mathbf{g}_t\|^2 = \mathbb{E}\|\nabla f(\mathbf{y}_t)\|^2 + \mathbb{E}\|\boldsymbol{\varepsilon}_t\|^2 \le \mathbb{E}\|\nabla f(\mathbf{y}_t)\|^2 + \varsigma^2$, which follows from $\mathbb{E}[\boldsymbol{\varepsilon}_t \mid \mathcal{F}_{t-1}] = 0$. This uses the realized gradients; they need not be unbiased at $\mathbf{u}_t$. The predictable difference between base and reference sequences is

$$\boldsymbol{\delta}_t^v = \mathbf{v}_t - \mathbf{u}_t = -\eta\sum_{k=1}^{t-1} e_{t,k}\mathbf{g}_k. \tag{37}$$

For compact notation in the following matrix calculation, write $\mathbf{h}_t = \nabla f(\mathbf{y}_t)$ and define the stacked vectors $\mathbf{h} = (\mathbf{h}_1^\top, \dots, \mathbf{h}_T^\top)^\top$ and $\boldsymbol{\varepsilon} = (\boldsymbol{\varepsilon}_1^\top, \dots, \boldsymbol{\varepsilon}_T^\top)^\top$ in $\mathbb{R}^{Td}$. Since $\boldsymbol{\delta}_t^v$ is $\mathcal{F}_{t-1}$-measurable, $\mathbb{E}\langle \boldsymbol{\varepsilon}_t, \boldsymbol{\delta}_t^v\rangle = 0$, and hence

$$\sum_{t=1}^{T}\mathbb{E}\langle \mathbf{g}_t, \boldsymbol{\delta}_t^v\rangle = -\eta\mathbb{E}\langle \mathbf{h}, E_T\mathbf{h}\rangle - \eta\mathbb{E}\langle \mathbf{h}, E_T\boldsymbol{\varepsilon}\rangle. \tag{38}$$

The symmetric part of $E_T$ gives $-\mathbb{E}\langle \mathbf{h}, E_T\mathbf{h}\rangle \le \chi_T Q_T$. For $k < \ell$, conditional mean zero gives $\mathbb{E}\langle \boldsymbol{\varepsilon}_k, \boldsymbol{\varepsilon}_\ell\rangle = 0$. Since the coefficients are deterministic,

$$\mathbb{E}\|E_T\boldsymbol{\varepsilon}\|^2 = \sum_{t=1}^{T}\sum_{k=1}^{t-1} e_{t,k}^2\mathbb{E}\|\boldsymbol{\varepsilon}_k\|^2 \le \varsigma^2\mathcal{E}_T. \tag{39}$$

Historical noise can be correlated with later gradients. Cauchy–Schwarz, rather than a zero-expectation argument, bounds its interaction:

$$\sum_{t=1}^{T}\mathbb{E}\langle \mathbf{g}_t, \boldsymbol{\delta}_t^v\rangle \le \eta\chi_T Q_T + \eta\varsigma\sqrt{\mathcal{E}_T Q_T}.$$

Combining with (32) and (36),

$$T\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] + \left(\frac{1}{2L} - \frac{\eta\Gamma}{2} - \eta\chi_T\right)Q_T \le \frac{\|\mathbf{v}_1 - \mathbf{x}^*\|^2}{2\eta\Gamma} + \frac{\eta\Gamma\varsigma^2 T}{2} + \eta\varsigma\sqrt{\mathcal{E}_T Q_T}.$$

The step-size condition leaves at least $Q_T/(4L)$ on the left. Young's inequality $\eta\varsigma\sqrt{\mathcal{E}_T Q_T} \le Q_T/(8L) + 2L\eta^2\varsigma^2\mathcal{E}_T$ proves (35). □

*Proof of Theorem 1.* The row bound implies $\mathcal{E}_T \le C_E T$ and $\chi_T \le \|E_T\|_{\mathrm{op}} \le \sqrt{C_E T}$. For $T \ge 1$, the stated $\eta_0$ ensures $\eta(\Gamma + 2\chi_T) \le \eta_0(\Gamma + 2\sqrt{C_E}) \le 1/(2L)$. Substitute $\eta = \eta_0/\sqrt{T}$ into (35), divide by $T$, and use $\mathcal{E}_T \le C_E T$ in the noise term. Dropping the nonnegative term $Q_T/(8LT)$ gives the stated function-value bound. □

The two quantities control different terms: $\chi_T$ bounds the adverse quadratic interaction of the gradient path with the cumulative weight deviations, whereas $\mathcal{E}_T$ bounds amplification of martingale noise. For equal-weight Schedule-Free SGD, $E_T = 0$, $\Gamma = 1$, and both penalties vanish.

### C.3 Bounding cumulative gradient weight deviations for HyM

Fix $\mu \in (0,1)$, $\beta \in [0,1)$, $\lambda = 1-\beta$, and $\Gamma = (1-\mu)^{-1}$. Write $q_t = (1-\mu^t)/(1-\mu)$ and $a_t = [1+\lambda(t-1)]^{-1}$, as in Algorithm 1. For $1 \le i \le t$, set

$$\Pi_{i+1,t} = \prod_{j=i+1}^{t} (1-a_j), \qquad \omega_{i,t} = 1 + \frac{\beta}{\lambda}\Pi_{i+1,t},$$

where empty products equal one. Define

$$C_H = \frac{\beta\Gamma(1+\Gamma)+\mu\Gamma^2}{\lambda}, \qquad C_E^{\mathrm{HyM}} = \frac{\pi^2 C_H^2}{3} + \frac{2\Gamma^2\mu^2}{1-\mu^2}. \tag{40}$$

**Lemma C.2.** *The reconstructed sequence has the exact representation*

$$\mathbf{v}_t = \mathbf{y}_1 - \eta \sum_{k=1}^{t-1} K_{t,k}\mathbf{g}_k, \tag{41}$$

$$K_{t,k} = \sum_{i=k}^{t-1} \omega_{i,t}\left(\frac{\beta q_i k}{i^2} + \lambda\frac{k}{i}\mu^{i-k}\right). \tag{42}$$

*If $e_{t,k} = K_{t,k} - \Gamma$, then*

$$|e_{t,k}| \le \frac{C_H}{k} + \Gamma\mu^{t-k}, \qquad \sum_{k=1}^{t-1} e_{t,k}^2 \le C_E^{HyM}. \tag{43}$$

*The coefficients and bounds are deterministic.*

*Proof.* Unrolling the two momentum states gives

$$\alpha_t \mathbf{m}_t = \frac{q_t}{t^2}\sum_{k=1}^{t} k\mathbf{g}_k, \qquad \mathbf{b}_t = \sum_{k=1}^{t}\frac{k}{t}\mu^{t-k}\mathbf{g}_k. \tag{44}$$

Let $\boldsymbol{\delta}_t = \mathbf{y}_t - \mathbf{x}_t$. The iterate-averaging recurrence implies $\boldsymbol{\delta}_{t+1} = (1-a_{t+1})(\boldsymbol{\delta}_t - \eta\boldsymbol{\Delta}_t)$. Since $\boldsymbol{\delta}_1 = 0$,

$$\mathbf{v}_t = \mathbf{y}_t + \frac{\beta}{\lambda}\boldsymbol{\delta}_t = \mathbf{y}_1 - \eta\sum_{i=1}^{t-1}\omega_{i,t}\boldsymbol{\Delta}_i.$$

Substitution of (44) proves (42).

The constant reference weight $\Gamma$ admits the identity

$$\Gamma\beta\sum_{i=k}^{t-1}\omega_{i,t}\frac{k}{i(i+1)} + \Gamma\lambda\omega_{k,t} = \Gamma. \tag{45}$$

For $\beta > 0$, this follows from

$$\sum_{i=k}^{t-1}\frac{k\Pi_{i+1,t}}{i(i+1)} = \frac{\lambda}{\beta}\left(\frac{k}{t} - \Pi_{k+1,t}\right) \tag{46}$$

and $\sum_{i=k}^{t-1} k/(i(i+1)) = 1 - k/t$. To verify (46), let $r_i = \Pi_{i+1,t}/i$. From $\Pi_{i+2,t} = \Pi_{i+1,t}(i+\lambda^{-1})/i$, one obtains $r_i - r_{i+1} = (1-\lambda^{-1})\Pi_{i+1,t}/(i(i+1))$, and the result follows by telescoping. For $\beta = 0$, (45) is immediate.

Subtracting (45) from (42) expresses $e_{t,k}$ as the sum of a slow-momentum difference and a fast-momentum difference. Since $1 \le \omega_{i,t} \le 1/\lambda$ and $q_i = \Gamma(1-\mu^i)$, the slow difference satisfies

$$\begin{aligned}\left|\beta \sum_{i=k}^{t-1} \omega_{i,t} k \left(\frac{q_i}{i^2} - \frac{\Gamma}{i(i+1)}\right)\right| &\le \frac{\beta\Gamma k}{\lambda} \sum_{i=k}^{t-1} \left(\frac{1}{i^2(i+1)} + \frac{\mu^i}{i^2}\right) \\ &\le \frac{\beta\Gamma}{\lambda k}(1+\Gamma\mu^k) \le \frac{\beta\Gamma(1+\Gamma)}{\lambda k}.\end{aligned}$$

Here we used $\sum_{i=k}^{\infty}[i^2(i+1)]^{-1} \le k^{-1}\sum_{i=k}^{\infty}[i(i+1)]^{-1} = k^{-2}$ and $\sum_{i=k}^{\infty}\mu^i/i^2 \le \Gamma\mu^k/k^2$.

