Adaptive Memory Momentum via a Model-Based Framework for Deep Learning Optimization

Kristi Topollai
New York University

Anna Choromanska
New York University

Abstract

The vast majority of modern deep learning models are trained with momentum-based first-order optimizers. The momentum term governs the optimizer's memory by determining how much each past gradient contributes to the current convergence direction. Fundamental momentum methods, such as Nesterov Accelerated Gradient and the Heavy Ball method, as well as more recent optimizers such as AdamW and Lion, all rely on the momentum coefficient that is customarily set to $\beta = 0.9$ and kept constant during model training, a strategy widely used by practitioners, yet suboptimal. In this paper, we introduce an adaptive memory mechanism that replaces constant momentum with a dynamic momentum coefficient that is adjusted online during optimization. We derive our method by approximating the objective function using two planes: one derived from the gradient at the current iterate and the other obtained from the accumulated memory of the past gradients. To the best of our knowledge, such a proximal framework was never used for momentum-based optimization. Our proposed approach is novel, extremely simple to use, and does not rely on extra assumptions or hyperparameter tuning. We implement adaptive memory variants of both SGD and AdamW across a wide range of learning tasks, from simple convex problems to large-scale deep learning scenarios, demonstrating that our approach can outperform standard SGD and Adam with hand-tuned momentum coefficients. Finally, our work opens doors for new ways of inducing adaptivity in optimization.

Proceedings of the 29[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

1 INTRODUCTION

Stochastic Gradient Descent (SGD) (Bottou, 1998) and its variants (Sutskever, 2013; Kingma, 2015) are widely used for training deep learning models due to their simplicity and efficiency. Many popularly used first-order optimizers rely on Heavy Ball Momentum, commonly defined as:

$$d_{t+1} = \beta d_t + (1-\beta)\nabla f(x_t), \quad x_{t+1} = x_t - \eta d_{t+1}, \quad (1)$$

where η is the learning rate, x_t denotes model parameters at iteration t, f is the loss function, and β is the momentum coefficient. Momentum methods augment the current gradient with an exponentially weighted moving average of past gradients, where β determines the optimizer's "memory", that is, how much past gradients influence the update direction.

In deterministic settings, momentum methods can provably accelerate convergence under mild assumptions (Polyak, 1964; Nesterov, 1983). Achieving such acceleration in practice with Heavy Ball (HB) momentum (Polyak, 1964) or Nesterov Accelerated Gradient (NAG) (Nesterov, 1983) requires carefully tuning η and β, or using time-dependent schedules (Nesterov, 1983). However, in non-convex or stochastic settings, these accelerated rates are not guaranteed. Surprisingly, despite the widespread adoption of momentum methods in deep learning due to their empirical effectiveness, theoretical analyses offer no justification of why they show empirical gains: under similar assumptions, stochastic Heavy Ball (HB) momentum achieves at best the same convergence rate as plain SGD, but no better. This disconnect between theory and practice is striking, especially given the near-universal use of a fixed momentum coefficient, typically $\beta = 0.9$, across models, datasets, and optimization setups.

But is this fixed choice really optimal? Intuitively, it seems unlikely that a single value of β should work equally well throughout the whole training process and across different data sets and models. Instead, we ask: can momentum adapt over time to better match the optimization landscape? In this work, we propose a sim-

ple yet principled answer, a time-varying momentum coefficient that evolves with the optimization process. We refer to this mechanism as *adaptive memory*, as it dynamically adjusts the extent to which the optimizer relies on past gradients at each step.

To derive this adaptive coefficient, we take inspiration from model-based optimization techniques (Asi, 2019; Davis, 2019) and the proximal bundle method (Krzysztof C Kiwiel, 2006). In this framework, the objective function $f(x)$ is approximated by a surrogate model $f_t^m(x)$, and at each step the following problem has to be solved:

$$x_{t+1} \in \operatorname*{argmin}_x f_t^m(x) + \frac{1}{2\eta}\|x - x_t\|^2. \tag{2}$$

We extend this framework by constructing $f_t^m(x)$ from two planes: one from the current gradient $\nabla f(x_t)$ and one from the previous descent direction $\frac{1}{\eta}(x_{t-1} - x_t)$ which encodes the accumulated momentum. Under this model, the proximal step in Equation (2) yields the update:

$$x_{t+1} = x_t - (1-\beta_t)\eta\nabla f(x_t) + \beta_t(x_t - x_{t-1}), \tag{3}$$

where β_t is an *adaptive* momentum coefficient computed in closed form from the model. This gives rise to our proposed *adaptive memory momentum* schemes. **Our contributions are summarized as follows:**

i) Motivation. We begin with a simple yet revealing empirical demonstration: using a fixed momentum coefficient can lead to suboptimal convergence, even when finely tuned. Identifying the optimal fixed value of β is often tedious, and more importantly, a static choice fails to adapt to the dynamics of the training process. This observation motivates the need for a time-adaptive momentum scheme.

ii) Methodology. We propose a novel approximate model of the loss function that combines two planes, one based on the current gradient and the other on the accumulated momentum. This model leads naturally to a reinterpretation of the proximal model-based update rule as an *adaptive memory* momentum scheme, in which the momentum coefficient evolves throughout optimization. We then incorporate our adaptive memory mechanism into both SGD and AdamW, yielding simple, yet effective, new variants of these optimizers. In each case, the momentum coefficient is updated dynamically based on our model-based formulation.

iii) Experimental Validation. We evaluate our approach across a wide range of settings, from deterministic convex problems to large-scale deep learning tasks. Beyond consistently outperforming fixed-momentum baselines, our adaptive memory methods offer significant benefits in challenging training regimes, particularly at high learning rates (Figure 6) and in the early stages of optimization. An important implication of adaptive memory is that it can offer an alternative to learning rate warm-up or scheduling (Figure 5), providing a robust and tuning-free alternative that could simplify the training pipeline of large models.

2 RELATED WORK

2.1 Momentum Methods

Momentum was first introduced by Polyak (Polyak, 1964) in the form of the Heavy Ball (HB) method, which incorporates the previous iterate into the update:

$$x_{t+1} = x_t - \eta\nabla f(x_t) + \beta(x_t - x_{t-1}), \tag{4}$$

where $\beta \in [0, 1)$ is the momentum coefficient. For L-smooth and m-strongly convex quadratic functions, tuned momentum yields accelerated convergence over gradient descent. Nesterov's Accelerated Gradient (NAG) (Nesterov, 1983) achieves similar benefits, and attains the optimal $O(1/t^2)$ rate for L-smooth convex functions.

Momentum has since been adapted to stochastic settings (Tseng, 1998; Ruszczyński, 1987), where it provably matches the convergence rate of SGD (Y. Yan, 2018; Sebbouh, 2021). However, the theoretical justification for its often superior empirical performance under common settings remains incomplete. Despite this gap, HB-style momentum has become a standard component in deep learning optimizers, offering faster convergence and often better generalization (Sutskever, 2013). The practical version used in most frameworks is [1]:

$$d_{t+1} = \beta d_t + (1-\beta)g_t, \qquad x_{t+1} = x_t - \eta d_{t+1}, \tag{5}$$

where g_t is a stochastic gradient such that $\mathbb{E}[g_t] = \nabla f(x_t)$. The success of momentum has motivated extensive theoretical analysis (Jelassi, 2022; Mai, 2020; Hu, 2009; Gitman, 2019; Y. Liu, 2020) and is used in numerous optimization algorithms. These include Adam(W) (Kingma, 2015; Loshchilov, 2017a) and its variants (You, 2020; Shazeer, 2018; Pagliardini, 2025), as well as other momentum-based methods like Lion (X. Chen, 2023), Sophia (H. Liu, 2024), SOAP (Vyas, 2025), MARS (Yuan, 2025), and Muon (Jordan, 2024; J. Liu, 2025). Although different, all of these methods rely on a fixed momentum coefficient β and do not adapt it during training.

2.2 Model-Based Optimization

We build on the framework of proximal point methods (Rockafellar, 1976), which update iterates via the

[1] PyTorch uses an equivalent momentum update with $d_{t+1} = \beta d_t + g_t$ and appropriate learning rate scaling.

objective:

$$x_{t+1} \in \operatorname*{argmin}_x f(x) + \frac{1}{2\eta}\|x - x_t\|_2^2. \tag{6}$$

Since exactly solving this is often intractable, it is common to replace $f(x)$ with a simpler surrogate $f_t^m(x)$ (Asi, 2020), which yields a general model-based optimization framework used in both deterministic and stochastic settings (Davis, 2019; Asi, 2019; Chadha, 2022). Gradient Descent, SGD (Robbins, 1951), and Subgradient Descent (Polyak, 1987) can all be recovered by linearizing f around x_t and substituting appropriate gradient or subgradient terms. Second-order updates such as Newton's method arise from quadratic approximations. More sophisticated models, such as cutting plane bundles (Kelley, 1960)(Equation 7), lead to the Proximal Bundle Method (Krzysztof Czesław Kiwiel, 1983), useful in non-smooth optimization:

$$f_t^m(x) = \max_{i=1,\dots,T} \big(f(x_i) + \langle g_i, x - x_i\rangle\big), \tag{7}$$

where g_i is a subgradient that defines a cutting plane at x_i. Model-based approaches have also been used to derive adaptive learning rates, such as the Polyak step size (Polyak, 1987; Loizou, 2021), and more recently, adaptive learning rates for momentum gradient descent (Schaipp, 2024; X. Wang, 2023; Oikonomou, 2025). Inspired by these ideas, we instead use a model-based approximation of the loss to derive adaptive momentum coefficients.

2.3 Adaptive Methods

In both stochastic and non-stochastic optimization, choosing effective values for the learning rate and momentum coefficient is critical. In smooth and (strongly) convex settings, their optimal values depend on the condition number of the function. Consequently, a significant line of research focuses on adaptively estimating the strong convexity and smoothness constants to tune these parameters for various accelerated methods (Malitsky, 2020; Barré, 2020; Saab Jr, 2022).

In stochastic optimization, most work focuses on adaptive learning rates. Methods such as stochastic Polyak step sizes (Loizou, 2021; Orvieto, 2022) adapt the step size based on the suboptimality gap $f(x_t) - f^*$, while others rely on distance-to-optimum estimates (Defazio, 2023; Ivgi, 2023). In deep learning, adaptive diagonal pre-conditioners, used in AdaGrad (Duchi, 2011), Adam (Kingma, 2015), and related methods (Hinton, 2012; H. Liu, 2024), are standard for per-parameter step size adjustment. Beyond adaptive schemes, fixed learning rate schedules (Smith, 2017; Loshchilov, 2017b) and warm-up strategies (Goyal, 2017) are widely used, particularly in large-scale deep learning. Warm-up, which gradually increases the learning rate at the start of training, is now common in training LLMs (Wortsman, 2024). However, it is not truly adaptive and requires manual tuning of hyperparameters such as warm-up duration, making it brittle in long or infinite horizon training regimes where the total number of steps may be unknown. Consequently, there is a growing interest in methods that alleviate these shortcomings (Kalra, 2024; Kosson, 2024).

In contrast, adapting the momentum coefficient has received little attention. Heuristic schedules (Sutskever, 2013; J. Chen, 2022) increase or decrease momentum over time, but have not been adopted in practice. Restart-based methods (Giselsson, 2014; O'donoghue, 2015) propose clearing the momentum buffer when certain technical conditions are met (O'donoghue, 2015) or according to pre-defined schedules (B. Wang, 2022) and, while effective, they often require handcrafted restart rules. Conjugate gradient (CG) and nonlinear CG (Hager, 2006) also adapt the weight on past directions and can be viewed as momentum methods with dynamic coefficients. However, they rely on line search to select learning rates. In this work, we focus solely on adapting the momentum coefficient β, without also requiring learning rate adaptation.

3 METHOD

3.1 A Simple Motivation

We begin by analyzing the deterministic setting and, before introducing our proposed method, we present a motivational example to highlight two key limitations of using a fixed, constant momentum coefficient. In Figure 1, we consider an unconstrained quadratic optimization problem, $\min\{x^T Ax + b^T x + c\}$, and plot the objective gap $f(x_t) - f^*$, where f^* is the minimum, for various fixed momentum coefficients. The results reveal that a fixed momentum is inherently suboptimal, consistently being outperformed by an adaptive strategy. Furthermore, optimization around the optimal value of the momentum coefficient, β^*, is highly unstable, i.e., a slightly different coefficient than the optimal radically degrades the performance.

3.2 Approximate Cutting Planes Framework

Inspired by bundle methods (Krzysztof C Kiwiel, 2006), our approach for deriving an adaptive coefficient β for momentum methods is based on the following model of the function f:

$$f_t^m(x) = \max\Big\{f(x_t) + g_t^\top (x - x_t), \tag{8}$$
$$\hat{f}(x_t) + \frac{1}{\eta}(x_{t-1} - x_t)^\top (x - x_t)\Big\},$$

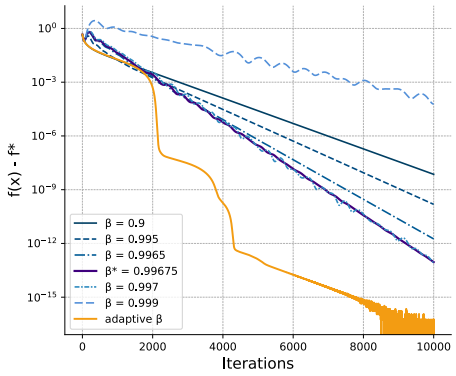


Figure 1: The Heavy-ball method with a fixed β vs our scheme. We plot the error $f(x_t) - f^*$ over t.

where $\frac{1}{\eta}(x_{t-1} - x_t)$ is the previous descent direction, the momentum term, $g_t = \nabla f(x_t)$, and $\hat{f}(x_t)$ is the momentum plane bias, which we discuss further in the end of this section. This formulation uses the cutting plane at the current point, defined by the first-order approximation of f at x_t, to construct a model of the function. To refine this model, we propose incorporating an additional plane with a slope determined by the direction $\frac{1}{\eta}(x_{t-1} - x_t)$.

Furthermore, inspired by (Kim, 2022; Q. Deng, 2021), we introduce an extra regularization term and regularization parameter λ to control the extent to which the descent direction aligns with the previous descent direction $\frac{1}{\eta}(x_{t-1} - x_t)$. Adding this regularization term to the proximal model-based objective leads to the following update at each step:

$$x_{t+1} \in \operatorname*{argmin}_x f_t^m(x) + \frac{\lambda+1}{2\eta}\|x - x_t\|_2^2 + \frac{\lambda}{\eta}\langle x_{t-1} - x_t, x - x_t\rangle. \tag{9}$$

The piecewise nature of the truncated model in $f_t^m(x)$ has a key property, when used in Equation (9), as detailed in the Supplement Section A along with all the derivations, the minimizer can be expressed in the following heavy-ball update:

$$x_{t+1} = x_t - \frac{\eta}{\lambda+1}\left(a_1 \nabla f(x_t) + (a_2 + \lambda)\frac{x_{t-1} - x_t}{\eta}\right). \tag{10}$$

where $a_1, a_2 \in [0, 1]$, with $a_1 + a_2 = 1$, are the dual variables associated with Equation (9). By setting $a_2 = \beta$ and $a_1 = 1 - \beta$, and defining the velocity vector $d_t = \frac{1}{\eta}(x_{t-1} - x_t)$, these variables are determined by solving the quadratic program:

$$\max_{0\le\beta\le 1}\left\{ -\frac{\eta}{2(\lambda+1)}\left\|(1-\beta)g_t + \beta d_t + \lambda d_t\right\|_2^2 + (1-\beta)f(x_t) + \beta\hat{f}(x_t)\right\}. \tag{11}$$

The solution to which, is given by:

$$\beta_t^* = \operatorname*{Clip}_{[0,1]}\left(\frac{(\hat{f}(x_t) - f(x_t))(\lambda+1)}{\eta} - \langle d_t - g_t, g_t + \lambda d_t\rangle}{\|d_t - g_t\|_2^2}\right). \tag{12}$$

Similar to aggregation in bundle methods, the computed β_t^* is also used to construct the approximation plane for the next iteration. This approach allows us to view the proximal step in Equation (9) as a momentum method with an adaptive momentum coefficient:

$$d_{t+1} = \frac{\beta_t^* + \lambda}{1+\lambda} d_t + \frac{1 - \beta_t^*}{1+\lambda} g_t, \qquad x_{t+1} = x_t - \eta d_{t+1}. \tag{13}$$

The scaling factor $\lambda + 1$ is introduced in order to define the momentum vector as a convex combination of the gradient and the current momentum. This ensures consistency with the current definition of the momentum gradient descent of Equation (1). The resulting algorithm is captured in Algorithm 1.

