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

Kristi Topollai
New York University
kt2664@nyu.edu

Anna Choromanska
New York University
ac5455@nyu.edu

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.

1 Introduction

Stochastic Gradient Descent (SGD) [5] and its variants [62, 31] 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:

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

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 [49, 44]. Achieving such acceleration in practice with Heavy Ball (HB) momentum [49] or Nesterov Accelerated Gradient (NAG) [44] requires carefully tuning η and β, or using time-dependent schedules [44]. 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 simple 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 [3, 10] and the proximal bundle method [33]. 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 decent 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 7) 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 [49] 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) [44] 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 [64, 55], where it provably matches the convergence rate of SGD [69, 58]. 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 [62]. 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 [25, 42, 23, 16, 37] and is used numerous optimization algorithms. These include Adam(W) [31, 40] and its variants [70, 59, 48], as well as other momentum-based methods like Lion[9], Sophia [35], SOAP [65], MARS [71], and Muon [26, 36]. 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 [53], which update iterates via the 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 $f_t^m(x)$ [2], which yields a general model-based optimization framework used in both deterministic and stochastic settings [10, 3, 6]. Gradient Descent, SGD [52], and Subgradient Descent [50] 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 [29](Equation 7), lead to the Proximal Bundle Method [32], especially useful in non-smooth convex optimization is:

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

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 [50, 39], and more recently, adaptive learning rates for momentum gradient descent [57, 67, 45]. 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 [43, 4, 56].

In stochastic optimization, most work focuses on adaptive learning rates. Methods such as stochastic Polyak step sizes [39, 46] adapt the step size based on the suboptimality gap $f(x_t) - f^*$, while others rely on distance-to-optimum estimates [12, 24]. In deep learning, adaptive diagonal pre-conditioners, used in AdaGrad [14], Adam [31], and related methods [22, 35], are standard for per-parameter step size adjustment. Beyond adaptive schemes, fixed learning rate schedules [61, 41] and warm-up strategies [17] 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 [68]. 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 [27, 34].

In contrast, adapting the momentum coefficient has received little attention. Heuristic schedules [62, 8] increase or decrease momentum over time, but have not been adopted in practice. Restart-based methods [15, 47] propose clearing the momentum buffer when certain technical conditions are met [47] or according to pre-defined schedules [66] and, while effective, they often require handcrafted restart rules. Conjugate gradient (CG) and nonlinear CG [20] 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.

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

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 14, 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.

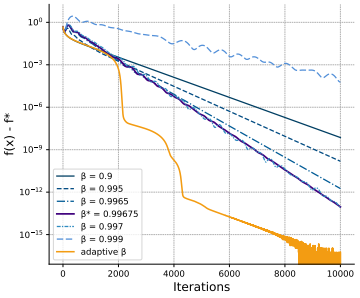

Figure 1: Optimization with a fixed β vs adaptive memory momentum. We plot the error $f(x_t) - f^*$ over time t.

3.2 Approximate Cutting Planes Framework

Inspired from bundle methods [33], our approach for deriving an adaptive coefficient β for momentum methods is based on the following model of the function f:

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

where $\frac{1}{\eta}(x_{t-1} - x_t)$ is the previous decent 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 [30, 13], we introduce an extra regularization term and regularization parameter λ to control the extent to which the descent direction aligns with the previous decent 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 along with all the derivations, the minimizer can be expressed in the following heavy-ball update:

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

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

$$\max_{0\le\beta\le1}\left\{-\frac{\eta}{2(\lambda+1)}\left\|(1-\beta)g_t + \beta d_t + \lambda d_t\right\|_2^2 + (1-\beta)f(x_t) + \beta\hat{f}(x_t)\right\}. \tag{11}$$

The solution to which, is given by:

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

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.

Regarding the definition of the momentum plane in the model f_t^m, aggregation in proximal bundle methods typically sets $\hat{f}(x_{t+1}) = f_t^m(x_{t+1}) \leq f(x_t)$, for the next iteration, leveraging the property that $d_{t+1} \in \partial f_t^m(x_{t+1})$ [32]. 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, where we prove that for quadratic functions, minimizing $\|x_{t+1} - x^*\|_2^2$ under Momentum GD 5 with respect to β implies $\hat{f}(x_t) > f(x_t)$ for all t.

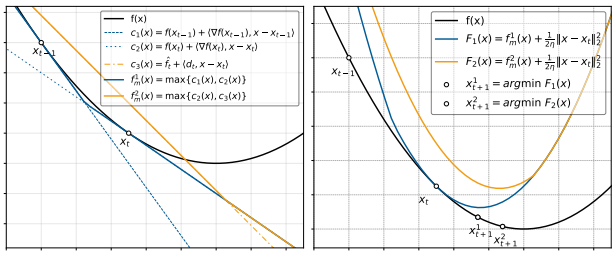

Figure 2: On the left, blue represents a typical cutting planes model, while yellow depicts our model, which may overestimate the function at x_t. On the right, we observe that with our model, the solution to Equation 6 lies closer to f^*, thus overestimation can get as closer to the optimum.