For the fast difference, the product recurrence gives $\omega_{j+1,t} - \omega_{j,t} = (\beta/\lambda)a_{j+1}\Pi_{j+2,t}$. Using $a_{j+1} \le 1/(\lambda j)$ and $\Pi_{j+2,t} \le 1$, for $k \le i < t$ we obtain

$$0 \le \omega_{i,t} - \omega_{k,t} = \frac{\beta}{\lambda}\sum_{j=k}^{i-1} a_{j+1}\Pi_{j+2,t} \le \frac{\beta(i-k)}{\lambda^2 k}.$$

Consequently, since $\beta + \lambda = 1$,

$$\begin{aligned}\left|\omega_{i,t}\frac{k}{i} - \omega_{k,t}\right| &\le \frac{k}{i}|\omega_{i,t} - \omega_{k,t}| + \omega_{k,t}\left(1 - \frac{k}{i}\right) \\ &\le \frac{\beta(i-k)}{\lambda^2 k} + \frac{i-k}{\lambda k} = \frac{i-k}{\lambda^2 k}.\end{aligned}$$

Separating the finite geometric sum from its tail therefore gives

$$\begin{aligned}\left|\lambda\sum_{i=k}^{t-1}\omega_{i,t}\frac{k}{i}\mu^{i-k} - \lambda\Gamma\omega_{k,t}\right| &\le \frac{1}{\lambda k}\sum_{j=0}^{t-k-1} j\mu^j + \lambda\omega_{k,t}\sum_{j=t-k}^{\infty}\mu^j \\ &\le \frac{\mu\Gamma^2}{\lambda k} + \Gamma\mu^{t-k},\end{aligned}$$

where $\sum_{j=0}^{\infty} j\mu^j = \mu\Gamma^2$ and $\lambda\omega_{k,t} \le 1$. Adding the two differences proves $|e_{t,k}| \le C_H/k + \Gamma\mu^{t-k}$. Finally, using $(a+b)^2 \le 2a^2 + 2b^2$,

$$\begin{aligned}\sum_{k=1}^{t-1} e_{t,k}^2 &\le 2C_H^2\sum_{k=1}^{t-1}\frac{1}{k^2} + 2\Gamma^2\sum_{j=1}^{t-1}\mu^{2j} \\ &\le \frac{\pi^2 C_H^2}{3} + \frac{2\Gamma^2\mu^2}{1-\mu^2} = C_E^{\texttt{HyM}}.\end{aligned}$$

This also holds at $t = 1$, when the sum is empty, and completes the uniform bound independently of the training horizon. □

### C.4 Convergence of the averaged iterate for HyM

*Proof of Corollary 1.* Equations (31) and (41) give (9) with $\mathbf{v}_1 = \mathbf{y}_1$, $\zeta_t = \beta$, and $\Gamma = (1-\mu)^{-1}$. Lemma C.2 gives

$$|e_{t,k}| \le \frac{C_H}{k} + \Gamma\mu^{t-k}, \qquad 1 \le k < t, \tag{47}$$

where $C_H = [\beta\Gamma(1+\Gamma) + \mu\Gamma^2]/(1-\beta)$. Using $(a+b)^2 \le 2a^2 + 2b^2$ and summing over $k$ yields

$$\sum_{k=1}^{t-1} e_{t,k}^2 \le \frac{\pi^2 C_H^2}{3} + \frac{2\Gamma^2\mu^2}{1-\mu^2} = C_E^{\texttt{HyM}} < \infty. \tag{48}$$

Thus the square-sum condition holds uniformly in $t$. Since $C_E^{\mathrm{HyM}}$ depends only on $\mu$ and $\beta$, we may choose any $0 < \eta_0 \le [2L(\Gamma + 2\sqrt{C_E^{\mathrm{HyM}}})]^{-1}$ independently of $T$. Theorem 1 therefore yields

$$\mathbb{E}[f(\mathbf{x}_T) - f(\mathbf{x}^*)] \le \frac{\|\mathbf{y}_1 - \mathbf{x}^*\|^2/(2\eta_0\Gamma) + \eta_0\Gamma\varsigma^2/2}{\sqrt{T}} + \frac{2L\eta_0^2 C_E^{\mathrm{HyM}}\varsigma^2}{T}. \tag{49}$$

The right-hand side is $\mathcal{O}(T^{-1/2})$, which proves the claim. □

All constants are independent of $T$ for fixed $\mu, \beta$; they can deteriorate as either parameter approaches one. The proof uses ordinary convexity and the stochastic oracle, without strong convexity, bounded iterates, or uniformly bounded gradients.

### C.5 Finite-EMA recovery and relation to other analyses

As an illustrative specialization, consider zero-initialized normalized EMA states $\mathbf{m}_t^{(r)} = \rho_r \mathbf{m}_{t-1}^{(r)} + (1-\rho_r)\mathbf{g}_t$ for $r = 1, \dots, R$, where $0 \le \rho_r < 1$, and $\mathbf{v}_{t+1} = \mathbf{v}_t - \eta \sum_{r=1}^R a_r \mathbf{m}_t^{(r)}$ with $a_r \ge 0$ and $\Gamma = \sum_{r=1}^R a_r > 0$. For passive iterate averaging, take $\zeta_t = 0$ and $\mathbf{x}_t = t^{-1}\sum_{i=1}^t \mathbf{v}_i$. Unrolling gives

$$e_{t,k} = -\sum_{r=1}^R a_r \rho_r^{t-k}, \qquad C_E^{\mathrm{EMA}} = \sum_{r=1}^R \sum_{j=1}^R \frac{a_r a_j \rho_r \rho_j}{1 - \rho_r \rho_j}, \qquad \chi_T \le \sum_{r=1}^R \frac{a_r \rho_r}{1-\rho_r}.$$

The bound on the row square sums follows by summing the geometric cross terms; the operator bound follows from the sum of the absolute convolution weights. Theorem 1 therefore recovers an averaged-iterate guarantee for these linear first-moment mixtures. This includes the fixed dual-EMA first-moment core associated with AdEMAMix, with $a_1 = 1$, $a_2 = \alpha$; the full adaptive AdEMAMix denominator and schedules are not part of this example. Aggregated Heavy-Ball analyzes multiple momentum states (Danilova, 2022). Generalized SGDM analyzes a single momentum state combined with the current gradient (Wang & Yurtsever, 2026).

The analysis builds on online-to-batch geometry (Defazio et al., 2024) and reference-iterate perturbation arguments used for error feedback, delay and compression (Karimireddy et al., 2019; Stich & Karimireddy, 2020). Memory-augmented optimizer theory also treats stored gradient histories (McRae et al., 2022). Here the uniform bound is imposed directly on the deterministic cumulative weight deviations $e_{t,k} = K_{t,k} - \Gamma$ in (9). The analysis uses the Bregman term to absorb the contribution of these weight deviations and bound the function-value gap at the final averaged iterate. `HyM` supplies a growing-memory instance with the error bound (47).

## D HyMuon matrix updates and implementation

**Algorithm 2** `HyMuon` matrix updates

**Input:** $\mathbf{Y}_1 \in \mathbb{R}^{m\times n}$, $\eta > 0$, $\mu \in (0,1)$, $\beta \in [0,1)$, $B, T \in \mathbb{N}$; initialize $\mathbf{M}_0 = \mathbf{B}_0 = \mathbf{0}$, $\mathbf{X}_0 = \mathbf{Y}_1$

1: **for** $t = 1, \dots, T$ **do**
2: Sample $\mathcal{B}_t = (\xi_{t,j})_{j=1}^B$ and set $\mathbf{G}_t = B^{-1}\sum_{j=1}^B \nabla f(\mathbf{Y}_t; \xi_{t,j})$
3: $\mathbf{M}_t = (1 - 1/t)\mathbf{M}_{t-1} + \mathbf{G}_t$ ▷ Slow momentum
4: $\mathbf{B}_t = \mu(1 - 1/t)\mathbf{B}_{t-1} + \mathbf{G}_t$ ▷ Fast momentum
5: $\mathbf{D}_t = \frac{\beta(1-\mu^t)}{t(1-\mu)}\mathbf{M}_t + (1-\beta)\mathbf{B}_t$
6: $a_t = \frac{1}{1+(1-\beta)(t-1)}$, $\mathbf{X}_t = (1-a_t)\mathbf{X}_{t-1} + a_t\mathbf{Y}_t$
7: $\mathbf{Y}_{t+1} = \mathbf{Y}_t - \eta\,\mathrm{NS}(\mathbf{D}_t)$
8: **end for**

**Output:** $\mathbf{X}_T$

Here $2\mathbf{M}_t/(t+1)$ equals the matrix-valued slow momentum in Algorithm 1; $\mathrm{NS}(\cdot)$ denotes the output of the finite Newton–Schulz iteration used to approximate the polar factor.

We extend the matrix updates above to the full-model optimizer used in the HyMuon experiments. Its combined momentum is the matrix-valued recurrence in Algorithm 2. The practical full-model optimizer uses five NS iterations and additionally includes shape scaling, decoupled weight decay, experiment-specific learning-rate schedules, and the adaptive recurrences below. For a hidden parameter $\mathbf{Y}_t \in \mathbb{R}^{m\times n}$ with gradient $\mathbf{G}_t$, the implementation initializes $\mathbf{M}_0 = \mathbf{B}_0 = \mathbf{0}$ and forms

$$\mathbf{M}_t = (1 - c_t)\mathbf{M}_{t-1} + \mathbf{G}_t, \tag{50}$$

$$\mathbf{B}_t = \mu(1 - c_t)\mathbf{B}_{t-1} + \mathbf{G}_t, \tag{51}$$

$$\mathbf{D}_t = \beta c_t q_t(\mu)\mathbf{M}_t + (1 - \beta)\mathbf{B}_t, \qquad q_t(\rho) = \frac{1 - \rho^t}{1 - \rho}, \tag{52}$$

$$\mathbf{U}_t = s_{m,n}\,\mathrm{NS}(\mathbf{D}_t), \qquad s_{m,n} = \sqrt{\max\{1, m/n\}}. \tag{53}$$

Here $\mathrm{NS}(\cdot)$ denotes the output of five quintic NS iterations; the shape factor $s_{m,n}$ is applied afterwards. Both $\mathbf{G}_t$ and the auxiliary gradient $\mathbf{g}_t$ are mini-batch gradients at the current training iterates, as in Algorithm 2. For an auxiliary parameter tensor, let $\mathbf{h}_t^{(1)} = \mathbf{g}_t$ and $\mathbf{h}_t^{(2)} = \mathbf{g}_t \odot \mathbf{g}_t$. The coefficients $\rho_1$ and $\rho_2$, corresponding to Adam's $\beta_1$ and $\beta_2$, control the fast first- and second-moment recurrences. With all four unnormalized auxiliary moment states initialized to zero, the adaptive HyM branch computes, for $r \in \{1, 2\}$,

$$\mathbf{m}_t^{(r)} = (1 - c_t)\mathbf{m}_{t-1}^{(r)} + \mathbf{h}_t^{(r)}, \tag{54}$$

$$\mathbf{b}_t^{(r)} = \rho_r(1 - c_t)\mathbf{b}_{t-1}^{(r)} + \mathbf{h}_t^{(r)}, \tag{55}$$

$$\mathbf{d}_t^{(r)} = \beta c_t q_t(\rho_r)\mathbf{m}_t^{(r)} + (1 - \beta)\mathbf{b}_t^{(r)}, \tag{56}$$

$$\mathbf{u}_t = \mathbf{d}_t^{(1)} \oslash \left(\sqrt{\mathbf{d}_t^{(2)}} + \varepsilon\right). \tag{57}$$