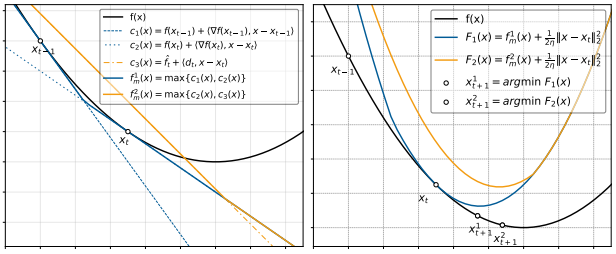


Figure 2: On the left, blue represents a typical cutting planes model, while yellow depicts our model, which may overestimate the function at x_t. On the right, we observe that with our model, the solution to Equation 6 lies closer to f^*, thus overestimation can get closer to the optimum.

Regarding the definition of the momentum plane in the model f_t^m, aggregation in proximal bundle methods typically sets $\hat{f}(x_{t+1}) = f_t^m(x_{t+1}) \le f(x_t)$, for the next iteration, leveraging the property that $d_{t+1} \in \partial f_t^m(x_{t+1})$ (Krzysztof Czesław Kiwiel, 1983). Although underestimating the true value of the objective ensures convergence and is used extensively in the literature on bundle methods, our experiments reveal that relying on overestimation and using $\hat{f}(x_t) = f(x_{t-1})$ instead (note that it is possible to have $\hat{f}(x_t) > f(x_t)$) results in faster convergence. An intuitive illustration of this mechanism is provided in Figure 2. This is a practical and efficient choice, shown in our experiments to yield strong empirical performance without added overhead. To support our argument, a detailed discussion is provided in the Supplement Section B, where we prove that for quadratics, minimizing $\|x_{t+1} - x^*\|_2^2$ under Equation 5 with respect to β implies $\hat{f}(x_t) > f(x_t)$ for all t.

3.3 Pre-conditioning and Decoupled Weight decay

The flexibility of the model-based approach allows seamless extension to state-of-the-art optimizers. Algorithms like Adam (Kingma, 2015) and AdaGrad (Duchi, 2011) use diagonal pre-conditioners to scale updates. To incorporate these into our Adaptive Memory framework, we replace the Euclidean metric with one induced by the pre-conditioner P_t, where $\langle x, y\rangle_{P_t} = x^\top P_t y$ for symmetric positive definite P_t. Weight decay can be accounted for by modeling the regularized function $f_{\mu\|\|}(x) = f(x) + \frac{\mu}{2}\|x\|_2^2$. However, optimizers like AdamW (Loshchilov, 2017a) implement decoupled weight decay, which is handled separately. We can incorporate a preconditioner and decoupled weight decay by solving at each step:

$$x_{t+1} \in \operatorname*{argmin}_x f_t^m(x) + \frac{1}{2\eta}\|x - x_t\|_{\tilde{P}_t}^2 + \lambda\langle d_t, x - x_t\rangle_{P_t} + \frac{\mu}{2}\|x\|_{\tilde{P}_t}^2, \tag{14}$$

where $\tilde{P}_t = (I + \lambda P_t)P_t$ is introduced to appropriately scale the momentum term. Solving this optimization problem yields the following update rules:

$$\begin{aligned} d_{t+1} &= (\lambda P_t + I)^{-1}\big((\beta_t^* I + \lambda P_t)d_t + (1-\beta_t^*)g_t\big), \\ x_{t+1} &= \frac{1}{1+\mu\eta}\big(x_t - \eta P_t^{-1} d_{t+1}\big). \\ \beta_t^* &= \operatorname*{Clip}_{[0,1]}\left(\frac{\mu\langle x_t, g_t - d_t\rangle}{\|d_t - g_t\|_{\tilde{P}_t^{-1}}^2}\right. \\ &\quad \left. + \frac{\frac{(1+\mu\eta)(\hat{f}(x_t) - f(x_t))}{\eta} - \langle d_t - g_t, g_t + \lambda P_t d_t\rangle_{\tilde{P}_t^{-1}}}{\|d_t - g_t\|_{\tilde{P}_t^{-1}}^2}\right). \end{aligned} \tag{15}$$

By using the Adam pre-conditioner: $P_t = (1-\beta_1^t)\mathrm{diag}\big(\epsilon + \sqrt{\frac{v_t}{1-\beta_2^t}}\big)$ where $v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t \odot g_t)$, we derive the AM-AdamW [2] optimizer. This method retains the benefits of AdamW while incorporating momentum adaptivity through the AM framework.

3.4 Adaptive Memory for Stochastic Gradient Descent

We found that the most variance-sensitive component in the adaptive momentum formula is the difference $\Delta f = f(x_{t-1}, s_t) - f(x_t, s_t)$, where $f(x_t, s_t)$ are stochastic function evaluations using a data batch s_t sampled at time t. Evaluating the loss at two different points on the same batch is inefficient, while using different batches introduces significant noise and instability into the estimate. To address this, we apply two practical modifications detailed in Algorithm 1:

- Clip β to the interval $[0, \beta_{\max}]$, with $\beta_{\max} < 1$ (0.9 in experiments of Section 4).
- Replace $f(x_{t-1}, s_t) - f(x_t, s_t)$ with a first-order approximation around x_t:

$$\Delta f \approx \nabla f(x_t, s_t)^\top (x_{t-1} - x_t) = \eta g_t^\top d_t,$$

We apply similar adjustments to the AdamW setting. While our notation assumes a fixed learning rate η, our framework also supports dynamic learning rates. Full algorithmic and implementation details for all Adaptive Memory variants are provided in the Supplement Section D.

Algorithm 1 Adaptive Memory–Momentum. AM-MGD, AM-MSGD

Initialize: η, λ, x_0, $d_0 = \nabla f(x_0)$, $\beta_{\max}, d_0 = g_0$
for each iteration $t = 1, 2, \ldots, T$ **do**
  $g_t = \nabla f(x_t)$, $\nabla f(x_t, s_t)$
  $\beta_t = \dfrac{(\hat{f}(x_t) - f(x_t))(\lambda+1) - \eta\langle d_t - g_t,\ g_t + \lambda d_t\rangle}{\eta\|d_t - g_t\|_2^2}$
  $\beta_t = \dfrac{(\lambda+1)g_t^\top d_t - \langle d_t - g_t,\ g_t + \lambda d_t\rangle}{\|d_t - g_t\|_2^2}$
  $\beta_t = \min(\max(\beta_t, 0), 1)$, $\min(\max(\beta_t, 0), \beta_{\max,t})$
  $d_{t+1} = \frac{\beta_t + \lambda}{1+\lambda} d_t + \frac{1-\beta_t}{1+\lambda} g_t$
  $x_{t+1} = x_t - \eta d_{t+1}$
end for

3.5 Convergence Guarantees

We provide convergence guarantees for our proposed method in the standard finite-sum setting:

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^n f_{S_i}(x),$$

where each f_{S_i} denotes the loss associated with a mini-batch S_i, and the goal is to minimize the average loss f. At iteration t, let $g_t := \nabla f_{S_t}(x_t)$, where S_t denotes the sampled mini-batch. We proceed under the following assumptions.

Assumption 3.1 (Smoothness)**.** The function f is L-smooth if for all $x, y \in \mathbb{R}^d$,

$$f(y) \le f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2. \tag{16}$$

Assumption 3.2 (Bounded stochastic gradient norm)**.** There exists $G > 0$ such that, for all x,

$$\mathbb{E}_S\big[\|\nabla f_S(x)\|^2\big] \le G^2. \tag{17}$$

We additionally assume that the stochastic gradient is unbiased:

$$\mathbb{E}[g_t \mid x_t] = \nabla f(x_t).$$

[2] for small η, $\frac{1}{1+\eta\mu} \approx 1 - \eta\mu$ and $\frac{\eta}{1+\eta\mu} \approx \eta$ (Zhuang, 2022)

For our Adaptive Memory Momentum SGD, presented in Algorithm 1, we establish convergence guarantees under standard assumptions. The analysis leverages the structural bound

$$\beta_t \|d_t - g_t\|_2^2 \le \|g_t\|_2^2,$$

which follows directly from the construction of the adaptive momentum coefficient β_t.

Theorem 3.3 (Convex Convergence). *Assume each f_{S_i} is convex, Assumption 3.2 holds, and $\mathbb{E}[g_t \mid x_t] = \nabla f(x_t)$. Let $x^* \in \arg\min_x f(x)$, and run the method with $\eta = 1/\sqrt{T}$ and $\beta_{\max} = 1/T$. Define*

$$x_T^{(a)} = \frac{\sum_{t=0}^{T-1} a_t x_t}{\sum_{t=0}^{T-1} a_t}, \qquad a_t = \left(1 + \frac{1}{T}\right)^{-(t+1)}.$$

Then

$$\mathbb{E}\big[f(x_T^{(a)}) - f(x^*)\big] \le \frac{1}{\sqrt{T}} \left(\|x_0 - x^*\|^2 + \frac{5}{2} G^2 \right). \quad (18)$$

Theorem 3.4 (Non-convex Convergence). *Assume f satisfies Assumption 3.1, Assumption 3.2 holds, and $\mathbb{E}[g_t \mid x_t] = \nabla f(x_t)$. Run the method with $\eta = 1/\sqrt{T}$ and $\beta_{\max} = c\eta$ for some $0 < c < 1$. Then*

$$\min_{0 \le t \le T-1} \mathbb{E}[\|\nabla f(x_t)\|_2^2] \le \frac{\mathbb{E}[f(x_0) - f(x_T)] + \frac{1+8L}{4} G^2}{\sqrt{T}(1-c)}. \quad (19)$$

Our convergence guarantees place AM within the standard landscape of stochastic first-order methods. Under smoothness and bounded stochastic gradient norms, we obtain the classical $O(1/\sqrt{T})$ rate, consistent with existing analyses of SGD and heavy-ball under comparable assumptions (Y. Yan, 2018; Sebbouh, 2021). The main technical obstacle is that β_t is itself a nonlinear function of the current stochastic gradient g_t, so the proof must control terms such as $-2\eta\beta_t\langle x_t - x^*, d_t - g_t\rangle$, and β_t cannot be treated as an external parameter inside expectations.

This is closely related to the difficulty that appears in analyses of adaptive step sizes, where the step size is also chosen from the same stochastic information used in the update, breaking the usual martingale structure unless additional assumptions are imposed (Loizou, 2020; Orvieto, 2022). In our setting, the key structural relation is $\beta_t\|d_t - g_t\|^2 \le \|g_t\|^2$, which follows directly from the definition of β_t and keeps the adaptive correction term controlled. At the same time, as in fixed-momentum and adaptive-step-size methods, the cleanest stochastic guarantees are often obtained when momentum is damped or effectively small (X. Wang, 2023; Oikonomou, 2025; Schaipp, 2024; Loizou, 2020). Our restriction on $\beta_{\max}$ should be viewed in this same light.

4 EXPERIMENTS

4.1 Convex Problems

For our convex experiments, we used logistic regression on 9 datasets from the LIBSVM repository (Chang, 2011) (4 shown here; full results in the Supplement Section F.1). Figure 4 shows that AM-MGD consistently outperforms constant momentum across all datasets. We compare with the commonly used $\beta = 0.9$ and the optimal β^*, found via grid search to minimize the final loss. Notably, β^* can exhibit instability or nonmonotonicity, whereas AM-MGD achieves lower loss without tuning.

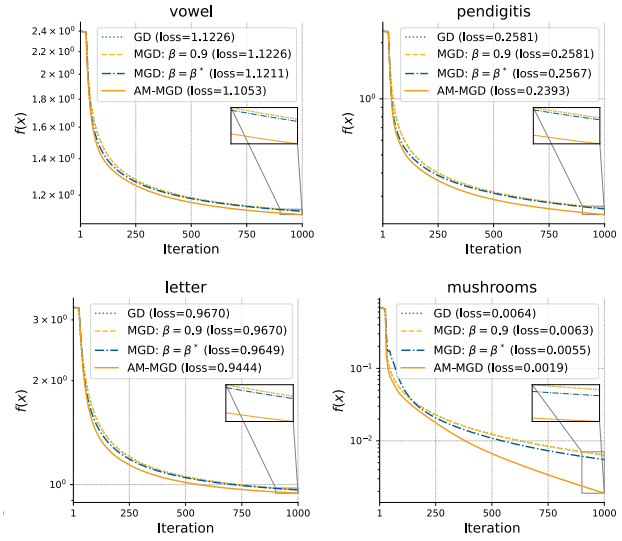


Figure 4: Logistic Loss over time, AM-MGD (red curve) outperforms fixed momentum on all 4 experiments. GD and MGD with $\beta = 0.9$ practically overlap. The λ was fixed to 0 in all runs.

4.2 Image Classification

We evaluate AM-MGD on standard image classification benchmarks, including CIFAR-10, CIFAR-100 (Alex, 2009), and ImageNet (Russakovsky, 2015), using VGG (Simonyan, 2015), ResNets (He, 2016), and Wide-ResNets (Zagoruyko, 2016). For these experiments, we directly adopt the training configuration of MoMo (Schaipp, 2024), which likewise considers damped-momentum SGD, and plug our adaptive-momentum update into this setup without performing a separate hyperparameter search for AM-MGD. Thus, all reported improvements are obtained under hyperparameters tuned for the vanilla fixed-momentum baseline, not for our method. Throughout, we keep the AM-specific hyperparameters fixed at $\beta_{\max} = 0.9$ and $\lambda = 0.1$; we do not retune them across datasets or

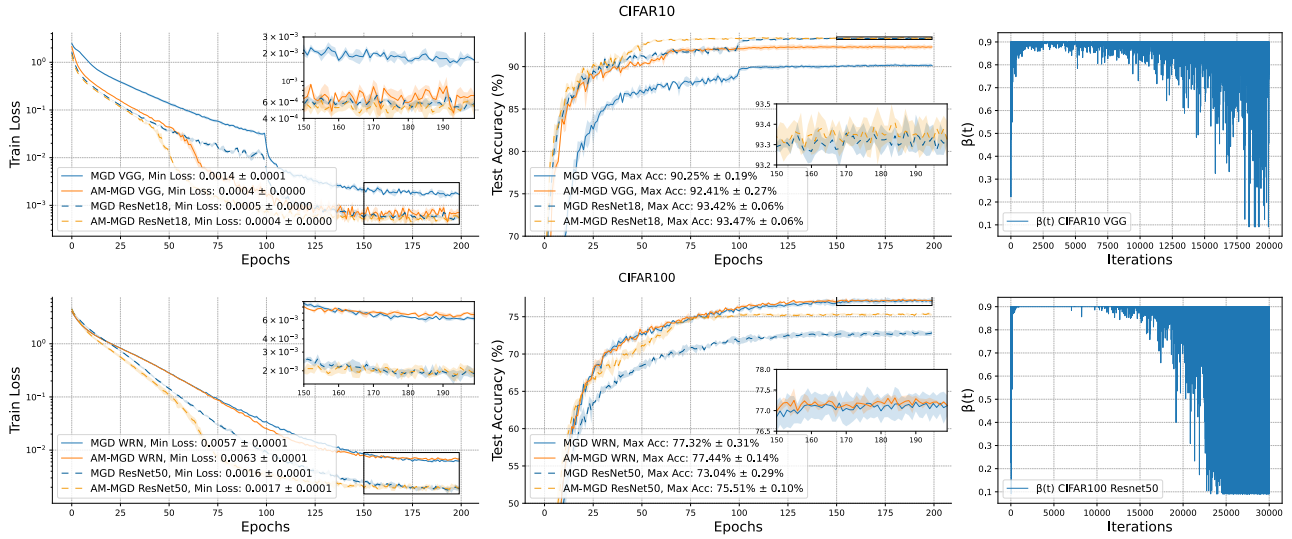


Figure 3: Train Loss, Test Accuracy and momentum parameter on the image classification experiments, (a) VGG19 and a ResNet18 on CIFAR10 (top), (b) ResNet50 and WideResNet on CIFAR100 (bottom)

architectures. For MGD, we set PyTorch's dampening parameter to 0.9 to ensure a fair comparison at identical learning rates. Finally, the learning-rate ablations in Figure 6 and in the Supplement Section F.2, show that the relative gains of AM-MGD persist across a broad range of step sizes. Full experimental details are provided in the Supplement Section E.

As shown in Figure 3, AM-MGD consistently outperforms fixed-momentum baselines, yielding both faster optimization and better generalization. This improvement is not only visible in the final accuracy, but also in the training dynamics: across the vision experiments, AM-MGD reaches strong performance substantially earlier, often requiring markedly fewer epochs to attain the same accuracy level as the fixed-momentum baseline. The trajectories of β_t further clarify how this improvement arises. Across the CIFAR experiments, we observe three recurring phases. Early in training, gradients are noisy and the EMA direction d_t is still poorly estimated, so the misalignment term $\|d_t - g_t\|_2^2$ is large and β_t remains small, effectively reducing the method closer to SGD and stabilizing the initial phase. As training progresses and gradients become more coherent, the misalignment shrinks, β_t increases, and momentum becomes reliable; this coincides with the main acceleration regime, where we observe the largest gains in convergence speed and learning-rate robustness. Finally, in interpolating settings such as CIFAR-10/100, once the training loss becomes very small, the gradient signal weakens while stochastic noise persists. In this regime, $\|d_t - g_t\|_2^2$ increases again and β_t exhibits frequent sharp drops toward zero, producing repeated restarts of the momentum buffer. This behaviour is qualitatively consistent with prior observations that momentum can become less beneficial near interpolation (Defazio, 2020). Larger-scale ImageNet experiments in the Supplement Section F.4 show the same overall pattern.