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 [31] and AdaGrad [14] 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 [40] 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:

$$d_{t+1} = (\lambda P_t + I)^{-1}\big((\beta_t^* I + \lambda P_t)d_t + (1 - \beta_t^*)g_t\big),$$

$$x_{t+1} = \frac{1}{1 + \mu\eta}\big(x_t - \eta P_t^{-1} d_{t+1}\big).$$

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

By using the Adam pre-conditioner: $P_t = (1 - \beta_1^t)\text{diag}\big(\epsilon + \sqrt{\frac{v_t}{1-\beta_2^t}}\big)$ 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

[2] for small η, $\frac{1}{1+\eta\mu} \approx 1 - \eta\mu$ and $\frac{\eta}{1+\eta\mu} \approx \eta$ [73]

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

Initialize: η, λ, x_0, $d_0 = \nabla f(x_0)$, $\beta_{\max,t}, d_0 = g_0$
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

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.

3.5 Convergence Guarantees

We provide convergence guarantees for our proposed methods in the standard finite-sum setting: $\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$, where each f_i denotes the loss on training example i, and the goal is to minimize the average loss f. We proceed under the following assumptions.

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

$$f(y) \le f(x) + \nabla f(x)^\top (y - x) + (L/2)\|y - x\|^2 \tag{15}$$

Assumption 3.2 (Bounded Stochastic Gradients)**.** There exist $G > 0$ such that :

$$\mathbb{E}[\|\nabla f_i(x)\|_2^2] \le G^2 \tag{16}$$

For our Adaptive Memory Momentum SGD, presented in Algorithm 1, we establish convergence guarantees under standard assumptions. The analysis leverages the natural bound on $\|d_t - \nabla f(x_t)\|_2^2$ that arises directly from the construction of the adaptive momentum coefficient β_t.

Theorem 3.3 (Convex Convergence)**.** *Let $f_i : \mathbb{R}^d \to \mathbb{R}$ convex, with stochastic gradients satisfying Assumption 3.2. Let $x^* \in \arg\min f > -\infty$ and define the averaged iterate $\bar{x}_T = \frac{1}{T}\sum_{t=1}^T x_t$. Then, for our method with $\beta_{\max} = 1/T$ and $\eta = 1/\sqrt{T}$ we have:*

$$\mathbb{E}\big[f(\bar{x}_T) - f(x^*)\big] \le \frac{1}{\sqrt{T}}\left(\frac{1}{2}\|x_0 - x^*\|^2 + G^2\right). \tag{17}$$

Theorem 3.4 (Non-convex Convergence)**.** *Let f be L-smooth , with stochastic gradients satisfying Assumption 3.2. Then, for our method with $\eta = 1/\sqrt{T}$ and $\beta_{\max} = c\eta$ with $0 < c < 1$, we have:*

$$\min_{0\le t\le T-1} \mathbb{E}\big[\|\nabla f(x_t)\|_2^2\big] \ \le\ \frac{1}{\sqrt{T}(1-c)}\left(\mathbb{E}[f(x_0) - f(x_T)] + \frac{(1+4L)G^2}{4}\right). \tag{18}$$

Our method attains the standard $O(1/\sqrt{T})$ convergence rate, consistent with existing analyses of heavy-ball-based algorithms under comparable assumptions [69, 58]. The principal technical difficulty arises from handling correlations introduced by the adaptive momentum coefficient β_t, which is defined as a function of the stochastic gradient at the current iterate. This dependence generates a nontrivial bias, since $\mathbb{E}_i[\beta_t \nabla f_i(x_t)] \neq \beta_t\, \mathbb{E}_i[\nabla f_i(x_t)]$. To establish our bounds, we impose an upper constraint on $\beta_{\max}$ that is inversely proportional with the horizon, effectively limiting the contribution of momentum. Although this may appear counterintuitive, it is in line with established theoretical insights suggesting that momentum can be detrimental in stochastic settings [67, 45, 57, 38]. Related decay conditions have also been employed in the analysis of the stochastic Polyak adaptive step to ensure convergence guarantees [46].

4 Experiments

4.1 Convex Problems

For our convex experiments, we used logistic regression on 9 datasets from the LIBSVM repository [7] (4 shown here; full results in the Supplement). Figure 3 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.