Here $\odot$ and $\oslash$ denote elementwise multiplication and division. Write $\mathbf{y}_t$ for the auxiliary training iterate and $\mathbf{x}_t$ for the corresponding averaged iterate. With $c_t = 1/t$, initialize $\mathbf{X}_0 = \mathbf{Y}_1$ and $\mathbf{x}_0 = \mathbf{y}_1$. Before each training step, average both parameter groups using $a_t = 1/[1+(1-\beta)(t-1)]$:

$$\mathbf{X}_t = (1 - a_t)\mathbf{X}_{t-1} + a_t\mathbf{Y}_t, \qquad \mathbf{x}_t = (1 - a_t)\mathbf{x}_{t-1} + a_t\mathbf{y}_t. \tag{58}$$

If an implementation exhibits anomalous convergence at the averaged iterate, the numerical precision used to store and update these averages should be checked, since $a_t$ becomes small late in training. Here $\rho_1$ and $\rho_2$ determine the horizons of the auxiliary first- and second-moment estimates, while $\beta$ mixes the slow and fast estimates in both recurrences and determines the iterate-averaging coefficient $a_t$. For a hidden matrix and an auxiliary tensor, respectively, the subsequent decoupled-weight-decay updates are

$$\mathbf{Y}_{t+1} = (1 - \eta_t^{\mathrm{mat}}\lambda_{\mathrm{mat}})\mathbf{Y}_t - \eta_t^{\mathrm{mat}}\mathbf{U}_t, \tag{59}$$

$$\mathbf{y}_{t+1} = (1 - \eta_t^{\mathrm{aux}}\lambda_{\mathrm{aux}})\mathbf{y}_t - \eta_t^{\mathrm{aux}}\mathbf{u}_t. \tag{60}$$

The optimizer's evaluation mode swaps all parameter groups to their respective $\mathbf{X}_t$ and $\mathbf{x}_t$ values and restores the training iterates before the next step. GPT-2 uses LayerNorm and contains no Batch Normalization, so the running-statistic refresh hook is a no-op.

# E Diagnostic experiment details

## E.1 Batch-size dependence of the HyM advantage

If temporal accumulation is most useful when an individual gradient contains less information, HyM's advantage should be larger in small-batch training. We examine this on CIFAR-10 with WideResNet-16-8 and CIFAR-100 with DenseNet-BC, training each model for 300 epochs on its full training set. We use $B \in \{8, 16, 32, 128, 256, 1024, 2048\}$ on CIFAR-10 and $B \in \{64, 128, 256, 512, 1024\}$ on CIFAR-100. Within each dataset, the model, augmentation, and non-LR optimizer settings are fixed across batch sizes as in Appendix F.1; both methods use a 5% warmup followed by cosine decay, ending at half the peak learning rate for HyM and zero for SGD.

Peak learning rates use method- and batch-specific candidates obtained by linear batch-size scaling. Figure 8 reports test loss for the selected configuration of each method and batch size.

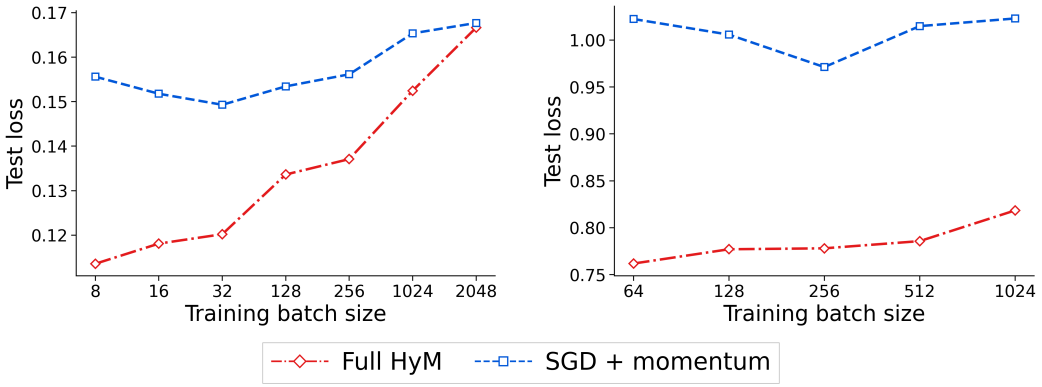


Figure 8: Test loss across training batch sizes on CIFAR-10 (left) and CIFAR-100 (right). Learning rates are chosen separately for each method and batch size.

`HyM` has lower observed test loss at every plotted batch size. On CIFAR-10, the gap narrows from about $0.042$ at $B = 8$ to $0.001$ at $B = 2048$. On CIFAR-100, the gap is about $0.261$ at $B = 64$ and $0.205$ at $B = 1024$, but is not monotone across the intervening batches. The CIFAR-10 trend is consistent with temporal accumulation being more useful when each step is based on fewer samples; the CIFAR-100 results show a broader advantage without the same monotone pattern.

### E.2 Semantic relevance and target-specific improvement

**Checkpoints and data.** Section 4.3 uses GPT-2 124M checkpoints at 10k, 50k, and 100k steps of an AdamW run initialized from public GPT-2 weights and continued on OpenWebText. Each step processes 491,520 tokens. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay $0.1$, a 2k-step warmup to $6 \times 10^{-4}$, and cosine decay to $6 \times 10^{-5}$ at 100k. The experiment loads model parameters only. Its 2,000 candidates are uniformly sampled 256-token training fragments contained within document boundaries and are screened for near-duplicates with the target texts.

**Targets and similarity.** A frozen `intfloat/e5-small-v2` encoder (Wang et al., 2022) embeds 16,384 validation fragments, with a `passage:` prefix and masked mean pooling. We subtract the embedding mean and its first three principal components estimated on 4,096 background training fragments, then normalize each embedding. MiniBatchKMeans forms 128 clusters. We select ten compact clusters, each providing 64 fragments from distinct documents, and alternate them into 32 reference and 32 evaluation fragments. Ten controls each mix 64 randomly chosen fragments from other documents and use the same split. All fragments contain 256 tokens. These are encoder-defined semantic neighborhoods, not annotated grammatical or factual categories. Similarity reads only reference text; the outcome reads only evaluation text.

For a candidate's single embedding, its E5 score is the mean of its maximum and mean cosine similarity to the 32 reference embeddings. We also regress this score on IDF-weighted token cosine and background-centered bigram cosine within each target; the residual measures the part not linearly explained by these lexical scores. Gradient cosine compares the candidate gradient with the reference-set gradient at the same checkpoint.

**Equal-length directional updates.** For each candidate $C$, restore the checkpoint $\mathbf{y}$ and evaluate

$$\mathbf{y}_C = \mathbf{y} - \delta \frac{\nabla \mathcal{L}_C(\mathbf{y})}{\|\nabla \mathcal{L}_C(\mathbf{y})\|}, \qquad \delta = 3 \times 10^{-4}.$$

There is no momentum, weight decay, iterate averaging, or carryover between candidates. Dropout is zero. For target $V$, the raw improvement is $I(C, V) = \mathcal{L}_{V_{\text{eval}}}(\mathbf{y}) - \mathcal{L}_{V_{\text{eval}}}(\mathbf{y}_C)$. Target-specific improvement subtracts from $I(C, V)$ the mean of $I(C, V')$ over the other 19 targets. The analysis contains 120,000 measurements over candidates, targets, and checkpoints.

**Aggregation.** For each target and checkpoint, rank all candidates by similarity and form 100 equal bins of 20 candidates. Section 4.3 reports the mean within-target Spearman correlation with target-specific improvement. For the net-share statistic, sum raw improvements over each narrow target's top 400 candidates, then divide by the sum over all 2,000 candidates and all ten narrow targets. Negative effects remain in both sums, so cancellation can increase this ratio. A second statistic clips negative *target-specific* improvements to zero and averages within-target shares; under it, the same top 20% accounts for $31.9\%$, $33.4\%$, and $33.3\%$ at 10k, 50k, and 100k. Both statistics indicate enrichment but quantify different outcomes. These measurements provide intuition for how semantic relevance can make useful gradient contributions uneven across batches.

**Additional statistics.** For the randomly mixed controls, mean raw improvements over 2,000 candidates and ten controls are $9.272 \times 10^{-7}$, $2.250 \times 10^{-6}$, and $1.398 \times 10^{-6}$ at 10k, 50k, and 100k, respectively. For narrow targets, the association with E5 similarity remains positive after removing the part linearly explained by token and bigram overlap. Gradient alignment has larger mean correlations with target-specific improvement: $0.410$, $0.354$, and $0.399$. At 100k, $52.67\%$ of narrow-target candidate pairs have positive raw improvement. The $92.15\%$ top-20% net share includes cancellation between positive and negative effects; it does not mean that the other $80\%$ of candidates are uniformly unhelpful.

**Percentile curves and contribution shares.** Figures 9 and 10 give the complete plots accompanying Section 4.3. Both figures use raw held-out improvement $I(C, V)$, whereas the reported correlations use target-specific improvement. Both figures rank candidates by E5 similarity.