4.3 Pretraining Large Language Models

We pretrain LLaMA models (Touvron, 2023; Grattafiori, 2024) of varying scales on the English subset of C4 (Raffel, 2020). Rather than training to convergence, these experiments focus on early-phase optimization over 1k iterations under a fixed token budget. For each model we tune the AdamW baseline over sequence length, batch size, learning rate, and β_2, subject to the constraint $\texttt{batch\_size} \times \texttt{seq\_len} = 10^6$, corresponding to a total of 10^9 tokens over 1k steps. We then reuse the same hyperparameters for AM-AdamW, so any performance differences are attributable to momentum adaptation rather than additional tuning. Unlike the vision experiments, where the adaptive momentum coefficient is global, here we compute it separately for each layer to account for layer heterogeneity and to avoid having any single layer dominate the momentum adaptation of the entire model. Each model is trained under two regimes, with and without the standard linear warm-up schedule (Hägele, 2024), in order to study the effect of our method during the earliest phase of training.

The results demonstrate a clear and consistent advantage for AM-AdamW over the standard AdamW optimizer across all model scales. Most notably, even without warm-up, AM-AdamW compares to the perfor-

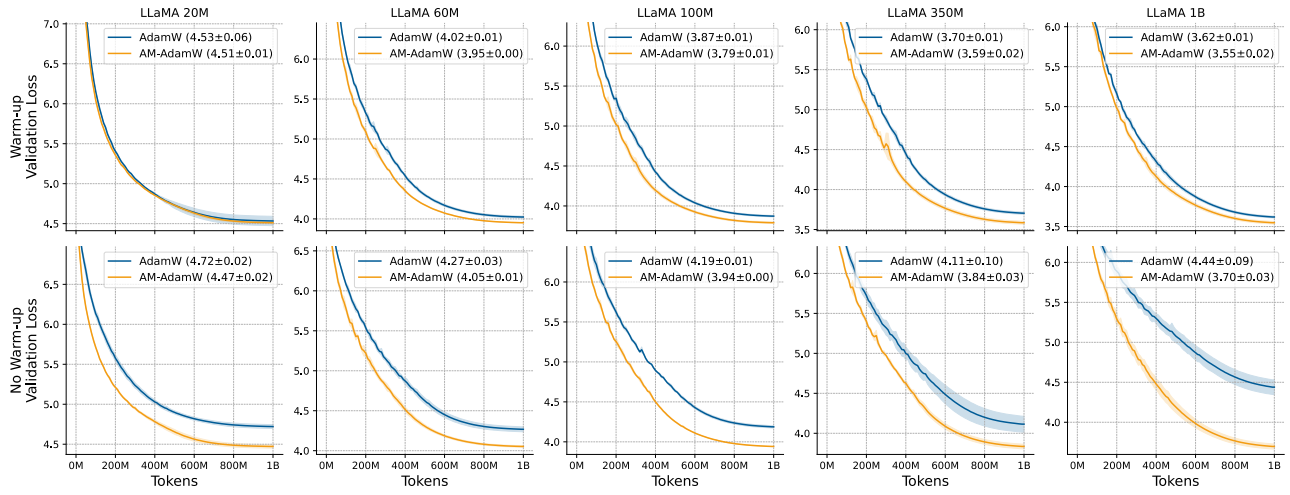


Figure 5: Validation loss curves for LLaMA pretraining on C4 across 5 different model scales.

mance of AdamW with warm-up. In contrast, AdamW struggles to train reliably without warm-up, particularly at larger scales. These findings highlight two key benefits of our adaptive memory approach: it stabilizes the early stages of training, leading to stronger and more consistent performance, and it shows promise as a hyperparameter-free alternative to warm-up.

4.4 Ablation Study

To further evaluate our method, we conducted the following three ablation studies on the image classification tasks: ResNet18/50 on CIFAR10/100.

Ablation on λ Figure 6a: We examine the robustness of our method with respect to the choice of the hyperparameter λ. In nearly all of our experiments, λ is set to 0.1. Notably, all values in the range $[0.01, 1]$ appear to yield comparable performance. Moreover, when λ becomes too large, the effective range of β_t is reduced due to the scaling factor $1 + \lambda$ (see Equation 13). As a result, β_t effectively reaches its upper bound, $\beta_{\max} = 0.9$. This explains why, for large values of λ, our method exhibits behaviour similar to that of a fixed momentum coefficient.

Ablation on batch size Figure 6b: The approximate cutting planes model of the loss used to estimate β_t at each iteration is constructed using stochastic gradients. Consequently, a key question arises: how does the stochasticity induced by the batch size influence the overall performance of our method? As expected, increasing the batch size while keeping the learning rate fixed tends to degrade generalization. However, smaller batch sizes, despite leading to more inaccurate models of the loss, do not negatively affect our method's relative performance gains.

Ablation on learning rate Figure 6c: We fix the hyperparameters to the values used in image classification experiments captured in Figure 3 and vary the learning rate. Interestingly, our adaptive memory scheme extends the range of admissible learning rates, enabling convergence even at higher values, unlike the static baseline, which fails to converge as fast with large learning rates.

4.5 What matters when adapting β

When observing the behaviour of β_t, we identify two regimes: a high-momentum phase, where β_t remains close to its maximum and carries long-term memory, and a low-momentum or "soft restart" phase, where β_t drops sharply to discard stale information. To better understand these two modes of operation, we explore two variants of adaptive memory: *clipping*, which enforces a lower bound on β_t, and *restarting*, which resets momentum when it falls below a threshold.

$$\beta_t^{\text{Clip}} = \max(\beta_{\text{AM}}, \beta_{\min}), \quad \beta_t^{\text{Reset}} = \begin{cases} 0.9, & \beta_{\text{AM}} \geq \theta, \\ 0.1, & \beta_{\text{AM}} < \theta \end{cases}$$

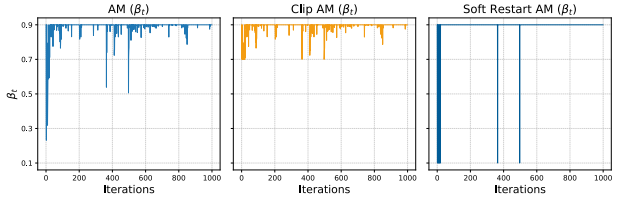


Figure 7: The three AM-derived policies.

We find that early restarts carry much of the benefit by rapidly adapting to unstable initial dynamics as we see in Table 1, however, these restarts alone do not explain

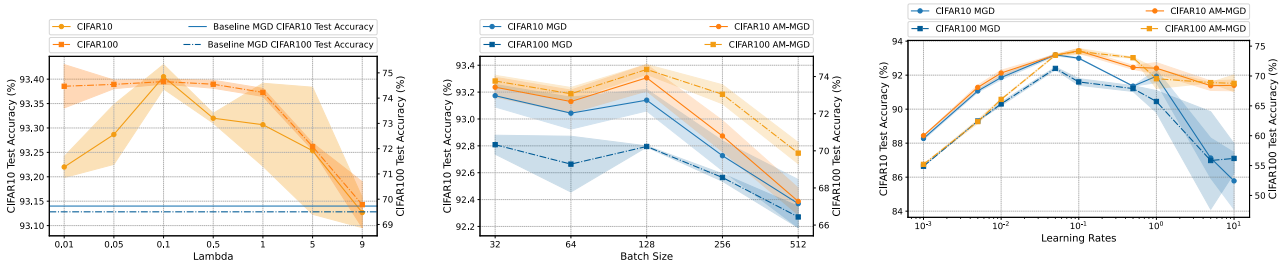


Figure 6: Final test accuracy for ResNet18 (CIFAR-10) and ResNet50 (CIFAR-100) under different (left) λ, (middle) batch size, and (right) learning rate.

all of the improvements, indicating that the ongoing adaptation of β_t throughout training is also essential.

	60M	**100M**
AdamW	4.02	3.87
AM-AdamW	3.95	3.79
AM-AdamW (Clip)	3.99	3.84
AM-AdamW(Restart)	3.97	3.82

Table 1: The three alternative AM optimizers, we set the restarting threshold $\theta = 0.7$.

4.6 Choice of $\beta_{\max}$

A natural question is how to set the ceiling $\beta_{\max}$ on the adaptive momentum coefficient. To keep the method as tuning-light as possible, in all main experiments we use the baseline-aligned heuristic $\beta_{\max} = \beta$, where β is the fixed momentum of the corresponding vanilla baseline (e.g., $\beta_1 = 0.9$ for AdamW/SGD). This provides a simple default recipe and isolates the effect of adaptivity, since gains over the baseline cannot be attributed to a different global momentum value. Intuitively, larger ceilings can help when gradients are coherent, but in static methods they also retain unreliable EMA memory for longer. AM is designed to mitigate this by down-weighting such memory, which helps explain why its advantage over the static baseline becomes larger at higher ceilings. In practice, $\beta_{\max} \in \{0.9, 0.95\}$ works best, while $\beta_{\max} = 0.99$, although still much better than the corresponding static baseline, is somewhat less reliable.

$\beta_{\max}$	60M		100M	
	AdamW	AM-AdamW	AdamW	AM-AdamW
0.95	4.387	**3.997**	4.217	**3.822**
0.99	6.083	**4.568**	6.133	**4.474**

Table 2: Sensitivity to the momentum ceiling $\beta_{\max}$ on LLaMA pretraining. Lower final validation loss is better.

4.7 Computational Requirements

In terms of computational cost, AM-MGD and AM-AdamW introduce minimal overhead compared to their non-adaptive counterparts. Specifically, each iteration of AM-MGD incurs an additional $3N$ FLOPs, and AM-AdamW adds $6N$ FLOPs, where N is the number of model parameters. For Transformer-based language models, the total FLOPs per iteration are approximately $6NB$ (Kaplan, 2020), where B is the number of tokens processed (batch size times sequence length). In our setup, this $1/B$ overhead translates into a 0.2% absolute time overhead. Moreover, the memory footprint of our methods is identical to that of the baselines, i.e., AM-MGD and AM-AdamW require no additional memory beyond what is already used by MGD and AdamW, respectively. Detailed wall-clock overhead can be found in the Supplement Section D.4.

5 DISCUSSION AND FUTURE WORK

We have presented Adaptive Memory (AM), a principled framework that endows momentum-based optimizers with a dynamic momentum coefficient. By casting each update as a proximal step on an approximate two-plane model of the loss, AM adapts to the local geometry of the loss landscape with limited additional hyperparameter overhead. AM integrates seamlessly with standard optimizers such as SGD and AdamW, incurs negligible overhead, and requires no changes to existing training pipelines. Empirically, AM delivers consistent gains across a spectrum of tasks, from convex benchmarks and vision models to large-scale LLM pretraining. In particular, AM-AdamW stabilizes the fragile early phase of LLM training and shows promise in eliminating the need for hand-tuned warm-up schedules, demonstrating robustness to large learning rates and simplifying large-scale workflows. Future work includes: (i) investigating memory in first-order methods and the impact of momentum restarts; (ii) advancing tuning-free momentum-based optimizers with dynamic and per-parameter momentum coefficients; and (iii) developing warm-up-free strategies for stable training.

Acknowledgements

The research presented in this paper was partially sponsored by the NSF CAREER Award #2041872.

References

Alex, Krizhevsky (2009). "Learning multiple layers of features from tiny images". In: *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*.

Asi, Hilal, Karan Chadha, Gary Cheng, and John C Duchi (2020). "Minibatch stochastic approximate proximal point methods". In: *Advances in neural information processing systems* 33, pp. 21958–21968.

Asi, Hilal and John C Duchi (2019). "Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity". In: *SIAM Journal on Optimization* 29.3, pp. 2257–2290.

Barré, Mathieu, Adrien Taylor, and Alexandre d'Aspremont (2020). "Complexity guarantees for Polyak steps with momentum". In: *Conference on learning theory*. PMLR, pp. 452–478.

Bottou, L. (1998). "Online Algorithms and Stochastic Approximations". In: *Online Learning and Neural Networks*. Cambridge University Press.

Chadha, Karan, Gary Cheng, and John Duchi (2022). "Accelerated, optimal and parallel: Some results on model-based stochastic optimization". In: *International Conference on Machine Learning*. PMLR, pp. 2811–2827.

Chang, Chih-Chung and Chih-Jen Lin (2011). "LIBSVM: A library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology* 2 (3). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 27:1–27:27.

Chen, John, Cameron Wolfe, Zhao Li, and Anastasios Kyrillidis (2022). "Demon: Improved Neural Network Training With Momentum Decay". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3958–3962. DOI: 10.1109/ICASSP43922.2022.9746839.

Chen, Xiangning, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le (2023). "Symbolic Discovery of Optimization Algorithms". In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine. URL: http://papers.nips.cc/paper%5C_files/paper/2023/hash/9a39b4925e35cf447ccba8757137d84f-Abstract-Conference.html.

Davis, Damek and Dmitriy Drusvyatskiy (2019). "Stochastic model-based minimization of weakly convex functions". In: *SIAM Journal on Optimization* 29.1, pp. 207–239.

Defazio, Aaron (2020). "Momentum via primal averaging: theoretical insights and learning rate schedules for non-convex optimization". In: *arXiv preprint arXiv:2010.00406*.

Defazio, Aaron and Konstantin Mishchenko (2023). "Learning-rate-free learning by d-adaptation". In: *International Conference on Machine Learning*. PMLR, pp. 7449–7479.

Deng, Qi and Wenzhi Gao (2021). "Minibatch and momentum model-based methods for stochastic weakly convex optimization". In: *Proceedings of the 35th International Conference on Neural Information Processing Systems*. NIPS '21. Red Hook, NY, USA: Curran Associates Inc. ISBN: 9781713845393.

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive subgradient methods for online learning and stochastic optimization." In: *Journal of machine learning research* 12.7.

Giselsson, Pontus and Stephen Boyd (2014). "Monotonicity and restart in fast gradient methods". In: *53rd IEEE Conference on Decision and Control*. IEEE, pp. 5058–5063.

Gitman, Igor, Hunter Lang, Pengchuan Zhang, and Lin Xiao (2019). "Understanding the role of momentum in stochastic gradient methods". In: *Advances in Neural Information Processing Systems* 32.

Goyal, Priya, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He (2017). "Accurate, large minibatch sgd: Training imagenet in 1 hour". In: *arXiv preprint arXiv:1706.02677*.

Grattafiori, Aaron, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, (2024). "The llama 3 herd of models". In: *arXiv preprint arXiv:2407.21783*.

Hägele, Alex, Elie Bakouch, Atli Kosson, Leandro Von Werra, Martin Jaggi, (2024). "Scaling laws and

compute-optimal training beyond fixed training durations". In: *Advances in Neural Information Processing Systems* 37, pp. 76232–76264.

Hager, William W and Hongchao Zhang (2006). "A survey of nonlinear conjugate gradient methods". In: *Pacific journal of Optimization* 2.1, pp. 35–58.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky (2012). "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent". In: *Cited on* 14.8, p. 2.

Hu, Chonghai, Weike Pan, and James Kwok (2009). "Accelerated gradient methods for stochastic optimization and online learning". In: *Advances in Neural Information Processing Systems* 22.

Ivgi, Maor, Oliver Hinder, and Yair Carmon (2023). "Dog is sgd's best friend: A parameter-free dynamic step size schedule". In: *International Conference on Machine Learning*. PMLR, pp. 14465–14499.

Jelassi, Samy and Yuanzhi Li (2022). "Towards understanding how momentum improves generalization in deep learning". In: *International Conference on Machine Learning*. PMLR, pp. 9965–10040.

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

Kalra, Dayal Singh and Maissam Barkeshli (2024). "Why warmup the learning rate? underlying mechanisms and improvements". In: *Advances in Neural Information Processing Systems* 37, pp. 111760–111801.

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

Kelley Jr, James E (1960). "The cutting-plane method for solving convex programs". In: *Journal of the society for Industrial and Applied Mathematics* 8.4, pp. 703–712.

Kim, Junhyung Lyle, Panos Toulis, and Anastasios Kyrillidis (2022). "Convergence and stability of the stochastic proximal point algorithm with momentum". In: *Learning for Dynamics and Control Conference*. PMLR, pp. 1034–1047.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980.

Kiwiel, Krzysztof C (2006). *Methods of descent for nondifferentiable optimization*. Vol. 1133. Springer.