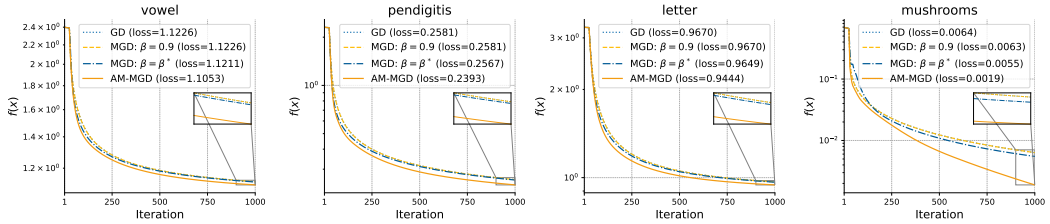

Figure 3: Logistic Loss over time, AM-MGD (red curve) outperforms fixed momentum on all 4 experiments. GD and MGD with $\beta = 0.9$ practically overlap. The λ was fixed to 0 in all runs.

4.2 Image Classification

We evaluated AM-MGD on standard image classification benchmarks: CIFAR-10, CIFAR-100 [1], and ImageNet [54], using architectures including VGG [60], ResNets [21] and Wide-ResNets [72]. Full experimental details and hyperparameters are provided in the Supplement. For all experiments, we adopted hyperparameter settings from prior works or tuned them using a standard baseline (MGD or AdamW); these settings were then held fixed when comparing against our adaptive variants. AM hyperparameters were kept constant across all experiments ($\beta_{\max} = 0.9$, $\lambda = 0.1$) without additional tuning, this choice is supported by ablation studies in Section 4.6. For MGD, we set PyTorch's dampening parameter to 0.9 to ensure a fair comparison at identical learning rates.

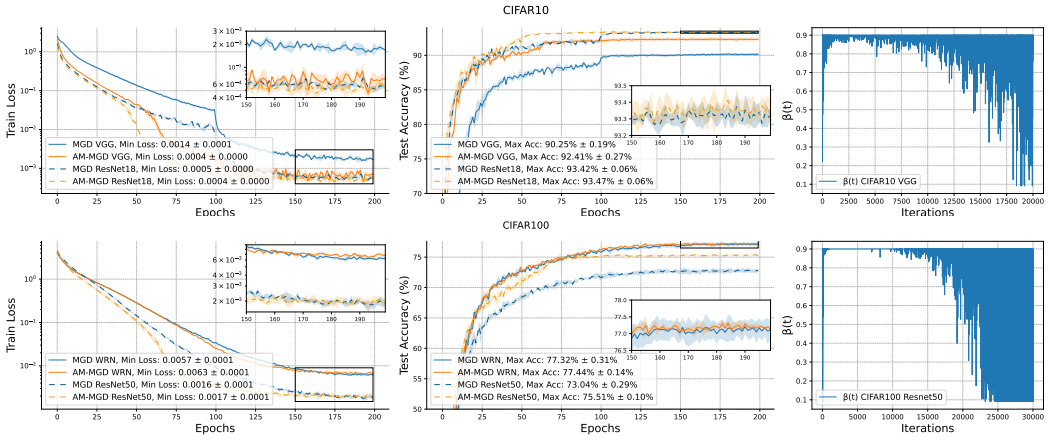

Figure 4: Train Loss, Test Accuracy and momentum parameter on the image classification experiments, (a) VGG19 and a ResNet18 on CIFAR10 (top), (b) ResNet50 and WideResNet on CIFAR100 (bottom)

As shown in Figure 4, AM-MGD consistently outperforms fixed-momentum baselines, demonstrating faster convergence and better generalization. On ResNets trained on CIFAR, a rapid convergence phase occurs between epochs 50 and 100, coinciding with dynamic changes in β Figure 4 (right column): initially stable, then oscillating between 0 and $\beta_{\max}$, effectively emulating SGD without momentum when needed. This behavior is consistent with theoretical findings [11], which suggest that momentum offers diminishing benefits as the iterates approach a local minimum. This reflects AM-MGD's ability to reset its memory when accumulation impedes progress. Further larger scale

experiments on Imagenet are included in the Supplement where AM-MGD again outperforms the fixed baseline.

4.3 Pretraining Large Language Models

We pretrained LLaMA models [63, 18] of varying scales on 1 billion tokens of the English subset of the C4 dataset [51]. Each model was trained under two regimes: one with the SOTA standard [19] linear learning rate warmup and one without, in order to investigate the effect of our method in the early 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 warmup, AM-AdamW compares to the performance of AdamW with warmup. In contrast, AdamW struggles to train reliably without warmup, 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 offers a hyperparameter-free alternative to warmup.