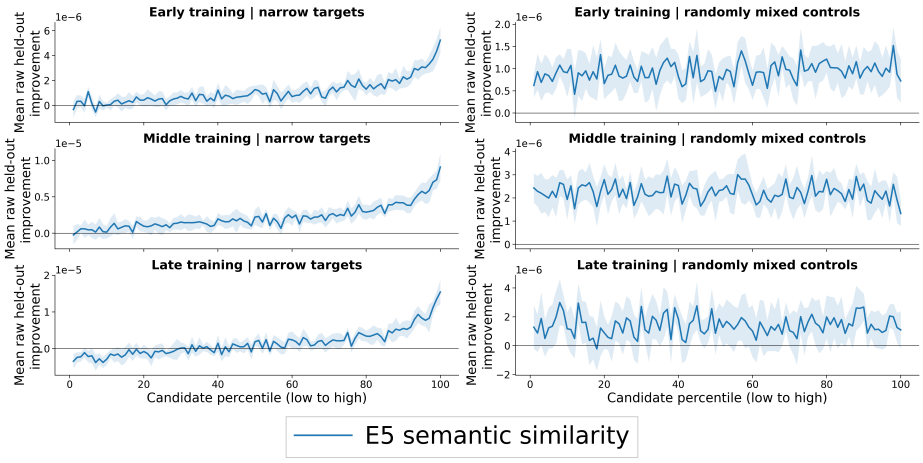


Figure 9: Complete raw held-out improvement curves across 100 E5-similarity percentiles. Rows correspond to 10k, 50k, and 100k checkpoints; columns show narrow targets (left) and randomly mixed controls (right). Rankings are formed separately within each target, with 20 candidates per percentile. Curves average over ten targets of each type; bands show $\pm 1.96$ standard errors across targets. The blue curve uses E5 similarity. Positive values indicate loss reduction; vertical scales differ. Figure 3 displays only the left-column panels, rearranged in one row.

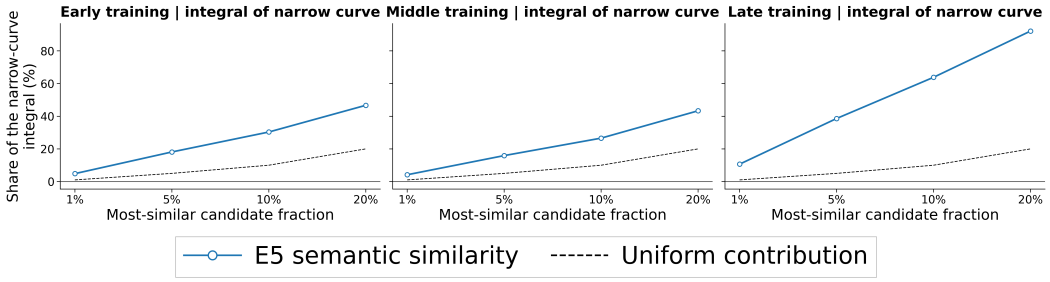


Figure 10: Share of narrow-target net improvement contributed by the most E5-similar $1\%, 5\%, 10\%$, and $20\%$ of candidates. Panels correspond to 10k, 50k, and 100k checkpoints from left to right. The blue curve uses the E5 ranking from Figure 9. The numerator sums raw improvements over the selected candidates and all ten narrow targets; the denominator sums over all candidates and the same targets. The dashed line indicates contribution proportional to subset size. Negative effects are retained, so cancellation can produce shares above $100\%$; these are net shares, not fractions of positive improvements.

### E.3 Single-batch forgetting diagnostic

The diagnostic in Figure 2 (left) uses the same GPT-2 124M, OpenWebText, context length 1024, and global optimizer batch of 480 sequences described above, but runs for 10,000 steps. We deterministically select one global batch $\mathcal{B}$. It contains 491,520 next-token targets. Ordinary training sampling excludes every sequence window that overlaps any token in $\mathcal{B}$, for both control and inject runs. The inject run replaces its ordinary global batch with the full $\mathcal{B}$ only at step 5000; the control never uses $\mathcal{B}$. Apart from this replacement, stateless sampling keyed by the run configuration, step, rank, and micro-step gives paired control and inject runs the same ordinary batch sequence. The control and inject branches start from the same complete checkpoint immediately before step 5000.

Training uses gradient clipping at 1.0, 200 warmup steps, and cosine decay through step 10,000. Loss on $\mathcal{B}$ is evaluated every 50 steps and additionally at steps 4999, 5000, and 5001; distributed ranks evaluate disjoint shards whose losses are averaged. `HyMuon` switches all parameter groups to the averaged iterate $\mathbf{x}_t$ for these measurements and then returns to its training iterate. Muon has no such iterate switch. Because the branches share the complete pre-injection checkpoint and subsequent sampling state, their only protocol difference is the batch used at step 5000.

For Muon, the hidden-matrix branch uses peak/minimum learning rates $2 \times 10^{-2}$ and $3 \times 10^{-5}$, momentum $0.95$, and zero weight decay. Its auxiliary AdamW branch uses peak/minimum learning rates $3 \times 10^{-3}$ and $3 \times 10^{-5}$, betas $(0.9, 0.95)$, and weight decay $0.1$; inverse-layer learning-rate scaling is enabled. For `HyMuon`, the hidden-matrix branch uses peak/minimum learning rates $0.1$ and $3 \times 10^{-5}$, Muon momentum $0.9$, `HyM` mixing weight $0.95$, and weight decay $0.05$. Its adaptive `HyM` branch uses peak/minimum learning rates $0.01$ and $3 \times 10^{-5}$, first- and second-moment coefficients $0.95$ and $0.99$, and weight decay $0.01$. These settings define the two optimizer-specific loss trajectories in the left panel. The state intervention below then tests the stored contribution directly while holding the remaining model and optimizer state fixed.

#### E.3.1 Slow-momentum intervention

Figure 2 (right) continues the injection protocol with a targeted intervention at step 7500. Because the injected batch replaces an ordinary batch at step $s = 5000$, the tracked signed gradient difference is $\Delta \mathbf{g}_s = \mathbf{g}_{\mathcal{B}}(s) - \mathbf{g}_{\text{ordinary}}(s)$, rather than the injected-batch gradient alone. In `HyMuon`'s slow momentum, a gradient from step $s$ has coefficient $s/t$ at step $t$; consequently, the coefficient of $\Delta \mathbf{g}_s$ is $2/3$ at the intervention. The Delete branch subtracts this exact contribution from the slow momentum states of the matrix and auxiliary parameter groups. The Refresh branch first performs the same deletion and then computes $\mathbf{g}_{\mathcal{B}}(7500)$ at the current training iterate, scaling its matrix and auxiliary components separately so that each has the same Euclidean norm as the corresponding removed component. At the same lag, the corresponding matrix-momentum and auxiliary first-moment coefficients for Muon are approximately $2.0 \times 10^{-56}$ and $4.0 \times 10^{-116}$, respectively. These

coefficients are numerically negligible at this lag, so the displayed intervention focuses on HyMuon's persistent slow momentum. Muon's 10k Keep, Delete, and Refresh target losses differ by less than $0.0024$ and are indistinguishable at the scale of the plot.

The operation does not modify model parameters, the averaged iterate $\mathbf{x}_t$, the fast momentum state, or either second-moment state. Immediately before the next ordinary step, the code applies the complete NS or adaptive transformation to the original and intervened states. The resulting learning-rate-scaled parameter-update differences have norms $0.104$ for Delete and $0.111$ for Refresh. At step 10,000, the held-out-batch losses are $3.047617$ for Delete, $3.038296$ for Keep, and $3.025494$ for Refresh.

**Common-checkpoint protocol.** Keep, Delete, and Refresh share the same pre-intervention training history: they were initialized from the same complete checkpoint immediately before the intervention at step 7500. Each branch received an identical copy of the model parameters and buffers, all optimizer states (including the averaged iterate $\mathbf{x}_t$), the learning-rate scheduler and training counters, and all random-number-generator and data-sampling states on every rank. The three interventions were then applied to these copies, followed by the same ordinary-batch sequence. The gradient computation required by Refresh preserved the random and model-buffer states used for subsequent training. The immediate update comparison holds parameters fixed and changes only the specified optimizer state.

# F BENCHMARK EXPERIMENT DETAILS

This section reports the architectures, training configurations, search dimensions, and selected hyperparameters used in the reported experiments. For each task, we report the tuned hyperparameter families and selected settings. The FineWeb baselines follow the published benchmark, while the C4 comparison uses the Muon baseline reproduced with the training configuration of Gu & Xie (2026), as described below.

**Results summary.** Table 1 collects the numerical results for Section 6. Image-classification and IWSLT entries report the mean and sample standard deviation across repeated runs; OpenWebText, FineWeb, and C4 entries give validation loss at the final training step.

Table 1: Main experimental results. Image accuracy and IWSLT metrics are reported as mean ± sample SD across repeated runs. OpenWebText, FineWeb, and C4 report final validation loss. ImageNet accuracy is validation top-1; CIFAR and SVHN use test accuracy. Dashes indicate methods not included in that comparison. FineWeb losses are rounded to two decimals.

| **Image classification: accuracy (%) ↑** | | | | |
|---|---|---|---|---|
| Dataset / model / budget | SGD Cosine | SGD Step | Schedule-Free | HyM |
| CIFAR-10 / WRN-16-8 300 epochs | $95.90 \pm 0.12$ | $95.43 \pm 0.17$ | $96.11 \pm 0.10$ | $96.25 \pm 0.10$ |
| CIFAR-100 / DenseNet-BC 300 epochs | $77.66 \pm 0.39$ | $77.22 \pm 0.20$ | $79.11 \pm 0.22$ | $79.14 \pm 0.22$ |
| SVHN / ResNet-3-96 300 epochs | $98.24 \pm 0.05$ | – | $98.20 \pm 0.08$ | $98.47 \pm 0.02$ |
| ImageNet-1K / ResNet-50 100 epochs | $76.68 \pm 0.04$ | – | $76.70 \pm 0.06$ | $76.84 \pm 0.10$ |

| **Language tasks** | | | | | |
|---|---|---|---|---|---|
| Dataset / model / budget | Metric | AdamW | AdEMAMix | Muon | HyMuon |
| IWSLT14 / LSTM 55 epochs | Test loss ↓ | – | – | $4.234 \pm 0.006$ | $4.184 \pm 0.004$ |
| | BLEU ↑ | – | – | $33.424 \pm 0.048$ | $33.874 \pm 0.072$ |
| OpenWebText / GPT-2 124M 100k steps | Val. loss ↓ | 2.905 | – | 2.915 | 2.886 |
| OpenWebText / GPT-2 124M 600k steps | Val. loss ↓ | 2.866 | – | 2.886 | 2.823 |
| FineWeb / LLaMA-124M 8k steps (≈1B tokens) | Val. loss ↓ | 3.53 | 3.48 | 3.50 | 3.47 |
| FineWeb / LLaMA-124M 16k steps (≈2.1B tokens) | Val. loss ↓ | 3.39 | 3.35 | 3.39 | 3.35 |
| C4 / LLaMA-1.3B 10k steps | | | | | |
| Mano-recipe Muon | Val. loss ↓ | – | – | 2.903 | 2.777 |

### F.1 Image-classification runs

**Models and training budgets.** Table 2 gives the common settings for each dataset. CIFAR-10 uses a WideResNet of depth 16 and width multiplier 8, without dropout. CIFAR-100 uses DenseNet-BC with block sizes $(6, 12, 24, 16)$, growth rate 12, 24 initial channels, bottleneck multiplier 4, compression factor $0.5$, and no dropout. The SVHN ResNet has three stages with $(96, 192, 384)$ channels and three two-convolution residual blocks per stage. ImageNet uses ResNet-50 initialized from scratch. All four tasks use cross-entropy loss without label smoothing, mixup, or CutMix; evaluation is performed after every epoch.