Kiwiel, Krzysztof Czesław (1983). "An aggregate subgradient method for nonsmooth convex minimization". In: *Mathematical Programming* 27, pp. 320–341.

Kosson, Atli, Bettina Messmer, and Martin Jaggi (2024). "Analyzing & reducing the need for learning rate warmup in GPT training". In: *Advances in Neural Information Processing Systems* 37, pp. 2914–2942.

Liu, Hong, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma (2024). "Sophia: A Scalable Stochastic Second-order Optimizer for Language Model Pre-training". In: *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. URL: https://openreview.net/forum?id=3xHDeA8Noi.

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

Liu, Yanli, Yuan Gao, and Wotao Yin (2020). "An improved analysis of stochastic gradient descent with momentum". In: *Advances in Neural Information Processing Systems* 33, pp. 18261–18271.

Loizou, Nicolas and Peter Richtárik (2020). "Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods". In: *Computational Optimization and Applications* 77.3, pp. 653–710.

Loizou, Nicolas, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien (2021). "Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence". In: *International Conference on Artificial Intelligence and Statistics*. PMLR, pp. 1306–1314.

Loshchilov, Ilya and Frank Hutter (2017a). "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. URL: https://api.semanticscholar.org/CorpusID:53592270.

– (2017b). "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=Skq89Scxx.

Mai, Vien and Mikael Johansson (2020). "Convergence of a stochastic gradient method with momentum for non-smooth non-convex optimization". In: *International conference on machine learning*. PMLR, pp. 6630–6639.

Malitsky, Yura and Konstantin Mishchenko (13–18 Jul 2020). "Adaptive Gradient Descent without Descent". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 6702–6712. URL: https://proceedings.mlr.press/v119/malitsky20a.html.

Nesterov, Yurii (1983). "A method for solving the convex programming problem with convergence rate O (1/k2)". In: *Dokl akad nauk Sssr*. Vol. 269, p. 543.

O'donoghue, Brendan and Emmanuel Candes (2015). "Adaptive restart for accelerated gradient schemes". In: *Foundations of computational mathematics* 15, pp. 715–732.

Oikonomou, Dimitris and Nicolas Loizou (2025). "Stochastic Polyak Step-sizes and Momentum: Convergence Guarantees and Practical Performance". In: *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net. URL: https://openreview.net/forum?id=nuX2yPejiL.

Orvieto, Antonio, Simon Lacoste-Julien, and Nicolas Loizou (2022). "Dynamics of sgd with stochastic polyak stepsizes: Truly adaptive variants and convergence to exact solution". In: *Advances in Neural Information Processing Systems* 35, pp. 26943–26954.

Pagliardini, Matteo, Pierre Ablin, and David Grangier (2025). "The AdEMAMix Optimizer: Better, Faster, Older". In: *The Twelfth International Conference on Learning Representations, ICLR 2025*.

Polyak, Boris T (1964). "Some methods of speeding up the convergence of iteration methods". In: *Ussr computational mathematics and mathematical physics* 4.5, pp. 1–17.

– (1987). "Introduction to optimization". In.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu (2020). "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *Journal of machine learning research* 21.140, pp. 1–67.

Robbins, Herbert and Sutton Monro (1951). "A stochastic approximation method". In: *The annals of mathematical statistics*, pp. 400–407.

Rockafellar, R Tyrrell (1976). "Monotone operators and the proximal point algorithm". In: *SIAM journal on control and optimization* 14.5, pp. 877–898.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, (2015). "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115, pp. 211–252.

Ruszczyński, Andrzej (1987). "A linearization method for nonsmooth stochastic programming problems". In: *Mathematics of Operations Research* 12.1, pp. 32–49.

Saab Jr, Samer, Shashi Phoha, Minghui Zhu, and Asok Ray (2022). "An adaptive polyak heavy-ball method". In: *Machine Learning* 111.9, pp. 3245–3277.

Schaipp, Fabian, Ruben Ohana, Michael Eickenberg, Aaron Defazio, and Robert M. Gower (2024). "MoMo: Momentum Models for Adaptive Learning Rates". In: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. Ed. by Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp. Proceedings of Machine Learning Research. PMLR / OpenReview.net, pp. 43542–43570. URL: https://proceedings.mlr.press/v235/schaipp24a.html.

Sebbouh, Othmane, Robert M Gower, and Aaron Defazio (2021). "Almost sure convergence rates for stochastic gradient descent and stochastic heavy ball". In: *Conference on Learning Theory*. PMLR, pp. 3935–3971.

Shazeer, Noam and Mitchell Stern (2018). "Adafactor: Adaptive learning rates with sublinear memory cost".

In: *International Conference on Machine Learning*. PMLR, pp. 4596–4604.

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1409.1556.

Smith, Leslie N (2017). "Cyclical learning rates for training neural networks". In: *2017 IEEE winter conference on applications of computer vision (WACV)*. IEEE, pp. 464–472.

Sutskever, Ilya, James Martens, George Dahl, and Geoffrey Hinton (2013). "On the importance of initialization and momentum in deep learning". In: *International conference on machine learning*. PMLR, pp. 1139–1147.

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur'elien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample (2023). "LLaMA: Open and Efficient Foundation Language Models". In: *ArXiv* abs/2302.13971. URL: https://api.semanticscholar.org/CorpusID:257219404.

Tseng, Paul (1998). "An incremental gradient (-projection) method with momentum term and adaptive stepsize rule". In: *SIAM Journal on Optimization* 8.2, pp. 506–531.

Vyas, Nikhil, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade (2025). "Soap: Improving and stabilizing shampoo using adam". In: *The Twelfth International Conference on Learning Representations, ICLR 2025*.

Wang, Bao, Tan Nguyen, Tao Sun, Andrea L Bertozzi, Richard G Baraniuk, and Stanley J Osher (2022). "Scheduled restart momentum for accelerated stochastic gradient descent". In: *SIAM Journal on Imaging Sciences* 15.2, pp. 738–761.

Wang, Xiaoyu, Mikael Johansson, and Tong Zhang (2023). "Generalized Polyak step size for first order optimization with momentum". In: *International Conference on Machine Learning*. PMLR, pp. 35836–35863.

Wortsman, Mitchell, Peter J Liu, Lechao Xiao, Katie Everett, Alex Alemi, Ben Adlam, John D Co-Reyes, Izzeddin Gur, Abhishek Kumar, Roman Novak, (2024). "Small-scale proxies for large-scale transformer training instabilities". In: *The Twelfth International Conference on Learning Representations, ICLR 2024*.

Yan, Yan, Tianbao Yang, Zhe Li, Qihang Lin, and Yi Yang (2018). "A Unified Analysis of Stochastic Momentum Methods for Deep Learning". In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*. Ed. by Jérôme Lang. ijcai.org, pp. 2955–2961. DOI: 10.24963/IJCAI.2018/410. URL: https://doi.org/10.24963/ijcai.2018/410.

You, Yang, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh (2020). "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. URL: https://openreview.net/forum?id=Syx4wnEtvH.

Yuan, Huizhuo, Yifeng Liu, Shuang Wu, Xun Zhou, and Quanquan Gu (2025). "MARS: Unleashing the Power of Variance Reduction for Training Large Models". In: *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. Ed. by Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu. Proceedings of Machine Learning Research. PMLR / OpenReview.net. URL: https://proceedings.mlr.press/v267/yuan25f.html.

Zagoruyko, Sergey and Nikos Komodakis (2016). "Wide Residual Networks". In: *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. Ed. by Richard C. Wilson, Edwin R. Hancock, and William A. P. Smith. BMVA Press. URL: https://bmva-archive.org.uk/bmvc/2016/papers/paper087/index.html.

Zhuang, Zhenxun, Mingrui Liu, Ashok Cutkosky, and Francesco Orabona (2022). "Understanding adamw through proximal methods and scale-freeness". In: *Transactions on machine learning research*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Yes
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Yes
   (b) Complete proofs of all theoretical results. Yes
   (c) Clear explanations of any assumptions. Yes
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes
   (b) The license information of the assets, if applicable. Not Applicable
   (c) New assets either in the supplemental material or as a URL, if applicable. Yes
   (d) Information about consent from data providers/curators. Not Applicable
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

Supplement

A DERIVATION OF ADAPTIVE MEMORY MOMENTUM

We use the cutting-plane model

$$f_t^m(x) = \max\left\{f(x_t) + g_t^\top(x - x_t),\ \hat{f}(x_t) + d_t^\top(x - x_t)\right\}, \qquad g_t := \nabla f(x_t). \tag{20}$$

Let P_t be a diagonal preconditioner, and define

$$\tilde{P}_t := (I + \lambda P_t)P_t. \tag{21}$$

The update is

$$x_{t+1} \in \arg\min_x \left\{f_t^m(x) + \frac{1}{2\eta}\|x - x_t\|_{\tilde{P}_t}^2 + \lambda\langle d_t, x - x_t\rangle_{P_t} + \frac{\mu}{2}\|x\|_{\tilde{P}_t}^2\right\}. \tag{22}$$

Define $\tilde{x} := x - x_t$. Then the problem becomes

$$\min_{\tilde{x},\zeta}\left\{\zeta + \frac{1}{2\eta}\|\tilde{x}\|_{\tilde{P}_t}^2 + \lambda\langle d_t, \tilde{x}\rangle_{P_t} + \frac{\mu}{2}\|x_t + \tilde{x}\|_{\tilde{P}_t}^2\right\} \tag{23}$$

subject to

$$\zeta \geq f(x_t) + g_t^\top\tilde{x}, \qquad \zeta \geq \hat{f}(x_t) + d_t^\top\tilde{x}. \tag{24}$$

Introducing dual variables $a_1, a_2 \geq 0$, the Lagrangian is

$$\begin{aligned}\mathcal{L}(\tilde{x}, \zeta, a_1, a_2) = \zeta + \frac{1}{2\eta}\|\tilde{x}\|_{\tilde{P}_t}^2 + \lambda\langle d_t, \tilde{x}\rangle_{P_t} + \frac{\mu}{2}\|x_t + \tilde{x}\|_{\tilde{P}_t}^2 \\ - a_1\big(\zeta - f(x_t) - g_t^\top\tilde{x}\big) - a_2\big(\zeta - \hat{f}(x_t) - d_t^\top\tilde{x}\big).\end{aligned} \tag{25}$$

The KKT conditions are

$$\frac{\partial\mathcal{L}}{\partial\tilde{x}} = 0 \iff \frac{1}{\eta}\tilde{P}_t\tilde{x} + \lambda P_t d_t + \mu\tilde{P}_t(\tilde{x} + x_t) + a_1 g_t + a_2 d_t = 0, \tag{26}$$

$$\frac{\partial\mathcal{L}}{\partial\zeta} = 0 \iff a_1 + a_2 = 1. \tag{27}$$

Solving (26) for $\tilde{x}$ yields

$$\tilde{x} = -\frac{\eta}{1 + \mu\eta}\,\tilde{P}_t^{-1}\Big((\lambda P_t + a_2 I)d_t + \mu\tilde{P}_t x_t + a_1 g_t\Big). \tag{28}$$

Therefore

$$x_{t+1} = \frac{1}{1 + \mu\eta}\left(x_t - \eta\tilde{P}_t^{-1}\big(a_1 g_t + a_2 d_t + \lambda P_t d_t\big)\right). \tag{29}$$

Substituting (28) back into the objective gives the dual problem

$$\max_{\substack{a_1,a_2\geq 0\\ a_1+a_2=1}}\left\{-\frac{\eta}{2(1+\mu\eta)}\left\|(\lambda P_t + a_2 I)d_t + \mu\tilde{P}_t x_t + a_1 g_t\right\|_{\tilde{P}_t^{-1}}^2 + \frac{\mu}{2}\|x_t\|_{\tilde{P}_t}^2 + a_1 f(x_t) + a_2\hat{f}(x_t)\right\}. \tag{30}$$

Setting $a_2 = \beta$ and $a_1 = 1 - \beta$, we obtain

$$\max_{0\le\beta\le 1}\left\{-\frac{\eta}{2(1+\mu\eta)}\left\|\mu\tilde{P}_t x_t + g_t + \lambda P_t d_t + \beta(d_t - g_t)\right\|^2_{\tilde{P}_t^{-1}} + \frac{\mu}{2}\|x_t\|^2_{\tilde{P}_t} + (1-\beta)f(x_t) + \beta\hat{f}(x_t)\right\}. \tag{31}$$

This is a concave quadratic program. Differentiating with respect to β, setting the derivative to zero, and projecting onto $[0,1]$ yields

$$\beta_t^* = \mathrm{Clip}_{[0,1]}\left(\frac{\frac{(1+\mu\eta)\left(\hat{f}(x_t)-f(x_t)\right)}{\eta} - \langle d_t - g_t,\ g_t + \lambda P_t d_t\rangle_{\tilde{P}_t^{-1}} - \mu\langle x_t,\ d_t - g_t\rangle}{\|d_t - g_t\|^2_{\tilde{P}_t^{-1}}}\right). \tag{32}$$

- Setting $\mu = 0$ and $P_t = I$ produces Momentum Gradient Descent
- Setting $\mu = 0$ and P_t equal to the Adam Preconditioner yields Adam.
- Setting $\mu > 0$ and P_t equal to the Adam Preconditioner yields AdamW.

In addition we can interpret Equation 29 in two different ways, which are equivalent over one step but differ over what we store in memory.

$$(I): \begin{cases} d_{t+1} = (\lambda P_t + I)^{-1}\left((1-\beta_t)\, g_t + (\lambda P_t + \beta_t I)\, d_t\right) \\ x_{t+1} = \frac{1}{1+\mu\eta}\left(x_t - \eta P_t^{-1} d_{t+1}\right) \end{cases}$$

or

$$(II): \begin{cases} d_{t+1} = (1-\beta_t)\, g_t + \beta_t d_t \\ x_{t+1} = \frac{1}{1+\mu\eta}\left(x_t - \eta P_t^{-1}(\lambda P_t + I)^{-1}\left((1-\beta_t)\, g_t + (\lambda P_t + \beta_t I)\, d_t\right)\right) \end{cases}$$

Of the two options, the first was found to perform slightly better. Additionally, it aligns with existing momentum-based methods. In contrast, the second option requires extra memory to store both d_{t+1}, which represents the momentum for the next iteration, and d_t, which is needed for the parameter update.

B DISCUSSION ON THE OVERESTIMATING MOMENTUM PLANE

In this section we give a simple motivating example showing that, in a non-overshooting momentum regime, a positive *overestimation term* can be aligned with the one-step optimal choice of momentum. We focus on heavy-ball (HB) momentum on a diagonal strongly convex quadratic, where the dynamics decouple across coordinates and can be analyzed exactly.

B.1 Non-overshooting heavy-ball dynamics

We consider the quadratic

$$f(x) = \tfrac{1}{2}x^\top A x, \qquad A = \mathrm{diag}(a_1, \dots, a_d), \qquad 0 < a_d \le \cdots \le a_1 = L,$$

whose minimizer is $x^* = 0$. The heavy-ball iteration is

$$d_{t+1} = \beta d_t + (1-\beta)Ax_t, \qquad x_{t+1} = x_t - \eta d_{t+1},$$

with $\beta \in [0,1)$, $\eta > 0$, and initialization

$$d_0 = g_0 = Ax_0.$$

Definition B.1 (Coordinate-wise monotonic decrease)**.** We say that the iterates satisfy the *coordinate-wise monotonic decrease condition* if for every $t \geq 0$ and every coordinate i,

$$|x_{t+1}^i - x^{*i}| \leq |x_t^i - x^{*i}| \quad \text{and} \quad \operatorname{sign}(x_{t+1}^i - x^{*i}) = \operatorname{sign}(x_t^i - x^{*i}).$$

Since $x^* = 0$ here, this is equivalent to

$$|x_{t+1}^i| \leq |x_t^i| \quad \text{and} \quad \operatorname{sign}(x_{t+1}^i) = \operatorname{sign}(x_t^i).$$

For the diagonal quadratic above, plain gradient descent corresponds to $\beta = 0$ and satisfies this condition whenever $\eta \leq 1/L$. The next proposition shows that heavy-ball also satisfies it in the overdamped regime.

Proposition B.2 (No-overshoot regime of HB on a diagonal quadratic)**.** *Assume*

$$0 < \beta < 1, \qquad \eta L < \frac{1-\sqrt{\beta}}{1+\sqrt{\beta}}.$$

Then for every coordinate i with $x_0^i \neq 0$ and every $t \geq 0$,

$$\operatorname{sign}(x_t^i) = \operatorname{sign}(x_0^i), \qquad \operatorname{sign}(g_t^i) = \operatorname{sign}(x_0^i), \qquad \operatorname{sign}(d_t^i) = \operatorname{sign}(x_0^i),$$

and

$$|x_{t+1}^i| < |x_t^i|.$$

If $x_0^i = 0$, then $x_t^i = d_t^i = g_t^i = 0$ for all t.

Proof. Fix a coordinate i and write $a := a_i > 0$. If $x_0^i = 0$, then $d_0^i = g_0^i = 0$, and the scalar recursion stays identically zero. We therefore assume $x_0^i \neq 0$. By symmetry it is enough to consider the case $x_0^i > 0$; the case $x_0^i < 0$ follows by multiplying the scalar dynamics by -1.

The scalar HB recursion is

$$d_{t+1} = \beta d_t + (1-\beta) a x_t, \qquad x_{t+1} = x_t - \eta d_{t+1}.$$

Define $z_t = (d_t, x_t)^\top$. Then

$$z_{t+1} = M z_t, \qquad M = \begin{pmatrix} \beta & (1-\beta)a \\ -\eta\beta & 1-\eta(1-\beta)a \end{pmatrix}.$$

The characteristic polynomial of M is

$$r^2 - \tau r + \beta = 0, \qquad \tau = 1 + \beta - \eta(1-\beta)a.$$

Since $a \leq L$ and

$$\eta L < \frac{1-\sqrt{\beta}}{1+\sqrt{\beta}},$$

we also have

$$\eta a < \frac{1-\sqrt{\beta}}{1+\sqrt{\beta}}.$$

This is exactly the overdamped condition, and it implies that the two roots r_1, r_2 satisfy

$$0 < r_2 < r_1 < 1.$$

For each root r_j, an eigenvector is

$$u_j = \begin{pmatrix} -\theta_j \\ 1 \end{pmatrix}, \qquad \theta_j = \frac{(1-\beta)a}{\beta - r_j}, \qquad j = 1, 2.$$

Hence

$$x_t = c_1 r_1^t + c_2 r_2^t.$$

Using the initialization

$$(d_0, x_0)^\top = (a x_0, x_0)^\top = c_1 u_1 + c_2 u_2,$$

we obtain

$$c_1 = -\frac{x_0(a+\theta_2)}{\theta_1-\theta_2}, \qquad c_2 = \frac{x_0(a+\theta_1)}{\theta_1-\theta_2}.$$

Because $0 < r_2 < r_1 < 1$ and $\beta = r_1 r_2$, we have

$$\beta - r_1 < \beta - r_2 < 0,$$

so

$$\theta_2 < \theta_1 < 0.$$

Moreover,

$$a + \theta_j = a + \frac{(1-\beta)a}{\beta - r_j} = \frac{(1-r_j)a}{\beta - r_j} < 0, \qquad j = 1, 2.$$

Since $x_0 > 0$, this shows that $c_1 > 0$ and $c_2 < 0$. Also,

$$\frac{|c_2|}{c_1} = \frac{-(a+\theta_1)}{-(a+\theta_2)} < 1,$$

because $\theta_1 > \theta_2$ implies $a + \theta_1 > a + \theta_2$, and both quantities are negative. Therefore

$$c_1 > |c_2|.$$

It follows that for every $t \geq 0$,

$$x_t = r_1^t\big(c_1 + c_2(r_2/r_1)^t\big) = r_1^t\big(c_1 - |c_2|(r_2/r_1)^t\big) \geq r_1^t(c_1 - |c_2|) = r_1^t x_0 > 0.$$

Thus $x_t > 0$ for all t, so the sign of x_t never changes.

Since $g_t = a x_t$, we also have $g_t > 0$ for all t. Finally, because $d_0 = a x_0 > 0$ and

$$d_{t+1} = \beta d_t + (1-\beta) g_t$$

is a convex combination of two positive quantities, it follows by induction that $d_t > 0$ for all t. Therefore

$$x_{t+1} = x_t - \eta d_{t+1} < x_t.$$

Combined with $x_{t+1} > 0$, this yields

$$0 < x_{t+1} < x_t,$$

which is exactly the claimed coordinate-wise monotonic decrease condition. The case $x_0^i < 0$ follows identically by symmetry. □

B.2 Optimal one-step momentum

A standard way to motivate adaptive step sizes is to choose the learning rate η_t by minimizing the one-step distance to the optimum,

$$J(\eta_t) = \|x_{t+1}(\eta_t) - x^*\|^2,$$

which leads to the classical Polyak step size after eliminating the unknown inner product by convexity. One can do the same with the momentum parameter.

Proposition B.3 (Optimal one-step momentum parameter)**.** *Fix an iterate x_t, a direction estimate d_t, and the gradient $g_t = \nabla f(x_t)$. For $\beta_t \in [0,1]$, define*

$$d_{t+1}(\beta_t) = \beta_t d_t + (1-\beta_t) g_t, \qquad x_{t+1}(\beta_t) = x_t - \eta d_{t+1}(\beta_t).$$

Then the choice of β_t minimizing the one-step distance to the optimum,

$$\beta_t^* \in \arg\min_{\beta \in [0,1]} \|x_{t+1}(\beta) - x^*\|^2,$$

is

$$\beta_t^* = \mathrm{Clip}_{[0,1]}\left(\frac{\frac{(d_t - g_t)^\top (x_t - x^*)}{\eta} - d_t^\top g_t + \|g_t\|^2}{\|d_t - g_t\|^2}\right).$$

Proof. Define

$$J(\beta_t) = \|x_{t+1}(\beta_t) - x^*\|^2 = \left\|(x_t - x^*) - \eta\big[\beta_t d_t + (1-\beta_t) g_t\big]\right\|^2.$$

Set

$$u = (x_t - x^*) - \eta g_t, \qquad v = d_t - g_t.$$

Then

$$x_{t+1}(\beta_t) - x^* = u - \eta\beta_t v,$$

and therefore

$$J(\beta_t) = \|u\|^2 - 2\eta\beta_t\, u^\top v + \eta^2\beta_t^2\|v\|^2.$$

Differentiating and setting the derivative to zero gives

$$\beta_t = \frac{u^\top v}{\eta\|v\|^2}.$$

Substituting back u and v yields

$$\beta_t = \frac{\big((x_t - x^*) - \eta g_t\big)^\top (d_t - g_t)}{\eta\|d_t - g_t\|^2} = \frac{\frac{(d_t - g_t)^\top (x_t - x^*)}{\eta} - d_t^\top g_t + \|g_t\|^2}{\|d_t - g_t\|^2}.$$

Projecting onto $[0,1]$ gives the stated formula. □

The only term in Proposition B.3 that depends explicitly on the unknown optimum is

$$(d_t - g_t)^\top (x_t - x^*).$$

We now show that in the non-overshooting quadratic regime above, this term is coordinate-wise nonnegative.

Lemma B.4. *Under the assumptions of Proposition B.2, for every $t \geq 0$ and every coordinate i,*

$$(d_t^i - g_t^i) x_t^i \geq 0.$$

Consequently,

$$(d_t - g_t)^\top (x_t - x^*) = (d_t - g_t)^\top x_t \geq 0.$$

Proof. We argue by induction on t.

For $t = 0$, since $d_0^i = g_0^i = a_i x_0^i$,

$$(d_0^i - g_0^i) x_0^i = 0.$$

Now assume $(d_t^i - g_t^i) x_t^i \geq 0$. By Proposition B.2, x_t^i and x_{t+1}^i have the same sign, and $g_t^i = a_i x_t^i$ has that sign as well. Also,

$$d_{t+1}^i = \beta d_t^i + (1-\beta) g_t^i, \qquad g_{t+1}^i = a_i x_{t+1}^i.$$

A direct calculation gives

$$d_{t+1}^i - g_{t+1}^i = \beta(1 + \eta a_i)(d_t^i - g_t^i) + \eta a_i g_t^i.$$

By the induction hypothesis, $d_t^i - g_t^i$ has the same sign as x_t^i, and g_t^i also has the same sign as x_t^i. Since all coefficients are nonnegative, the right-hand side has the same sign as x_t^i, hence also the same sign as x_{t+1}^i. Therefore

$$(d_{t+1}^i - g_{t+1}^i) x_{t+1}^i \geq 0.$$

This proves the coordinate-wise claim. Summing over i and using $x^* = 0$ gives

$$(d_t - g_t)^\top (x_t - x^*) = \sum_{i=1}^{d} (d_t^i - g_t^i) x_t^i \geq 0.$$

□

Lemma B.4 shows that, in this non-overshooting quadratic regime, the one-step optimal momentum parameter contains a nonnegative contribution from the term

$$\frac{(d_t - g_t)^\top (x_t - x^*)}{\eta}.$$

This provides a concrete setting in which replacing the unknown optimum-dependent quantity by a positive surrogate, equivalently, using an *overestimating* model term, is directionally consistent with improving one-step progress.

C PROOFS OF SECTION 3.5

We provide convergence guarantees for our proposed method in the standard finite-sum setting:

$$\min_{x \in \mathbb{R}^d} f(x) \;=\; \frac{1}{n} \sum_{i=1}^{n} f_{S_i}(x),$$

where each f_{S_i} denotes the loss associated with a mini-batch S_i, and the goal is to minimize the average loss f. We proceed under the following assumptions.

Assumption C.1 (Smoothness)**.** The function f is L-smooth if, for all $x, y \in \mathbb{R}^d$,

$$f(y) \le f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2} \|y - x\|^2. \tag{33}$$

Assumption C.2 (Bounded stochastic gradient norm)**.** There exists $G > 0$ such that, for all x,

$$\mathbb{E}_S \left[\|\nabla f_S(x)\|^2 \right] \le G^2. \tag{34}$$

Algorithm. We prove convergence for a simplified version of AM-SGD-M with $\lambda = 0$. Let $\eta > 0$ be the step size, let $0 \le \beta_t \le \beta_{\max} < 1$, and initialize $d_0 = 0$. At iteration t, let

$$g_t := \nabla f_{S_t}(x_t),$$

where S_t denotes the sampled mini-batch. The updates are

$$\begin{aligned} \beta_t &= \mathrm{Clip}_{[0,\beta_{\max}]} \left(\frac{\|g_t\|^2}{\|d_t - g_t\|^2} \right), \\ d_{t+1} &= \beta_t d_t + (1 - \beta_t) g_t, \\ x_{t+1} &= x_t - \eta d_{t+1}. \end{aligned} \tag{35}$$

When $d_t = g_t$, the denominator vanishes; in that case we define $\beta_t := 0$ by convention, noting that then $d_{t+1} = g_t$ regardless of the value of β_t.

We denote by x^* an optimal point, i.e. $x^* \in \arg\min_x f(x)$.

Lemma C.3. *Consider the iterates generated by (35). Assume that*

$$\mathbb{E}\left[\|g_t\|^2 \right] \le G^2.$$

Then, for every t,

$$\mathbb{E}\left[\|d_{t+1}\|^2 \right] \le 4G^2.$$

Proof. We rewrite the momentum update as

$$d_{t+1} = g_t + \beta_t (d_t - g_t).$$

Using Young's inequality, for any $\rho > 0$,

$$\|a + b\|^2 \le (1 + \rho) \|a\|^2 + \left(1 + \frac{1}{\rho} \right) \|b\|^2.$$

Applying this with

$$a = g_t, \qquad b = \beta_t(d_t - g_t),$$

gives

$$\|d_{t+1}\|^2 \le (1+\rho)\|g_t\|^2 + \left(1 + \frac{1}{\rho}\right)\beta_t^2\|d_t - g_t\|^2.$$

By the definition of β_t,

$$\beta_t \le \frac{\|g_t\|^2}{\|d_t - g_t\|^2}$$

whenever $d_t \neq g_t$, and trivially also when $d_t = g_t$ under the convention above. Hence

$$\beta_t\|d_t - g_t\|^2 \le \|g_t\|^2.$$

Since $0 \le \beta_t \le \beta_{\max} < 1$, we also have $\beta_t^2 \le \beta_t$, so

$$\beta_t^2\|d_t - g_t\|^2 \le \beta_t\|d_t - g_t\|^2 \le \|g_t\|^2.$$

Substituting this into the previous bound yields

$$\|d_{t+1}\|^2 \le \left(2 + \rho + \frac{1}{\rho}\right)\|g_t\|^2.$$

Choosing $\rho = 1$, we obtain

$$\|d_{t+1}\|^2 \le 4\|g_t\|^2.$$

Taking expectation and using $\mathbb{E}[\|g_t\|^2] \le G^2$ gives

$$\mathbb{E}\big[\|d_{t+1}\|^2\big] \le 4G^2.$$

□

The following technical lemmas provide upper bounds on the cross terms that arise when expanding the momentum recursion. Both results rely on the definition of β_t to relate the correction term $d_t - g_t$ to the stochastic gradient itself, and on Young's inequality to separate mixed inner products.

Lemma C.4. *Consider the iterates generated by (35). For any $a > 0$,*

$$-2\eta\beta_t\langle x_t - x^*,\, d_t - g_t\rangle \;\le\; \beta_{\max}a\|x_t - x^*\|^2 + \frac{\eta^2}{a}\|g_t\|^2,$$

where $g_t := \nabla f_{S_t}(x_t)$.

Proof. By Cauchy–Schwarz,

$$-2\eta\beta_t\langle x_t - x^*,\, d_t - g_t\rangle \;\le\; 2\eta\beta_t\|x_t - x^*\|\,\|d_t - g_t\|.$$

Applying Young's inequality $2uv \le au^2 + \frac{1}{a}v^2$ with

$$u = \sqrt{\beta_t}\,\|x_t - x^*\|, \qquad v = \eta\sqrt{\beta_t}\,\|d_t - g_t\|,$$

gives

$$2\eta\beta_t\|x_t - x^*\|\,\|d_t - g_t\| \;\le\; \beta_t a\|x_t - x^*\|^2 + \frac{\eta^2\beta_t}{a}\|d_t - g_t\|^2.$$

Since $\beta_t \le \beta_{\max}$, the first term satisfies

$$\beta_t a\|x_t - x^*\|^2 \;\le\; \beta_{\max}a\|x_t - x^*\|^2.$$

Moreover, by the definition of β_t,

$$\beta_t\|d_t - g_t\|^2 \le \|g_t\|^2,$$

with the inequality being trivial when $d_t = g_t$. Therefore,

$$\frac{\eta^2\beta_t}{a}\|d_t - g_t\|^2 \;\le\; \frac{\eta^2}{a}\|g_t\|^2.$$

Combining the two bounds yields

$$-2\eta\beta_t\langle x_t - x^*,\, d_t - g_t\rangle \;\le\; \beta_{\max} a\|x_t - x^*\|^2 + \frac{\eta^2}{a}\|g_t\|^2.$$

□

Lemma C.5. *Consider the iterates generated by (35). For any $a_t > 0$,*

$$-\eta\beta_t\langle\nabla f(x_t),\, d_t - g_t\rangle \;\le\; \frac{\beta_{\max}}{2a_t}\|\nabla f(x_t)\|^2 + \frac{a_t\eta^2}{2}\|g_t\|^2,$$

where $g_t := \nabla f_{S_t}(x_t)$.