Table 2: Image-classification training settings. Batch sizes are global numbers of images per optimizer step.

| Dataset | Epochs | Batch size | Input size |
|---|---|---|---|
| CIFAR-10 | 300 | 128 | $32 \times 32$ |
| CIFAR-100 | 300 | 64 | $32 \times 32$ |
| SVHN (train + extra) | 300 | 32 | $32 \times 32$ |
| ImageNet-1K | 100 | 256 | $224 \times 224$ |

**Preprocessing.** CIFAR and SVHN training images use four-pixel reflection padding, random $32 \times 32$ crops, and horizontal flips with probability $0.5$; test images are not augmented. SVHN training includes both the official training and extra splits. After scaling pixels to $[0, 1]$, CIFAR-10 and SVHN use channel-wise mean and standard deviation $(0.5, 0.5, 0.5)$. CIFAR-100 uses mean $(0.5074, 0.4867, 0.4411)$ and standard deviation $(0.2011, 0.1987, 0.2025)$. ImageNet training uses random resized crops to $224 \times 224$ and random horizontal flips; evaluation resizes the shorter edge to 256 and takes a $224 \times 224$ center crop. Its channel-wise mean and standard deviation are also $(0.5, 0.5, 0.5)$.

**Optimizers and schedules.** Table 3 lists the settings held fixed across repeated runs. SGD uses ordinary momentum with zero dampening and no Nesterov term; `HyM` also uses no additional Nesterov term. Both apply coupled weight decay to all parameters. Cosine schedules have linear warmup over the first $5\%$ of optimizer steps. The cosine endpoint is zero for SGD on all four datasets and for `HyM` on SVHN. For `HyM` on CIFAR-10, CIFAR-100, and ImageNet, the endpoint is instead half the peak learning rate: $0.025$, $0.25$, and $0.003$, respectively. The Step baseline has no warmup and multiplies its learning rate by $0.1$ at zero-indexed epochs 150 and 225. Schedule-Free uses a $5\%$ linear warmup followed by a constant learning rate, using the `SGDScheduleFree` implementation.

Table 3: Fixed optimizer settings for image classification. LR denotes the peak learning rate. The momentum column gives the SGD or `HyM` momentum coefficient; $\beta$ gives the Schedule-Free interpolation coefficient or the `HyM` mixing weight.

| Dataset | Optimizer | LR | Momentum | $\beta$ | Weight decay |
|---|---|---|---|---|---|
| CIFAR-10 | SGD + Cosine | 0.2 | 0.9 | – | $10^{-4}$ |
| | SGD + Step | 0.1 | 0.9 | – | $10^{-4}$ |
| | Schedule-Free | 10 | – | 0.9 | $10^{-4}$ |
| | `HyM` | 0.05 | 0.98 | 0.9 | $5 \times 10^{-4}$ |
| CIFAR-100 | SGD + Cosine | 0.05 | 0.9 | – | $2 \times 10^{-4}$ |
| | SGD + Step | 0.05 | 0.9 | – | $2 \times 10^{-4}$ |
| | Schedule-Free | 5 | – | 0.9 | $2 \times 10^{-4}$ |
| | `HyM` | 0.5 | 0.9 | 0.9 | $3 \times 10^{-4}$ |
| SVHN | SGD + Cosine | 0.1 | 0.9 | – | $10^{-4}$ |
| | Schedule-Free | 1 | – | 0.9 | $2 \times 10^{-4}$ |
| | `HyM` | 0.05 | 0.99 | 0.95 | $3 \times 10^{-4}$ |
| ImageNet-1K | SGD + Cosine | 0.05 | 0.9 | – | $10^{-4}$ |
| | Schedule-Free | 1.5 | – | 0.9 | $5 \times 10^{-5}$ |
| | `HyM` | 0.006 | 0.99 | 0.90 | $2 \times 10^{-4}$ |

**Evaluation and repeated runs.** All displayed methods on CIFAR-10, CIFAR-100, SVHN, and ImageNet are evaluated over repeated runs. Curves show means across runs, with standard-error bands and sample standard deviations in the legends. ImageNet is evaluated on the official validation split.

`HyM` and Schedule-Free refresh Batch-Normalization statistics using 50 training mini-batches before evaluation. All ImageNet runs use a global batch of 256.

**Training time and memory.** We profiled `HyM` and cosine SGD in separate short runs on an NVIDIA H100 80GB GPU. Within each row of Table 4, the model, profiling batch size, data pipeline, worker count, and optimizer steps per epoch match. The table reports mean training-loop wall-clock time over epochs 2–5 and the maximum PyTorch CUDA allocated memory over those epochs.

Table 4: H100 resource profiles for `HyM` versus cosine SGD. Batch sizes in parentheses are the profiling batch sizes. Time is seconds per training epoch; memory is peak CUDA allocated MiB. Changes are `HyM` relative to cosine SGD.

| | Training time | | | Peak allocated memory | | |
|---|---|---|---|---|---|---|
| Dataset (batch) | SGD | `HyM` | Change | SGD | `HyM` | Change |
| CIFAR-10 (128) | 4.71 | 4.98 | $+5.7\%$ | 1122 | 1254 | $+132$ |
| CIFAR-100 (512) | 6.80 | 6.73 | $-0.9\%$ | 7410 | 7423 | $+12$ |
| ImageNet-1K (256) | 591.62 | 601.43 | $+1.7\%$ | 11257 | 11561 | $+304$ |

Across the three configurations, `HyM`'s measured training time differs from cosine SGD by $-0.9\%$ to $+5.7\%$, while peak allocated memory increases by 12–304 MiB; in these profiles the time and memory overhead is modest.

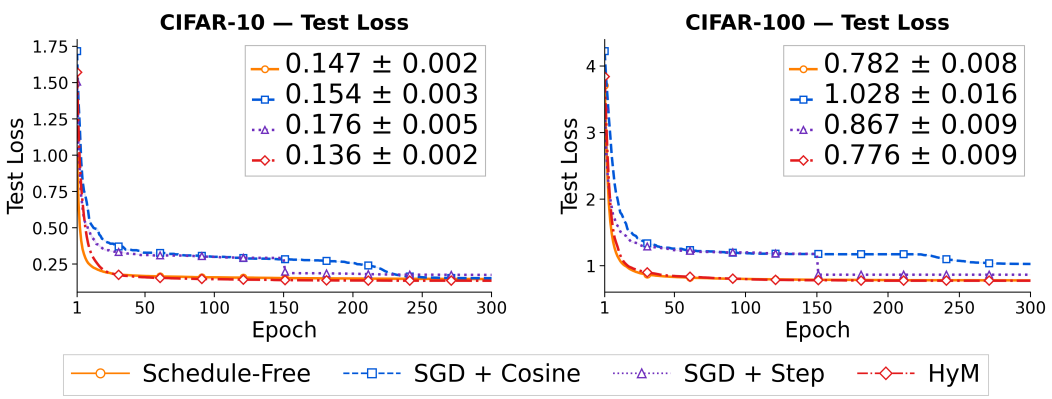


Figure 11: CIFAR test-loss curves accompanying Figure 4. Left: CIFAR-10; right: CIFAR-100. Curves show run means with one-standard-error bands, and legends report sample standard deviations. HyM is evaluated at the averaged iterate $\mathbf{x}_t$.

### F.2 HyM at the training and averaged iterates

Figure 12 compares a HyM configuration evaluated at the training iterate $y$ with a HyM configuration evaluated at the averaged iterate $x$ on CIFAR-10 and CIFAR-100. Both configurations train for 300 epochs with the architectures and batch sizes in Table 2.

The configuration evaluated at the training iterate uses momentum $0.9$, mixing weight $\beta = 0.5$, and weight decay $5 \times 10^{-4}$ on CIFAR-10; the corresponding CIFAR-100 values are $0.9$, $0.3$, and $5 \times 10^{-4}$. Its peak learning rates are $0.2$ and $0.05$, respectively, and both decay to approximately zero. The configuration evaluated at the averaged iterate is given in Table 3 and retains half its peak learning rate. Each configuration uses the best hyperparameters found in its own search, allowing the comparison to evaluate each iterate under its selected setting.

At the training iterate $y$, hybrid momentum attains lower test loss than cosine SGD on both datasets: $0.1329$ versus $0.1536$ on CIFAR-10, and $0.9645$ versus $1.0279$ on CIFAR-100. At the averaged iterate $x$, the corresponding HyM losses are $0.1358$ and $0.7764$. The configuration evaluated at the training iterate has slightly lower loss on CIFAR-10, whereas the configuration evaluated at the averaged iterate has lower loss on CIFAR-100 and higher accuracy on both datasets. Thus, hybrid momentum is effective at the training iterate, with iterate averaging providing an additional benefit in the paired component study below.

Appendix F.3 separates these effects by evaluating every training trajectory at the training and averaged iterates.