Proof. By Cauchy–Schwarz,

$$-\eta\beta_t\langle\nabla f(x_t),\, d_t - g_t\rangle \;\le\; \eta\beta_t\|\nabla f(x_t)\|\,\|d_t - g_t\|.$$

Applying Young's inequality $uv \le \frac{u^2}{2a_t} + \frac{a_t}{2}v^2$ with

$$u = \sqrt{\beta_t}\,\|\nabla f(x_t)\|, \qquad v = \eta\sqrt{\beta_t}\,\|d_t - g_t\|,$$

gives

$$\eta\beta_t\|\nabla f(x_t)\|\,\|d_t - g_t\| \;\le\; \frac{\beta_t}{2a_t}\|\nabla f(x_t)\|^2 + \frac{a_t\eta^2\beta_t}{2}\|d_t - g_t\|^2.$$

Using $\beta_t \le \beta_{\max}$ for the first term and

$$\beta_t\|d_t - g_t\|^2 \le \|g_t\|^2$$

for the second term, we obtain

$$-\eta\beta_t\langle\nabla f(x_t),\, d_t - g_t\rangle \;\le\; \frac{\beta_{\max}}{2a_t}\|\nabla f(x_t)\|^2 + \frac{a_t\eta^2}{2}\|g_t\|^2.$$

□

Theorem C.6 (Convex convergence)**.** *Assume that each stochastic component f_S is convex, that the stochastic gradient is unbiased,*

$$\mathbb{E}[\,\nabla f_{S_t}(x_t) \mid x_t\,] = \nabla f(x_t),$$

and that Assumption 34 holds. Let $x^* \in \arg\min_x f(x)$. Consider the iterates generated by (35) with

$$\eta = \frac{1}{\sqrt{T}}, \qquad \beta_{\max} = \frac{1}{T}.$$

Define the weights

$$a_t := \left(1 + \frac{1}{T}\right)^{-(t+1)}, \qquad t = 0, \ldots, T-1,$$

and the weighted average iterate

$$x_T^{(a)} := \frac{\sum_{t=0}^{T-1} a_t x_t}{\sum_{t=0}^{T-1} a_t}.$$

Then

$$\mathbb{E}\big[f(x_T^{(a)}) - f(x^*)\big] \le \frac{1}{2\sqrt{T}}\left(2\|x_0 - x^*\|^2 + 5G^2\right).$$

Proof. Let

$$g_t := \nabla f_{S_t}(x_t).$$

From the update $x_{t+1} = x_t - \eta d_{t+1}$, we have

$$\begin{aligned}\|x_{t+1} - x^*\|^2 &= \|x_t - x^*\|^2 - 2\eta\langle x_t - x^*,\, d_{t+1}\rangle + \eta^2\|d_{t+1}\|^2 \\ &= \|x_t - x^*\|^2 - 2\eta\langle x_t - x^*,\, g_t\rangle - 2\eta\beta_t\langle x_t - x^*,\, d_t - g_t\rangle + \eta^2\|d_{t+1}\|^2.\end{aligned}$$

By convexity of f_{S_t},

$$\langle x_t - x^*,\, g_t\rangle \geq f_{S_t}(x_t) - f_{S_t}(x^*).$$

Applying Lemma C.4 with $a = 1$,

$$-2\eta\beta_t\langle x_t - x^*,\, d_t - g_t\rangle \leq \beta_{\max}\|x_t - x^*\|^2 + \eta^2\|g_t\|^2.$$

Therefore,

$$\|x_{t+1} - x^*\|^2 \leq (1 + \beta_{\max})\|x_t - x^*\|^2 - 2\eta\big(f_{S_t}(x_t) - f_{S_t}(x^*)\big) + \eta^2\|g_t\|^2 + \eta^2\|d_{t+1}\|^2.$$

Taking expectation, using unbiasedness, Assumption 34, and Lemma C.3, we obtain

$$\mathbb{E}\big[\|x_{t+1} - x^*\|^2\big] \leq (1 + \beta_{\max})\mathbb{E}\big[\|x_t - x^*\|^2\big] - 2\eta\,\mathbb{E}\big[f(x_t) - f(x^*)\big] + 5\eta^2G^2.$$

Now define the weights recursively by

$$a_{t-1} = (1 + \beta_{\max})a_t, \qquad a_{-1} = 1.$$

Since $\beta_{\max} = 1/T$, this yields

$$a_t = \left(1 + \frac{1}{T}\right)^{-(t+1)}.$$

Multiplying the previous inequality by a_t gives

$$a_t\,\mathbb{E}\big[\|x_{t+1} - x^*\|^2\big] \leq a_{t-1}\,\mathbb{E}\big[\|x_t - x^*\|^2\big] - 2\eta a_t\,\mathbb{E}\big[f(x_t) - f(x^*)\big] + 5\eta^2G^2a_t.$$

Rearranging and summing from $t = 0$ to $T - 1$,

$$2\eta\sum_{t=0}^{T-1} a_t\,\mathbb{E}\big[f(x_t) - f(x^*)\big] \leq \|x_0 - x^*\|^2 - a_{T-1}\mathbb{E}\big[\|x_T - x^*\|^2\big] + 5\eta^2G^2\sum_{t=0}^{T-1} a_t.$$

Dropping the nonnegative term $a_{T-1}\mathbb{E}[\|x_T - x^*\|^2]$, we get

$$2\eta\sum_{t=0}^{T-1} a_t\,\mathbb{E}\big[f(x_t) - f(x^*)\big] \leq \|x_0 - x^*\|^2 + 5\eta^2G^2\sum_{t=0}^{T-1} a_t.$$

Since f is convex,

$$f(x_T^{(a)}) \leq \sum_{t=0}^{T-1} \frac{a_t}{\sum_{s=0}^{T-1} a_s} f(x_t).$$

Taking expectation and subtracting $f(x^*)$,

$$\mathbb{E}\big[f(x_T^{(a)}) - f(x^*)\big] \leq \frac{\sum_{t=0}^{T-1} a_t\,\mathbb{E}[f(x_t) - f(x^*)]}{\sum_{t=0}^{T-1} a_t}.$$

Combining with the previous bound yields

$$\mathbb{E}\big[f(x_T^{(a)}) - f(x^*)\big] \leq \frac{\|x_0 - x^*\|^2}{2\eta\sum_{t=0}^{T-1} a_t} + \frac{5}{2}\eta G^2.$$

It remains to lower bound $\sum_{t=0}^{T-1} a_t$. Since

$$a_t = \left(1 + \frac{1}{T}\right)^{-(t+1)},$$

we have

$$\sum_{t=0}^{T-1} a_t = \frac{1 - (1+\frac{1}{T})^{-T}}{\frac{1}{T} \cdot (1+\frac{1}{T})^{-1}} \cdot \frac{1}{T} = T\left(1 - \left(1 + \frac{1}{T}\right)^{-T}\right).$$

Since $(1+1/T)^T$ is increasing in T and equals 2 at $T=1$, we have $(1+1/T)^{-T} \le 1/2$ for all $T \ge 1$, and therefore

$$\sum_{t=0}^{T-1} a_t \;\ge\; \frac{T}{2}.$$

Using $\eta = 1/\sqrt{T}$, we conclude

$$\mathbb{E}\big[f(x_T^{(a)}) - f(x^*)\big] \le \frac{\|x_0 - x^*\|^2}{2\eta \cdot T/2} + \frac{5}{2}\eta G^2 = \frac{\|x_0 - x^*\|^2}{\sqrt{T}} + \frac{5G^2}{2\sqrt{T}} = \frac{1}{2\sqrt{T}}\left(2\|x_0 - x^*\|^2 + 5G^2\right).$$

□

Theorem C.7 (Non-convex convergence)**.** *Assume that f is L-smooth, that the stochastic gradient is unbiased,*

$$\mathbb{E}[\,\nabla f_{S_t}(x_t) \mid x_t\,] = \nabla f(x_t),$$

and that Assumption 34 holds. Consider the iterates generated by (35) with

$$\eta = \frac{1}{\sqrt{T}}, \qquad \beta_{\max} = c\eta, \qquad 0 < c < 1.$$

Then

$$\min_{0 \le t \le T-1} \mathbb{E}\big[\|\nabla f(x_t)\|^2\big] \le \frac{1}{\sqrt{T}(1-c)}\left(\mathbb{E}[f(x_0) - f(x_T)] + \frac{1+8L}{4}G^2\right).$$

Proof. Let

$$g_t := \nabla f_{S_t}(x_t).$$

By L-smoothness of f,

$$\begin{aligned} f(x_{t+1}) &\le f(x_t) + \langle \nabla f(x_t),\, x_{t+1} - x_t\rangle + \frac{L}{2}\|x_{t+1} - x_t\|^2 \\ &= f(x_t) - \eta\langle \nabla f(x_t),\, d_{t+1}\rangle + \frac{L\eta^2}{2}\|d_{t+1}\|^2. \end{aligned}$$

Using

$$d_{t+1} = g_t + \beta_t(d_t - g_t),$$

we obtain

$$f(x_{t+1}) \le f(x_t) - \eta\langle \nabla f(x_t),\, g_t\rangle - \eta\beta_t\langle \nabla f(x_t),\, d_t - g_t\rangle + \frac{L\eta^2}{2}\|d_{t+1}\|^2.$$

Applying Lemma C.5 with $a_t = \frac{1}{2}$,

$$-\eta\beta_t\langle \nabla f(x_t),\, d_t - g_t\rangle \le \beta_{\max}\|\nabla f(x_t)\|^2 + \frac{\eta^2}{4}\|g_t\|^2.$$

Hence,

$$f(x_{t+1}) \le f(x_t) - \eta\langle \nabla f(x_t),\, g_t\rangle + \beta_{\max}\|\nabla f(x_t)\|^2 + \frac{\eta^2}{4}\|g_t\|^2 + \frac{L\eta^2}{2}\|d_{t+1}\|^2.$$

Taking expectation and using unbiasedness,

$$\mathbb{E}\big[\langle \nabla f(x_t),\, g_t\rangle\big] = \mathbb{E}\big[\|\nabla f(x_t)\|^2\big].$$

Using Assumption 34 and Lemma C.3,

$$\mathbb{E}\big[\|g_t\|^2\big] \le G^2, \qquad \mathbb{E}\big[\|d_{t+1}\|^2\big] \le 4G^2.$$

Therefore,

$$\mathbb{E}[f(x_{t+1})] \le \mathbb{E}[f(x_t)] - \big(\eta - \beta_{\max}\big)\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] + \eta^2 G^2\left(\frac{1}{4} + 2L\right).$$

Equivalently,

$$\big(\eta - \beta_{\max}\big)\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] \le \mathbb{E}[f(x_t) - f(x_{t+1})] + \eta^2 G^2\left(\frac{1}{4} + 2L\right).$$

Summing from $t = 0$ to $T-1$,

$$\big(\eta - \beta_{\max}\big)\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] \le \mathbb{E}[f(x_0) - f(x_T)] + T\eta^2 G^2\left(\frac{1}{4} + 2L\right).$$

Since

$$\frac{1}{4} + 2L = \frac{1 + 8L}{4},$$

we obtain

$$\big(\eta - \beta_{\max}\big)\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] \le \mathbb{E}[f(x_0) - f(x_T)] + \frac{1+8L}{4}\,T\eta^2 G^2.$$

Using $\beta_{\max} = c\eta$, we have

$$\eta - \beta_{\max} = \eta(1 - c).$$

Therefore,

$$\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] \le \frac{1}{\eta(1-c)}\left(\mathbb{E}[f(x_0) - f(x_T)] + \frac{1+8L}{4}\,T\eta^2 G^2\right).$$

Dividing by T and using $\eta = 1/\sqrt{T}$,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] \le \frac{1}{\sqrt{T}(1-c)}\left(\mathbb{E}[f(x_0) - f(x_T)] + \frac{1+8L}{4}G^2\right).$$

Finally,

$$\min_{0\le t\le T-1}\mathbb{E}\big[\|\nabla f(x_t)\|^2\big] \le \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(x_t)\|^2\big],$$

which proves the claim. □

C.1 A fixed-cap variant.

The main text uses a shrinking cap on $\beta_{\max}$ to obtain convergence to the exact minimizer. If instead the cap is kept fixed, one can still obtain an $O(1/\sqrt{T})$ convergence rate to a noise-dependent neighborhood under the following finite-gap condition.

Assumption C.8 (Finite optimal mini-batch gap)**.** Let $x^* \in \arg\min_x f(x)$, and define

$$f_S^* := \inf_x f_S(x).$$

Assume there exists $\sigma_f^2 < \infty$ such that

$$\mathbb{E}_S[f_S(x^*) - f_S^*] \le \sigma_f^2.$$

We consider the same simplified AM-SGD-M update as in (35), but now with a fixed cap

$$0 \le \beta_t \le \beta_{\max} < 1.$$

For convenience, define

$$\epsilon := 1 - \beta_{\max} > 0.$$

Lemma C.9. *Under Assumption 3.2, for every t,*

$$\mathbb{E}\big[\|d_{t+1}\|^2\big] \le \frac{G^2}{\epsilon}.$$

Proof. Using the update $d_{t+1} = \beta_t d_t + (1-\beta_t) g_t$ and Jensen's inequality,

$$\|d_{t+1}\|^2 = \|\beta_t d_t + (1-\beta_t) g_t\|^2 \le \beta_t \|d_t\|^2 + (1-\beta_t)\|g_t\|^2 \le \beta_{\max}\|d_t\|^2 + \|g_t\|^2.$$

Taking expectation and using Assumption 3.2,

$$\mathbb{E}\big[\|d_{t+1}\|^2\big] \le \beta_{\max}\, \mathbb{E}\big[\|d_t\|^2\big] + G^2.$$

Since $d_0 = 0$, unrolling the recursion gives

$$\mathbb{E}\big[\|d_{t+1}\|^2\big] \le G^2 \sum_{k=0}^{t} \beta_{\max}^k \le \frac{G^2}{1-\beta_{\max}} = \frac{G^2}{\epsilon}.$$

□

Lemma C.10. *Assume each f_S is convex, Assumption 3.2 holds, and Assumption C.8 holds. Then, for every t,*

$$\mathbb{E}\big[-\beta_t \langle x_t - x^*,\, d_t\rangle\big] \le \frac{\eta G^2}{\epsilon^2} + \frac{\sigma_f^2}{\epsilon}.$$

Proof. Set

$$A_t := -\langle x_t - x^*,\, d_t\rangle.$$

Using $x_t = x_{t-1} - \eta d_t$, we write

$$\begin{aligned} A_t &= -\langle x_t - x^*,\, d_t\rangle \\ &= -\langle x_t - x_{t-1},\, d_t\rangle - \langle x_{t-1} - x^*,\, d_t\rangle \\ &= \eta\|d_t\|^2 - \langle x_{t-1} - x^*,\, \beta_{t-1} d_{t-1} + (1-\beta_{t-1}) g_{t-1}\rangle \\ &= \eta\|d_t\|^2 + \beta_{t-1} A_{t-1} - (1-\beta_{t-1})\langle x_{t-1} - x^*,\, g_{t-1}\rangle. \end{aligned}$$

By convexity of $f_{S_{t-1}}$,

$$\langle x_{t-1} - x^*,\, g_{t-1}\rangle \ge f_{S_{t-1}}(x_{t-1}) - f_{S_{t-1}}(x^*) \ge f^*_{S_{t-1}} - f_{S_{t-1}}(x^*).$$

Hence

$$-(1-\beta_{t-1})\langle x_{t-1} - x^*,\, g_{t-1}\rangle \le (1-\beta_{t-1})\big(f_{S_{t-1}}(x^*) - f^*_{S_{t-1}}\big) \le f_{S_{t-1}}(x^*) - f^*_{S_{t-1}}.$$

Therefore

$$A_t \le \eta\|d_t\|^2 + \beta_{t-1} A_{t-1} + \big(f_{S_{t-1}}(x^*) - f^*_{S_{t-1}}\big).$$

Since $A_0 = 0$ (because $d_0 = 0$), unrolling the recursion gives

$$A_t \le \sum_{j=1}^{t} \left(\prod_{k=j}^{t-1} \beta_k\right) \Big[\eta\|d_j\|^2 + f_{S_{j-1}}(x^*) - f^*_{S_{j-1}}\Big].$$

Using $\beta_k \le \beta_{\max} = 1 - \epsilon$, we obtain

$$A_t \le \sum_{j=1}^{t} (1-\epsilon)^{\,t-j} \Big[\eta\|d_j\|^2 + f_{S_{j-1}}(x^*) - f^*_{S_{j-1}}\Big].$$

Since $0 \le \beta_t \le 1$, we also have $\beta_t A_t \le A_t$, and thus

$$-\beta_t \langle x_t - x^*, d_t \rangle \le \sum_{j=1}^{t} (1-\epsilon)^{t-j} \left[\eta \|d_j\|^2 + f_{S_{j-1}}(x^*) - f^*_{S_{j-1}} \right].$$

Taking expectation and using Lemma C.9 together with Assumption C.8,

$$\begin{aligned} \mathbb{E}\big[-\beta_t \langle x_t - x^*, d_t \rangle\big] &\le \sum_{j=1}^{t} (1-\epsilon)^{t-j} \left[\eta \frac{G^2}{\epsilon} + \sigma_f^2 \right] \\ &\le \frac{1}{\epsilon} \left(\eta \frac{G^2}{\epsilon} + \sigma_f^2 \right) = \frac{\eta G^2}{\epsilon^2} + \frac{\sigma_f^2}{\epsilon}. \end{aligned}$$

□

Theorem C.11 (Convex convergence under a fixed cap)**.** *Assume each f_S is convex, Assumption 3.2 holds, Assumption C.8 holds, and $\mathbb{E}[g_t \mid x_t] = \nabla f(x_t)$. Consider the iterates generated by (35) with a fixed cap $0 \le \beta_t \le \beta_{\max} < 1$, and let $\epsilon := 1 - \beta_{\max}$. Define*

$$\bar{x}_T := \frac{1}{T} \sum_{t=0}^{T-1} x_t.$$

Then

$$\mathbb{E}\big[f(\bar{x}_T) - f(x^*)\big] \le \frac{\|x_0 - x^*\|^2}{2\eta\epsilon T} + \eta G^2 \frac{2+\epsilon}{2\epsilon^3} + \sigma_f^2 \frac{1+\epsilon-\epsilon^2}{\epsilon^2}.$$

In particular, with $\eta = T^{-1/2}$,

$$\mathbb{E}\big[f(\bar{x}_T) - f(x^*)\big] \le \frac{\|x_0 - x^*\|^2}{2\epsilon\sqrt{T}} + \frac{G^2(2+\epsilon)}{2\epsilon^3\sqrt{T}} + \sigma_f^2 \frac{1+\epsilon-\epsilon^2}{\epsilon^2},$$

that is, the method approaches a σ_f^2-dependent neighborhood at rate $O(1/\sqrt{T})$.

Proof. Let

$$\Delta_t := \|x_t - x^*\|^2.$$

Using the update $x_{t+1} = x_t - \eta d_{t+1}$,

$$\Delta_{t+1} = \Delta_t - 2\eta \langle x_t - x^*, d_{t+1} \rangle + \eta^2 \|d_{t+1}\|^2.$$

Expanding $d_{t+1} = \beta_t d_t + (1-\beta_t) g_t$,

$$\Delta_{t+1} = \Delta_t - 2\eta\beta_t \langle x_t - x^*, d_t \rangle - 2\eta(1-\beta_t) \langle x_t - x^*, g_t \rangle + \eta^2 \|d_{t+1}\|^2.$$

By convexity of f_{S_t},

$$\langle x_t - x^*, g_t \rangle \ge f_{S_t}(x_t) - f_{S_t}(x^*).$$

Therefore

$$\Delta_{t+1} \le \Delta_t - 2\eta\beta_t \langle x_t - x^*, d_t \rangle - 2\eta(1-\beta_t)\big(f_{S_t}(x_t) - f_{S_t}(x^*)\big) + \eta^2 \|d_{t+1}\|^2.$$

Now write

$$f_{S_t}(x_t) - f_{S_t}(x^*) = \big(f_{S_t}(x_t) - f^*_{S_t}\big) - \big(f_{S_t}(x^*) - f^*_{S_t}\big).$$

Since $f_{S_t}(x_t) - f^*_{S_t} \ge 0$ and $1 - \beta_t \ge \epsilon$,

$$-(1-\beta_t)\big(f_{S_t}(x_t) - f_{S_t}(x^*)\big) \le -\epsilon\big(f_{S_t}(x_t) - f_{S_t}(x^*)\big) + (1-\epsilon)\big(f_{S_t}(x^*) - f^*_{S_t}\big).$$

Substituting this bound yields

$$\Delta_{t+1} \le \Delta_t - 2\eta\epsilon\big(f_{S_t}(x_t) - f_{S_t}(x^*)\big) + 2\eta(1-\epsilon)\big(f_{S_t}(x^*) - f^*_{S_t}\big) - 2\eta\beta_t \langle x_t - x^*, d_t \rangle + \eta^2 \|d_{t+1}\|^2.$$

Taking expectation and using $\mathbb{E}[g_t \mid x_t] = \nabla f(x_t)$,

$$\mathbb{E}\big[f_{S_t}(x_t) - f_{S_t}(x^*)\big] = \mathbb{E}\big[f(x_t) - f(x^*)\big].$$

Applying Assumption C.8, Lemma C.10, and Lemma C.9, we get

$$\mathbb{E}[\Delta_{t+1}] \le \mathbb{E}[\Delta_t] - 2\eta\epsilon\,\mathbb{E}\big[f(x_t) - f(x^*)\big] + 2\eta(1-\epsilon)\sigma_f^2 + 2\eta\left(\frac{\eta G^2}{\epsilon^2} + \frac{\sigma_f^2}{\epsilon}\right) + \eta^2\frac{G^2}{\epsilon}.$$

Rearranging,

$$2\eta\epsilon\,\mathbb{E}\big[f(x_t) - f(x^*)\big] \le \mathbb{E}[\Delta_t] - \mathbb{E}[\Delta_{t+1}] + \eta^2 G^2\left(\frac{2}{\epsilon^2} + \frac{1}{\epsilon}\right) + 2\eta\sigma_f^2\left((1-\epsilon) + \frac{1}{\epsilon}\right).$$

Summing from $t = 0$ to $T-1$ and telescoping,

$$\sum_{t=0}^{T-1}\mathbb{E}\big[f(x_t) - f(x^*)\big] \le \frac{\|x_0 - x^*\|^2}{2\eta\epsilon} + T\left[\eta G^2\,\frac{2+\epsilon}{2\epsilon^3} + \sigma_f^2\,\frac{1+\epsilon-\epsilon^2}{\epsilon^2}\right].$$

Finally, by convexity of f and Jensen's inequality,

$$\mathbb{E}\big[f(\bar{x}_T) - f(x^*)\big] \le \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[f(x_t) - f(x^*)\big],$$

which proves the claim. □

The fixed-cap result differs from the main theorem in that a nonvanishing cap $\beta_{\max} < 1$ leaves a residual stochastic bias in the adaptive memory term, which appears through the mini-batch mismatch quantity σ_f^2. As a result, the method approaches the solution set at rate $O(1/\sqrt{T})$, but only up to a noise-dependent neighborhood whose radius is controlled by σ_f^2. This is analogous to the usual behaviour of stochastic methods under persistent gradient noise: with a fixed amount of memory, one retains a nonzero bias floor unless the noise itself vanishes. In the common interpolating regime, however, x^* minimizes each mini-batch loss as well, so $f_S(x^*) = f_S^*$ almost surely and hence $\sigma_f^2 = 0$. In that case the neighborhood term disappears, and the bound reduces to exact $O(1/\sqrt{T})$ convergence.

D ALGORITHM DETAILS

Algorithm 2 AdamW: Adam with Decoupled Weight Decay

Initialize: Learning rate η, weight decay μ, $\beta_1, \beta_2 \in [0, 1)$, $\epsilon > 0$
Initialize parameters x_0, first moment $d_0 = 0$, second moment $v_0 = 0$
for each iteration $t = 1, 2, \ldots, T$ **do**
  Compute gradient: $g_t = \nabla f(x_t)$
  Update biased first moment estimate: $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  Update biased second moment estimate: $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  Compute bias-corrected first moment: $\hat{m}_t = \frac{m_t}{1-\beta_1^t}$
  Compute bias-corrected second moment: $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$
  Compute update direction: $\hat{d}_t = \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$
  Apply weight decay: $x_{t+1} = x_t - \eta(\hat{d}_t + \mu x_t)$
end for
Return: Learned parameters x_T

Algorithm 3 AM-AdamW: Adaptive Memory - Adam with Decoupled Weight Decay

Initialize: Learning rate η, weight decay μ, $\beta_{1\max}, \beta_2 \in [0, 1)$, $\epsilon > 0$, λ
Initialize parameters x_0, first moment $m_0 = 0$, second moment $v_0 = 0$
for each iteration $t = 1, 2, \ldots, T$ **do**
  Compute gradient: $g_t = \nabla f(x_t)$
  Update biased second moment estimate: $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  Compute bias-corrected second moment: $\hat{v}_t = \frac{v_t}{1-\beta_2^t}$
  Compute Adam Preconditioner: $P_t = (1 - \beta_{1\max} \prod_t \beta_{1t}) \mathrm{diag}\big(\epsilon + \sqrt{\hat{v}_t}\big)$
  Compute adaptive β_{1t}: $\beta_{1t} = \mathrm{Clip}_{[0,\beta_{1\max}]}\left(\frac{\frac{(1+\mu\eta)(\hat{f}(x_t) - f(x_t))}{\eta} - \langle d_t - g_t, g_t + \lambda P_t d_t \rangle_{P_t^{-1}(\lambda P_t + I)^{-1}} - \mu \langle x_t^\top, d_t - g_t \rangle}{\|d_t - g_t\|^2_{P_t^{-1}(\lambda P_t + I)^{-1}}} \right)$
  Update first moment estimate: $d_{t+1} = (\lambda P_t + I)^{-1} \left((1 - \beta_{1t}) g_t + (\lambda P_t + \beta_{1t} I) d_t \right)$
  if Proximal Weight Decay **then**
    Update Parameters: $x_{t+1} = \frac{1}{1+\mu\eta}\left(x_t - \eta P_t^{-1} d_{t+1} \right)$
  else if Decoupled Weight Decay **then**
    Update Parameters: $x_{t+1} = (1 - \mu\eta) x_t - \eta P_t^{-1} d_{t+1}$
  end if
end for
Return: Learned parameters x_T

D.1 Practical Considerations

In Adam, the preconditioner depends slightly on the value of β_{1t} due to the bias correction term. This dependence prevents closed-form computation of β_{1t}, but because the bias correction decays exponentially, it can be safely ignored in practice by using the approximation $\prod_{i=1}^{t-1} \beta_{1i}$, which has been shown to incur no practical drawbacks. Alternatively, the correction can be conservatively bounded by multiplying with the maximum value $\beta_{1\max}$. Moreover, for standard choices of learning rate and weight decay, both coupled and decoupled weight decay behave similarly. In our experiments, we chose to multiply the bias correction factor by $\beta_{1\max}$ and to use decoupled weight decay throughout.

D.2 Stochastic AdamW

We again replace $f(x_{t-1}, s_t) - f(x_t, s_t)$ with its first-order approximation around x_t, given by:

$$\begin{aligned} f(x_{t-1}, s_t) - f(x_t, s_t) &\approx \nabla f(x_t, s_t)^\top (x_{t-1} - x_t) \\ &= \eta_{t-1} g_t^T (P_{t-1}^{-1} d_t + \mu x_t) \\ &\approx \eta_{t-1} g_t^T (P_t^{-1} d_t + \mu x_t) \qquad \text{(Assuming } P_{t-1}^{-1} \approx P_t^{-1}\text{)} \end{aligned}$$

Where to avoid storing 2 preconditioners (2 second moment estimates), we approximate P_{t-1}^{-1} with P_t^{-1}.

D.3 Per-Layer AdamW

Our method requires computing the preconditioner twice: once to determine the value of β_1 and once again to perform the parameter update. Although this overhead is negligible in large-scale experiments, it can be eliminated, and even lead to minor performance gains, by computing a separate β_1 for each layer. In this variant, the same preconditioner can be reused within each layer: first to compute β_1, and then to update that layer's parameters. This approach not only improves efficiency but also allows different momentum parameters to be assigned to different layers. However, our empirical observations indicate that the momentum dynamics across layers are relatively similar. A comparison of our method with and without per-layer adjustment on the LLaMA 100M experiment is provided in Table 3.

Table 3: Comparison of our method on LLaMA-100M with shared vs. per-layer momentum parameter β_1 and with or without the $P_{t-1}^{-1} \approx P_t^{-1}$ approximation . In all cases the performance is similar, for computational efficiency all experiments in the main paper were conducted using per-layer β_1 and $P_{t-1}^{-1} \approx P_t^{-1}$

Method	$\mathbf{P_{t-1}^{-1}}$	$\mathbf{P_{t-1}^{-1} \approx P_t^{-1}}$
Shared β_1	3.811±0.088	3.814±0.088
Per-layer β_1	3.792±0.085	3.792±0.084

D.4 Computation Overhead and Convergence Speedup

We compare the computational cost and convergence speedup of **AM-AdamW** relative to the standard AdamW optimizer across models of varying sizes. Table 4 shows that the adaptive momentum mechanism introduces negligible overhead, below 0.5% even for small models, while consistently accelerating convergence by up to 1.5× for larger models.

Table 4: Average per-iteration runtime and relative convergence speedup of AM-AdamW compared to AdamW. Overhead is the relative increase in iteration time; speedup is the ratio of steps to reach the same validation loss.

Model Size	**Avg. Time per Iteration (AdamW)**	**Avg. Time per Iteration (AM-AdamW)**	**AM-AdamW Overhead (%)**	**Convergence Speedup (AM-AdamW/AdamW)**
20M	1.21s	1.21s	0.31%	1.11×
60M	2.77s	2.78s	0.36%	1.41×
100M	6.23s	6.26s	0.48%	1.45×
350M	13.91s	13.95s	0.29%	1.52×
1B	78.89s	78.98s	0.11%	1.54×

E EXPERIMENTAL SETUP FOR SECTION 4

E.1 Convex problems

For the binary classification tasks, the learning rate was chosen based on an estimate of the smoothness of the feature matrix. The same procedure was applied to the multiclass classification problems. While this approach is not theoretically justified for all optimizers, it provided a consistent and practical basis for comparison. Importantly, for each dataset, the same learning rate was used across all optimizers to ensure fairness. For Adam, we fixed the learning rate to 0.001 for all experiments. We used the codebase from https://github.com/konstmish/opt_methods

E.2 Image Classification

Dataset Augmentations

- **ConvNets for CIFAR10/100:** For CIFAR-10 and CIFAR-100, the training pipeline consists of random cropping to 32×32 with a padding of 4, random horizontal flipping and normalization using dataset-specific mean and standard deviation. The validation pipeline includes only normalization.
- **ResNet50 for ImageNet:** For ResNet-50, the training pipeline consists of random resized cropping to 224×224, random horizontal flipping and normalization using ImageNet mean and standard deviation. During validation, the transformations include resizing to 256 pixels, center cropping to 224×224, conversion to a tensor, and the same normalization applied in training.

Table 5: Hyper-parameter settings for the CIFAR100 experiments.

Hyper-parameter	**Value**
Architecture	ResNet50 and WRN-40-10
Epochs	200
Batch size	128
Optimizers	MGD
LR schedule	cosine
Weight decay	0
Momentum for MGD	0.9
Learning Rate	0.1
AM-MGD λ	0.1
$\beta_{\max}$	$0.9 - 0.1\lambda$

Table 6: Hyper-parameter settings for the CIFAR10 experiments.

Hyper-parameter	**Value**
Architecture	ResNet18 and VGG-19
Epochs	200
Batch size	128
Optimizers	MGD
LR schedule	step with r=0.1 at [100,150]
Weight decay	0
Momentum for MGD	0.9
Learning Rate	0.1
AM-MGD λ	0.1
$\beta_{\max}$	$0.9 - 0.1\lambda$

For the ablation studies, we used the same experimental settings, varying only the hyperparameter under investigation. Additionally, no learning rate schedulers were applied.

Table 7: Hyper-parameter settings for the ResNet18 trained on ImageNet experiments.

Hyper-parameter	**Value**
Architecture	ResNet18
Epochs	100
Batch size	256
Optimizers	MGD
LR schedule	step with r=0.1 at [30,60,90]
Weight decay	1e-4
Momentum for MGD	0.9
Learning Rate	1
AM-MGD λ	0.1
$\beta_{\max}$	$0.9 - 0.1\lambda$

E.3 Pretraining Large Language Models

Table 8: Hyper-parameter settings for the LLaMA experiments.

Hyper-parameter	**Value**
Learning Rate	0.001
Weight Decay	0.0001
Batch Size	1024
Model Precision	BF16
Scheduler	Cosine with warm-up
Warm-up Ratio	10%
Grad Clipping	1.0
β_1	0.9
β_2	0.999
ϵ	1e-8
Seq-len	1024
AM-AdamW λ	0.1
$\beta_{1\,\max}$	$0.9 - 0.1\lambda$
Eval Precision	BF16
Eval Seq-len	1024

Table 9: Architectural details for the LLaMA models.