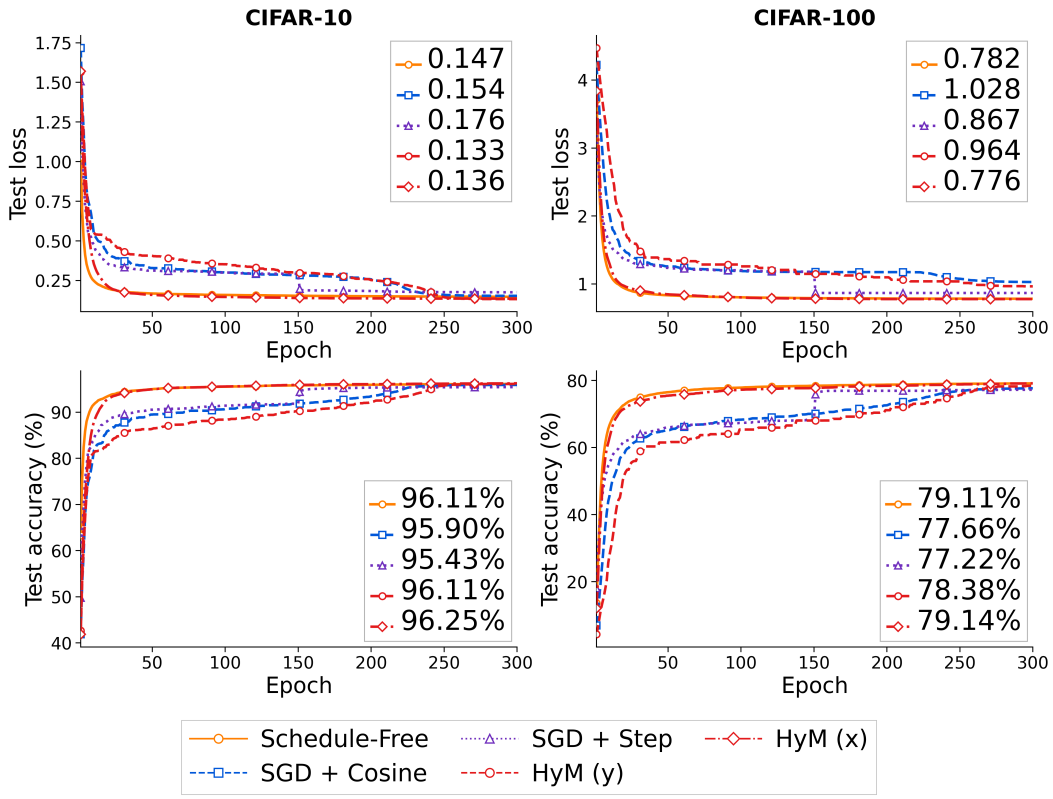


Figure 12: CIFAR comparison of the training and averaged iterates. Columns show CIFAR-10 and CIFAR-100; rows show test loss and test accuracy. HyM $(y)$ is evaluated at the training iterate without iterate averaging, while HyM $(x)$ is evaluated at the averaged iterate. The other curves reuse the baselines from Figure 4.

### F.3 COMPONENT ABLATION OF HyM ON CIFAR-10

We compare the fast momentum, slow momentum, their combination, and three replacements of the slow momentum by a fixed-decay EMA. Each training trajectory is evaluated at both the training iterate $\mathbf{y}$ and the averaged iterate $\mathbf{x}$, so the effect of iterate averaging can be separated from the choice of gradient history.

**Training and evaluation.** We train WideResNet-16-8 for 300 epochs on a fixed class-stratified split of CIFAR-10: 45,000 training images and 5,000 validation images, with 500 validation images per class. Batch size is 128 without dropping the final batch, giving 352 steps per epoch and $T = 105{,}600$ steps in total. The three peak learning rates $\eta_{\max} \in \{0.05, 0.1, 0.2\}$ define separate comparison groups, all of which are reported. All groups use $\mu = 0.99$, direction mixing $\beta_{\mathrm{dir}} = 0.95$, averaging $\beta_{\mathrm{avg}} = 0.95$, and coupled L2 decay $\lambda_{\mathrm{wd}} = 5 \times 10^{-4}$ on all parameters. The learning rate warms up linearly for 5,280 steps, then follows a cosine decay to $\eta_{\max}/2$. Training uses cross-entropy, deterministic cuDNN, and no cuDNN benchmarking. Augmentation is a random $32 \times 32$ crop after four-pixel reflection padding and a horizontal flip with probability $0.5$; each channel is normalized as $(\mathrm{pixel}/255 - 0.5)/0.5$.

Initialization, training-image order, and crop/flip randomness are shared across configurations. After every epoch, $\mathbf{y}$ and $\mathbf{x}$ are copied in turn into a separate evaluation model. For each evaluated iterate, Batch-Normalization statistics are reset and recalibrated by cumulative averaging on the same 6,400 unaugmented training images (50 batches). Neither parameters nor BN buffers are copied back into the training model. We report epoch-300 validation metrics.

**Directions with matched total weight.** Let $\mathbf{m}_t$ and $\mathbf{b}_t$ be the states of Algorithm 1, with input $\mathbf{g}_t = \nabla f(\mathbf{y}_t; \xi_t) + \lambda_{\mathrm{wd}} \mathbf{y}_t$. Use the quantities $Z_t, A_t, C_t$ and total coefficient sum $S_t = A_t + C_t$ defined in (19)–(20), with $\beta = \beta_{\mathrm{dir}}$. The normalized fast direction is $\widehat{\mathbf{b}}_t := \mathbf{b}_t / Z_t$. For the EMA replacements, define the bias-corrected direction $\widehat{\mathbf{g}}_t^{(\rho)} = \frac{1-\rho}{1-\rho^t} \sum_{i=1}^{t} \rho^{t-i} \mathbf{g}_i$. All variants update $\mathbf{y}_{t+1} = \mathbf{y}_t - \eta_t \mathbf{D}_t$, where

$$\mathbf{D}_t = \begin{cases} S_t \widehat{\mathbf{b}}_t, & \text{fast momentum only,} \\ S_t \mathbf{m}_t, & \text{slow momentum only,} \\ A_t \mathbf{m}_t + C_t \widehat{\mathbf{b}}_t, & \text{full HyM,} \\ A_t \widehat{\mathbf{g}}_t^{(\rho)} + C_t \widehat{\mathbf{b}}_t, & \text{slow-EMA replacement,} \quad \rho \in \{0.99, 0.999, 0.9999\}. \end{cases}$$

For a constant input gradient $\mathbf{g}$, every variant gives $\mathbf{D}_t = S_t \mathbf{g}$. This matches the total gradient weight, not the norm of each realized update. Fast momentum only retains HyM's factor $\mu(t-1)/t$; it is not ordinary SGD momentum. The EMA replacements retain the fast momentum and the rest of HyM, and are not full AdEMAMix. All variants use the same pre-update average $\mathbf{x}_t = (1 - a_t)\mathbf{x}_{t-1} + a_t \mathbf{y}_t$, with $a_t = [1 + (1 - \beta_{\mathrm{avg}})(t-1)]^{-1}$. After $T$ steps, the paired training and averaged iterates are $\mathbf{y}_{T+1}$ and $\mathbf{x}_T$.

Table 5: Final CIFAR-10 validation accuracy (%) in the component ablation. Each learning-rate group compares the same six configurations; $y$ and $x$ are paired evaluations at epoch 300. EMA rows replace only the slow momentum. These are validation results on the 45k/5k split, not the test accuracies in the main image-classification experiment.

| | $\eta_{\max} = 0.05$ | | $\eta_{\max} = 0.1$ | | $\eta_{\max} = 0.2$ | |
|---|---|---|---|---|---|---|
| Direction | $y$ | $x$ | $y$ | $x$ | $y$ | $x$ |
| Fast momentum only | 84.26 | 90.44 | 76.78 | 83.76 | 65.68 | 72.18 |
| Slow momentum only | 89.64 | 92.06 | 86.80 | 89.02 | 81.82 | 84.48 |
| Full HyM | 93.94 | 96.22 | 93.44 | 96.10 | 91.92 | 95.52 |
| EMA, $\rho = 0.99$ | 85.04 | 90.86 | 77.60 | 83.40 | 68.50 | 73.18 |
| EMA, $\rho = 0.999$ | 89.30 | 93.40 | 84.18 | 89.58 | 79.88 | 82.74 |
| EMA, $\rho = 0.9999$ | 92.94 | 95.62 | 92.56 | 95.64 | 89.46 | 93.76 |

**Results.** Full HyM has the highest final validation accuracy at both iterates in each learning-rate group (Table 5). The advantage over either individual momentum state is already present at the

training iterate, so iterate averaging alone does not account for this difference. Averaging improves final accuracy in all 18 configurations, by 2.22–6.98 percentage points; for full HyM the gains are 2.28, 2.66, and 3.60 points at the three learning rates. Among the EMA replacements, $\rho = 0.9999$ is closest to full HyM, with accuracies at the averaged iterate of $95.62\%$ versus $96.22\%$ at $\eta_{\max} = 0.05$, and $95.64\%$ versus $96.10\%$ at $\eta_{\max} = 0.1$.

Figure 13 shows the corresponding validation loss histories. Fast momentum only and the $\rho = 0.99$ replacement exhibit a pronounced intermediate loss increase at the larger learning rates; full HyM and the $\rho = 0.9999$ replacement avoid this large rebound. Full HyM also has the lowest final loss at both iterates in each group. These comparisons support using the two momentum states together under the shared training recipe, with an additional benefit from iterate averaging.

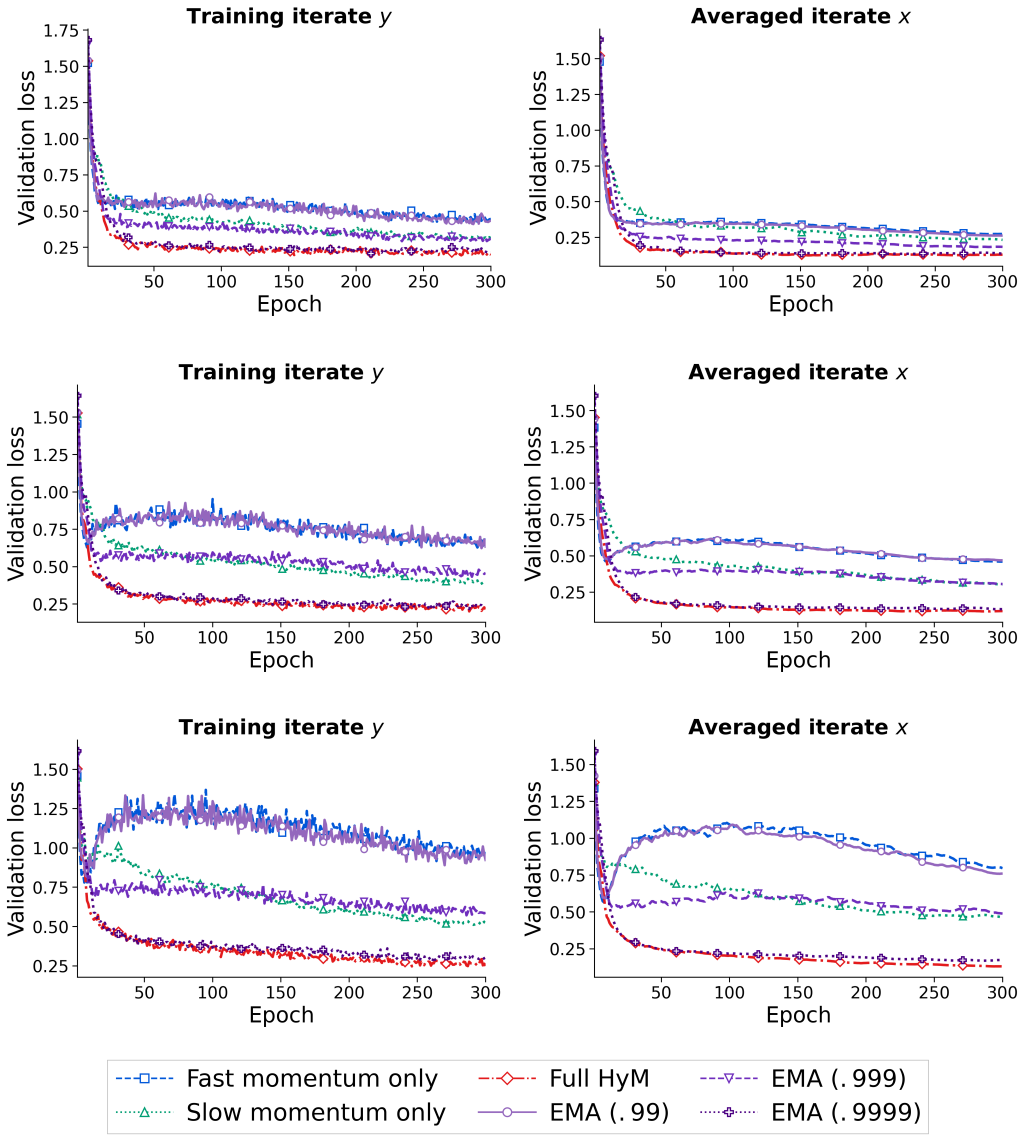


Figure 13: CIFAR-10 component ablation: validation cross-entropy at the training iterate $y$ (left) and averaged iterate $x$ (right). Rows use peak learning rates $0.05$, $0.1$, and $0.2$, respectively. Curves show per-epoch values. Every evaluation uses the same BN calibration protocol. "Slow EMA" retains the fast momentum and replaces only the slow momentum.

### F.4 IWSLT14 German–English

We use the IWSLT14 German–English corpus (Cettolo et al., 2014), with 160,239 training, 7,283 validation, and 6,750 test sentence pairs. The model is Fairseq's one-layer attention-based LSTM encoder–decoder preset (Ott et al., 2019), `lstm_wiseman_iwslt_de_en`, with 256-dimensional embeddings and hidden states, shared decoder input/output embeddings, dropout 0.3, and label smoothing 0.1. Each task uses one GPU, a dynamic token-batch limit of 4096, and a gradient-accumulation factor of one. Runs are capped at 55 epochs and 60,000 steps, with evaluation after every epoch. Tokenized BLEU uses beam size 5, length parameters 1.2 and 10, space detokenization, and removal of BPE continuation markers.

**Optimizer configuration.** We search the hidden-matrix and auxiliary learning rates, weight decay, and optimizer-specific momentum, mixing, and auxiliary second-moment coefficients. Table 6 reports the selected configurations. Both parameter groups use a 4,000-step linear warmup followed by linear decay to zero at step 60,000, retaining their peak learning-rate ratio throughout the schedule. Gradient clipping is disabled. Hidden weight matrices use five quintic NS iterations and the shape factor $\sqrt{\max\{1, m/n\}}$ for an $m \times n$ matrix, without an additional Nesterov term. Embeddings, output heads, and one-dimensional parameters belong to the auxiliary group. Muon uses AdamW for this group; `HyMuon` uses the adaptive `HyM` moments described in Appendix D and evaluates both groups at their averaged iterates.

Table 6: IWSLT14 optimizer hyperparameters. Auxiliary coefficients refer to AdamW for Muon and to the adaptive `HyM` branch for `HyMuon`.

| Hyperparameter | Muon | `HyMuon` |
|---|---|---|
| Hidden-matrix peak LR | 0.05 | 0.05 |
| Hidden-matrix momentum $\mu$ | 0.99 | 0.9 |
| `HyM` mixing weight $\beta$ | – | 0.9 |
| Hidden-matrix weight decay | 0.05 | 0.05 |
| Auxiliary peak LR | 0.01 | 0.05 |
| Auxiliary first-moment coefficient | 0.9 | 0.9 |
| Auxiliary second-moment coefficient | 0.98 | 0.95 |
| Auxiliary weight decay | 0.05 | 0.05 |
| Auxiliary $\varepsilon$ | $10^{-8}$ | $10^{-8}$ |

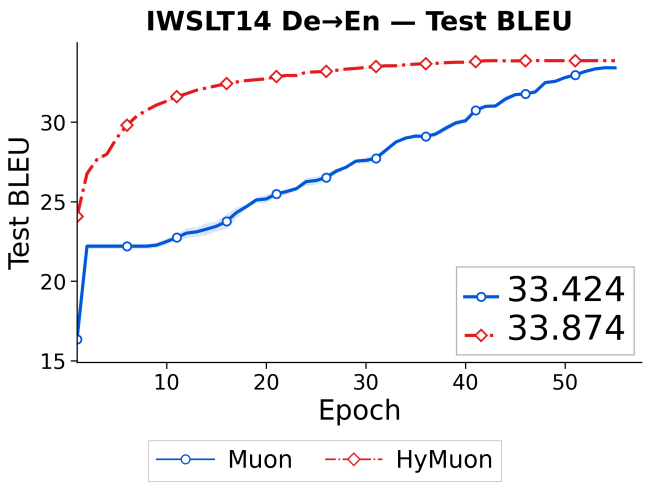


Figure 14: IWSLT14 German–English test BLEU accompanying the test-loss panel in Figure 5. Curves average repeated runs for each method with one-standard-error bands; panel legends report the endpoint means.

For the IWSLT14 test-loss panel in Figure 5 and the BLEU panel in Figure 14, the selected Muon and `HyMuon` configurations are each evaluated over repeated runs. At each epoch, the curves report

test metrics from the checkpoint with the highest validation BLEU observed so far. Curves are aligned by epoch and show the mean with one-standard-error bands.

**Averaged-iterate ablation.** We additionally evaluate a `HyMuon` variant that retains the same matrix and auxiliary updates formed from the two momentum states as the full method but evaluates and checkpoints the training iterate $y$, without constructing the averaged iterate $x$. The search covers the matrix and auxiliary learning rates, weight decay, matrix momentum, mixing weight, and auxiliary second-moment coefficient. The selected configuration uses matrix and auxiliary peak learning rates $0.05$ and $0.1$, matrix momentum $0.9$, mixing weight $0.9$, auxiliary coefficients $(0.9, 0.9)$, and weight decay $0.05$ and $0.1$ for the matrix and auxiliary groups, respectively, with a 4,000-step warmup followed by linear decay. It attains test BLEU $33.580$ and test loss $4.233$. The corresponding selected Muon and full `HyMuon` runs attain test BLEU $33.370$ and $33.930$, with test losses $4.229$ and $4.186$, respectively. Thus the variant evaluated at the training iterate improves BLEU over Muon, showing that the updates formed from the two momentum states remain useful without the averaged iterate, while evaluation at the averaged iterate provides a further improvement.

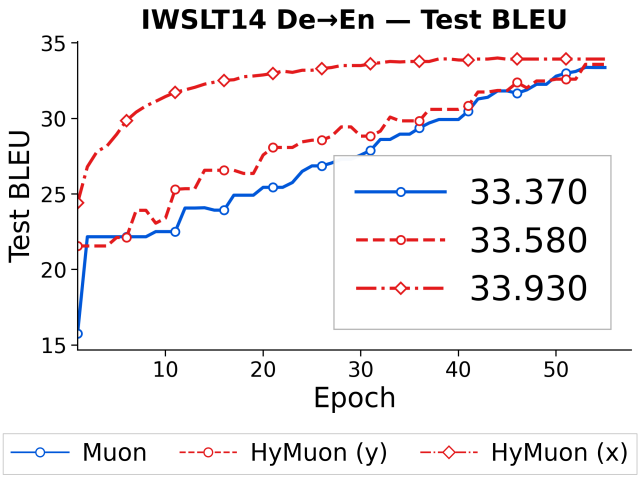


Figure 15: IWSLT14 averaged-iterate ablation in test BLEU. `HyMuon` $(y)$ retains the matrix and auxiliary updates formed from the two momentum states but evaluates the training iterate; `HyMuon` $(x)$ uses the averaged iterate. The curves show the selected configuration for each method.

### F.5 NanoGPT on OpenWebText

We train a GPT-2 Small model (Radford et al., 2019) with 12 layers, 12 attention heads, hidden size 768, context length 1024, zero dropout, and 124M parameters using nanoGPT (Karpathy, 2022) on OpenWebText (Gokaslan & Cohen, 2019). A micro-batch contains 12 sequences and the global gradient-accumulation factor is 40, giving 480 sequences or 491,520 tokens per optimizer step. Validation is run every 1000 steps and averages 200 randomly sampled mini-batches on the master process (2,457,600 validation tokens per evaluation). The two panels use distinct 100k- and 600k-step regimes, corresponding to 49.152B and 294.912B processed training tokens. Table 7 records the `HyMuon` configurations used in the two regimes. Training uses gradient clipping at norm 1 and a 2,000-step linear warmup. Cosine decay reaches its stated minimum at the end of each regime. `HyMuon` uses five NS iterations on hidden matrices and $\varepsilon = 10^{-8}$ in its auxiliary adaptive branch; its auxiliary first-moment coefficient equals the mixing weight, $0.99$.