Component	**Value**
# Parameters	20M / 60M / 100M / 350M / 1B
Hidden Size	256 / 512 / 640 / 1024 / 2048
Intermediate Size	688 / 1376 / 1708 / 2736 / 5461
Attention Heads	4 / 8 / 10 / 16 / 32
Hidden Layers	4 / 8 / 12 / 24 / 24
Activation Function	`silu`
Normalization	RMSNorm (ϵ = 1e−6)
Vocab Size	32,000
Max Seq. Length	1024
Initializer Range	0.02
Model Type	`llama`
Transformers Version	4.28.1

F OMITTED EXPERIMENTAL RESULTS

F.1 Convex Problems

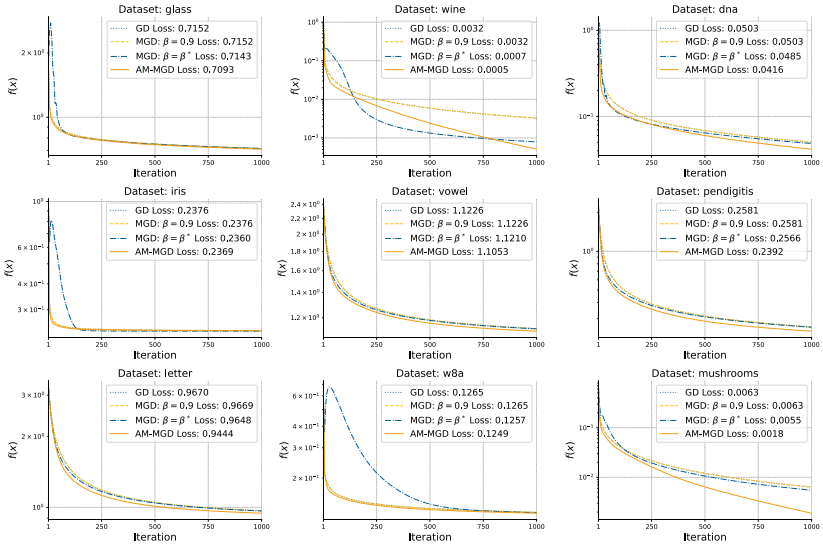


Figure 8: Logistic Loss over time, AM-MGD (red curve) outperforms fixed momentum on all but one experiments. GD and MGD with $\beta = 0.9$ practically overlap. We run all algorithms for 10000 iterations with a learning rate $\eta = 1/L$ where L is the smoothness

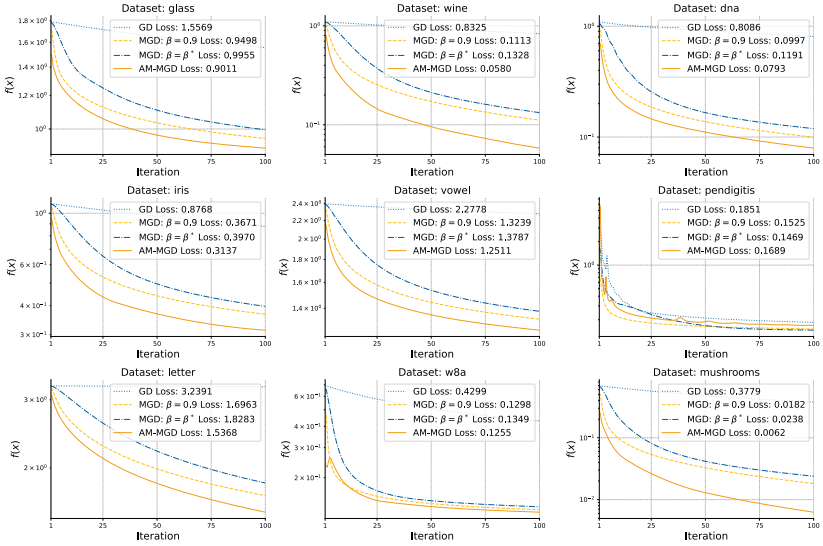


Figure 9: Logistic Loss over time, AM-Adam (purple curve) outperforms fixed momentum on all but one experiments. We run all algorithms for 1000 iterations with a learning rate of η=0.001

F.2 Detailed Curves for Figure 6c

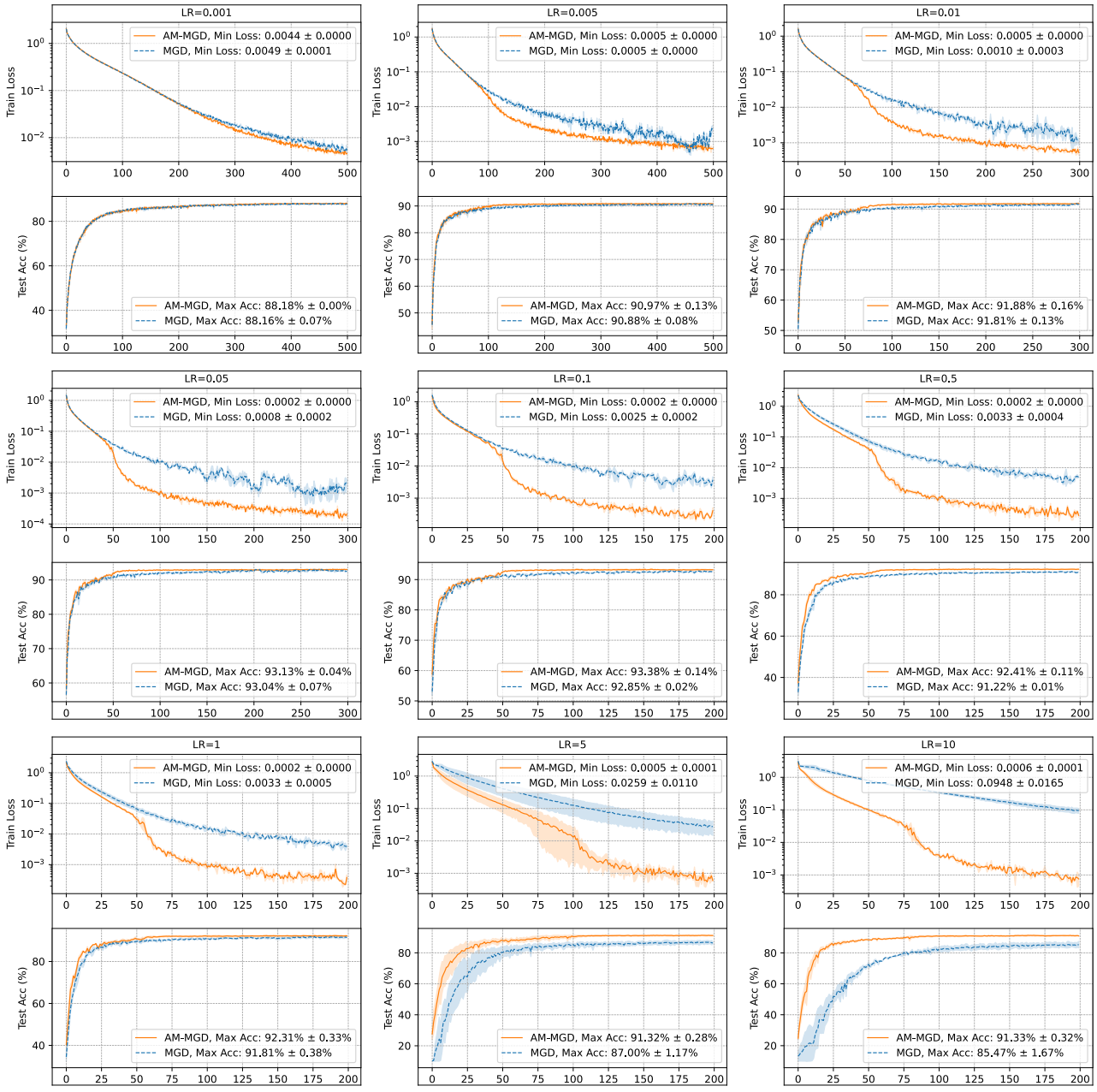


Figure 10: Training curves for different learning rates on ResNet18 trained on CIFAR10

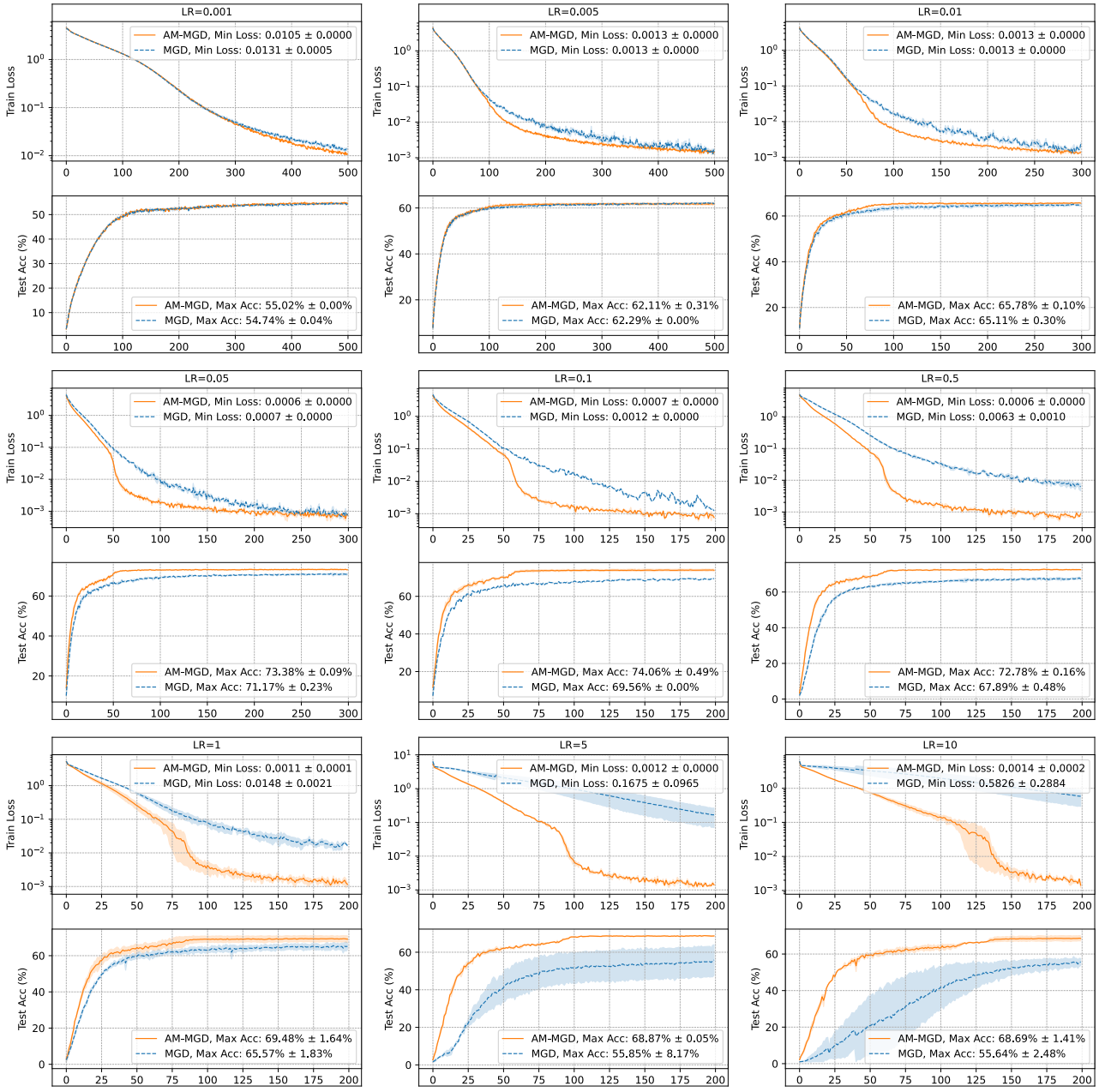


Figure 11: Training curves for different learning rates on ResNet50 trained on CIFAR100

F.3 More plots on the behaviour of β_t

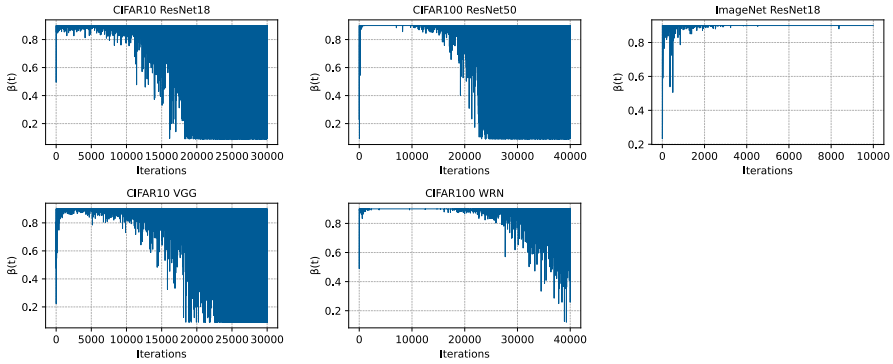


Figure 12: The adaptive β_t across different experiments

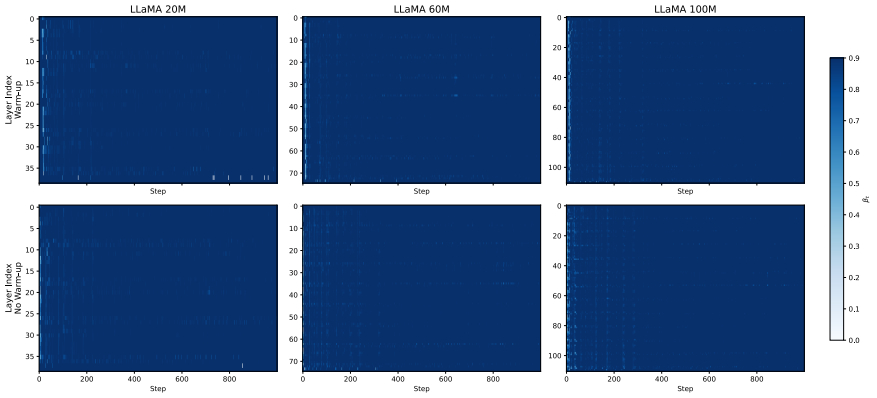


Figure 13: The layer-wise adaptive β_t across different LLaMA experiments

F.4 ImageNet Experiment

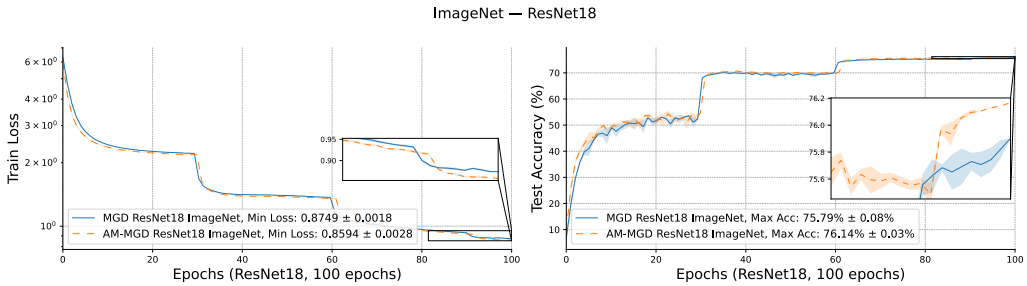


Figure 14: ResNet18 on ImageNet

F.5 Comparison with other Optimizers

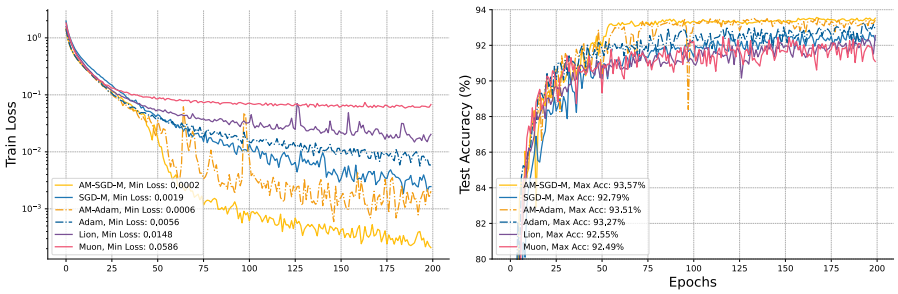
We also compare our method against the Muon and Lion optimizers. We use the following publicly available implementations:

- **Lion:** github.com/huggingface/pytorch-image-models
- **Muon:** github.com/KellerJordan/Muon

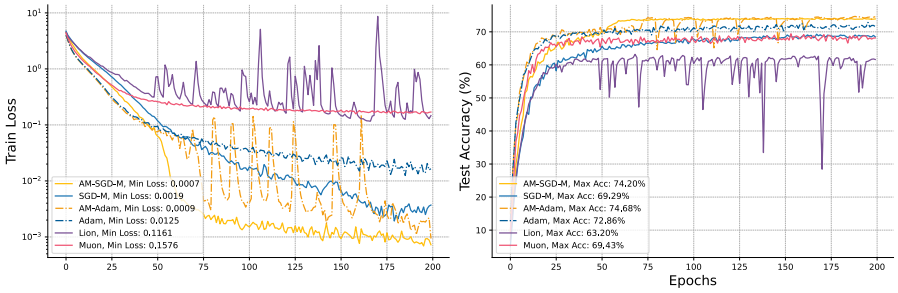
For each optimizer, we perform a coarse hyperparameter grid search shown in Table 10. In our LLaMA experiments, we adopt the setup with warm-up enabled. For both Lion and Muon, the hyperparameter search is conducted only on the 20M model to assess the transferability of hyperparameters to larger model scales.

Table 10: Hyperparameter search ranges for Muon and Lion optimizers.

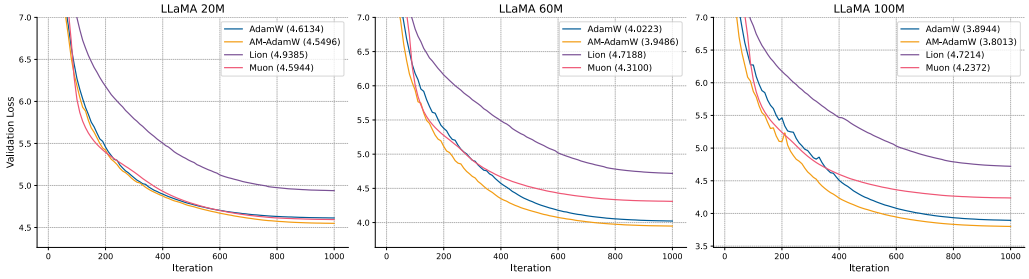
Optimizer	**Hyperparameter**	**Values Tested**
Muon	Learning rate η	[1e-4, 5e-4, 1e-3, 5e-3, 1e-2]
	Momentum coefficient β	[0.9, 0.95, 0.99]
Lion	Learning rate η	[1e-4, 5e-4, 1e-3, 5e-3, 1e-2]
	Second-moment coefficient β_2	[0.9, 0.95, 0.99, 0.999]



(a) ResNet18 on CIFAR10



(b) ResNet50 on CIFAR100



(c) Three scales of LLaMA on C4

Figure 15: Optimizer comparisons across different models and datasets.