In the 100k setting, AdamW uses learning rate $6 \times 10^{-4} \to 3 \times 10^{-5}$, $(\beta_1, \beta_2) = (0.9, 0.95)$, and weight decay $0.1$. Muon uses hidden-matrix learning rate $2 \times 10^{-2} \to 3 \times 10^{-5}$, momentum $0.95$, and zero weight decay; its auxiliary AdamW branch uses learning rate $3 \times 10^{-3} \to 3 \times 10^{-5}$, $(\beta_1, \beta_2) = (0.9, 0.95)$, and weight decay $0.1$.

In the 600k setting, AdamW uses learning rate $6 \times 10^{-4} \to 6 \times 10^{-5}$, $(\beta_1, \beta_2) = (0.9, 0.95)$, and weight decay $0.1$. Muon uses an auxiliary learning rate of $6 \times 10^{-4}$, a matrix learning rate of $0.02$,

Table 7: `HyMuon` hyperparameters in the GPT-2 124M OpenWebText experiments.

| **Hyperparameter** | **100k regime** | **600k regime** |
|---|---|---|
| Hidden-matrix LR | 0.01 | 0.01 |
| Hidden-matrix minimum LR | $3 \times 10^{-5}$ | $6 \times 10^{-5}$ |
| Hidden-matrix momentum $\mu$ | 0.95 | 0.95 |
| `HyM` mixing weight $\beta$ | 0.99 | 0.99 |
| Hidden-matrix weight decay | 0.2 | 0.2 |
| Auxiliary LR | 0.005 | 0.005 |
| Auxiliary minimum LR | $3 \times 10^{-5}$ | $6 \times 10^{-5}$ |
| Auxiliary $\beta_2$ | 0.999 | 0.999 |
| Auxiliary weight decay | 0.1 | 0.1 |
| Warmup steps | 2000 | 2000 |
| Learning-rate schedule | cosine | cosine |

and a minimum learning rate of $6 \times 10^{-5}$. Matrix momentum is $0.95$; the auxiliary Adam branch uses $(\beta_1, \beta_2) = (0.9, 0.95)$ and $\varepsilon = 10^{-8}$. Both groups use weight decay $0.01$. The matrix group contains parameters with at least two axes whose names contain neither `embed` nor `lm_head`; the remaining parameters use the auxiliary branch. Both groups use the 2,000-step warmup and cosine decay through 600k steps.

Each OpenWebText panel in Figure 5 shows the run using the configuration listed above through its stated training budget. The 100k and 600k panels are separate training regimes rather than continuations of the same runs.

At the last plotted step, the validation losses for AdamW, Muon, and `HyMuon` are approximately $2.905$, $2.915$, and $2.886$ in the 100k regime, and $2.866$, $2.886$, and $2.823$ in the 600k regime, respectively.

### F.6 LLaMA-1.3B pretraining on C4

This experiment compares `HyMuon` with Muon for LLaMA-1.3B pretraining on C4. We reproduce the Muon baseline reported in Table 2 of Gu & Xie (2026) using the training configuration in their Table 5 and Appendix B.1. Muon's peak learning rate $3 \times 10^{-4}$ follows that configuration. The model has 1.339B parameters, 24 layers, hidden size 2048, feed-forward size 5461, 32 attention heads, and a 32k vocabulary. C4/en is streamed with document packing and a `t5-base` tokenizer; each step contains 512 sequences of length 1024. Both methods train for 10,000 steps, 1,000 linear warmup steps, cosine decay to one tenth of the peak learning rate, gradient clipping at 1.0, and a 10,017,216-token validation evaluation every 1,000 steps. Each method processes 5,242,880,000 input tokens.

Muon follows the reproduced Mano recipe, using five NS iterations, momentum 0.95, matrix and auxiliary learning rates $3 \times 10^{-4}$, and weight decay 0.1. For `HyMuon`, we search the matrix and auxiliary learning rates, matrix momentum, mixing weight, auxiliary second-moment coefficient, and per-group weight decay. The selected configuration uses matrix and auxiliary learning rates $3 \times 10^{-2}$, mixing weight 0.99, Muon momentum 0.95, auxiliary second-moment coefficient 0.999, and per-group weight decay 0.2; evaluation swaps both parameter groups to their averaged iterates.

At 10k steps, validation loss and perplexity are $2.903$ and $18.221$ for Muon, and $2.777$ and $16.078$ for `HyMuon`, respectively.

### F.7 LLaMA-124M FineWeb benchmark

**Protocol and data.** We use the official framework of Semenov et al. (2025) and its 124M large-batch protocol (Section 4.1 and Figure 3(b) of that work). We run separate 8k-step (approximately 1B-token) and 16k-step (2.1B-token) experiments. Each budget starts a new training run with its own learning-rate schedule. The model has 123.59M parameters, 12 layers, hidden size 768, 12 attention

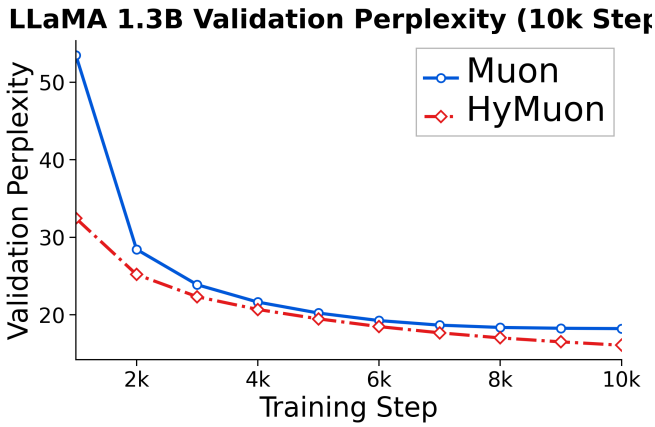


Figure 16: LLaMA-1.3B packed-C4 validation perplexity through 10k steps in the comparison against the reproduced Muon baseline. This is the exponential transform of the validation-loss curves in Figure 5; `HyMuon` is evaluated at its averaged iterate.

heads, SwiGLU feed-forward size 2048, RMSNorm, RoPE, and tied input/output embeddings. GPT-2 tokenization uses a padded vocabulary of 50,304 and context length 512.

All four methods use a common FineWeb `sample-10BT` subset and the same deterministic split. Documents are processed in shard order with an end-of-text token appended; the first 10,000,001 token IDs form validation, and subsequent documents supply 8,400,000,001 training token IDs. Four-way data parallelism uses 64 sequences per rank, giving a global batch of 256 sequences and 131,072 tokens per step. The two budgets process 1,048,576,000 and 2,097,152,000 training tokens, respectively. Dropout is zero and the gradient-norm clipping threshold is 0.5. Each parameter group's learning rate warms up over 2,000 steps and then decays by a cosine schedule to 1% of its peak.

**Optimizer configurations.** AdamW and AdEMAMix use the benchmark's published 124M recipes; Muon uses its basic-Muon implementation and large-batch configuration (Figure 4 and Table 12 of Semenov et al. (2025)). The AdEMAMix algorithm itself is from Pagliardini et al. (2025). Table 8 lists the fixed configurations used for Figure 5. AdamW and AdEMAMix apply weight decay to linear-layer weights, excluding the tied embedding/output weights and normalization parameters. For Muon and `HyMuon`, hidden attention/MLP matrices form the matrix group; the tied embedding/output head and RMSNorm parameters form the auxiliary group. Both matrix optimizers use five NS iterations. Basic Muon uses Nesterov momentum and applies decay only to its AdamW auxiliary group. `HyMuon` uses ordinary momentum, decoupled decay in both groups, and the adaptive auxiliary recurrence of Appendix D. Its `HyM` mixing weight is $\beta = 0.9$. The auxiliary decay coefficients are $(\rho_1, \rho_2) = (0.97, 0.999)$ at 1B tokens and $(0.99, 0.999)$ at 2.1B tokens. The model uses RMSNorm, so no BatchNorm recalibration is required.

Table 8: Fixed optimizer hyperparameters for the 124M FineWeb comparison. Baseline columns apply to both budgets; the `HyMuon` column gives the 1B configuration, with its 2.1B changes specified below. Learning rates are peak values; paired entries are matrix/auxiliary values. For AdamW and AdEMAMix, the single rate applies to all parameters.

| | AdamW | AdEMAMix | Muon | `HyMuon` |
|---|---|---|---|---|
| Learning rate | 0.001 | 0.001 | 0.01/0.001 | 0.045/0.045 |
| Weight decay | 0.1 | 0.1 | 0/0.1 | 0.1/0.1 |
| Matrix momentum | – | – | 0.95 | 0.85 |
| Adaptive $(\beta_1, \beta_2)$ | $(0.8, 0.999)$ | $(0.9, 0.999)$ | $(0.8, 0.999)$ | $(0.97, 0.999)$ |
| Adaptive $\varepsilon$ | $10^{-8}$ | $10^{-8}$ | $10^{-8}$ | $10^{-10}$ |
| `HyM` mixing weight $\beta$ | – | – | – | 0.9 |
| Slow EMA $\beta_3$ | – | 0.999 | – | – |
| Slow EMA multiplier $\alpha$ | – | 8 | – | – |

At 2.1B tokens, `HyMuon` uses matrix/auxiliary learning rates $0.015/0.015$ and auxiliary betas $(0.99, 0.999)$; its remaining settings are unchanged. AdEMAMix warms up both its slow decay and mixing multiplier over the full training horizon: 8,000 or 16,000 steps, respectively. `HyMuon` uses the adaptive `HyM` recurrence in Appendix D for both token budgets. It computes gradients at the training iterate and evaluates all parameter groups at the averaged iterate; after evaluation, the training iterates are restored. AdamW, AdEMAMix, and Muon evaluate their current training parameters.

**Evaluation and curves.** Validation is evaluated at initialization, every 115 steps, and at the final step (8,000 or 16,000). Intermediate evaluations use a fixed set of 64 batches of 64 sequences; the final evaluation uses all complete batches of the validation reader. Figure 5 shows the resulting validation-loss curves.