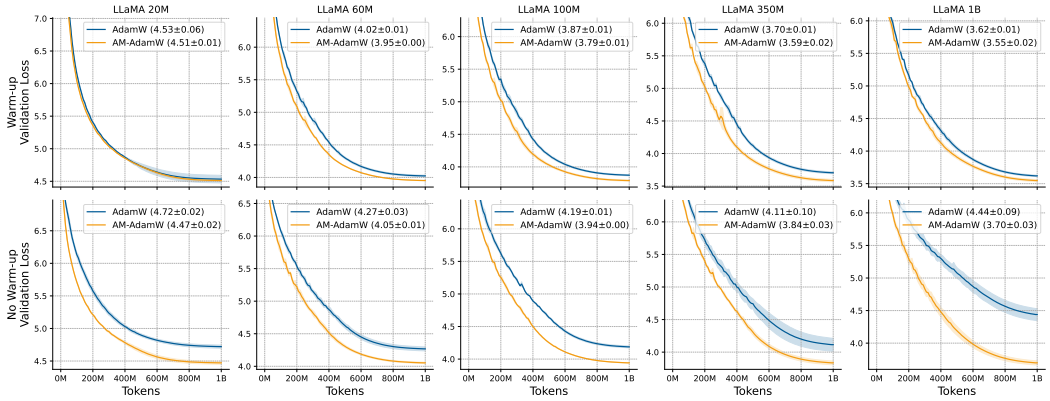

Figure 5: Validation loss curves for Llama pretraining on C4 across 5 different model scales.

4.4 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 $6NB$ [28], 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.

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.

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

We find that early restarts carry much of the benefit by rapidly adapting to unstable initial dynamics as we see in Table 6, however, these restarts alone do not explain 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

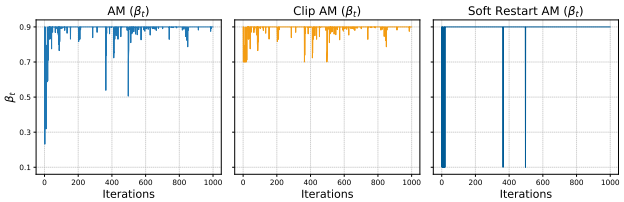

Figure 6: **Left:** The two alternative AM optimizers, with restarting threshold $\theta = 0.7$. **Right:** The three AM-derived policies.

4.6 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 7a: 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 behavior similar to that of a fixed momentum coefficient.

Ablation on batch size Figure 7b: 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 7c: We fix the hyperparameters to the values used in image classification experiments captured in Figure 4 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.

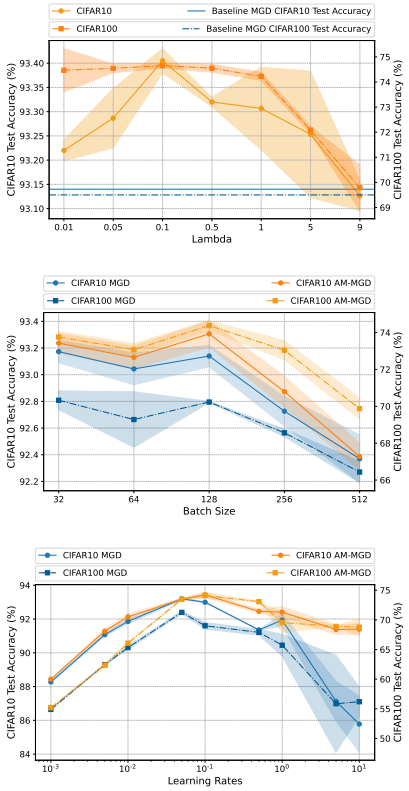

Figure 7: Final test accuracy for ResNet18 (CIFAR-10) and ResNet50 (CIFAR-100) under (top) λ, (middle) batch size, and (bottom) learning rate.

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 dynamically adapts to the local geometry of the loss landscape without introducing additional hyperparameters. 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 notoriously fragile early phase of LLM training and shows promise in eliminating the need for hand-tuned warmup schedules, demonstrating robustness to large learning rates and simplifying large-scale workflows. Future work includes: (i) further investigating the role of 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 warmup-free strategies to stabilize early training.

References

[1] K. Alex. Learning multiple layers of features from tiny images. *https://www. cs. toronto. edu/kriz/learning-features-2009-TR. pdf*, 2009.

[2] H. Asi, K. Chadha, G. Cheng, and J. C. Duchi. Minibatch stochastic approximate proximal point methods. *Advances in neural information processing systems*, 33:21958–21968, 2020.

[3] H. Asi and J. C. Duchi. Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity. *SIAM Journal on Optimization*, 29(3):2257–2290, 2019.

[4] M. Barré, A. Taylor, and A. d'Aspremont. Complexity guarantees for polyak steps with momentum. In *Conference on learning theory*, pages 452–478. PMLR, 2020.

[5] L. Bottou. Online algorithms and stochastic approximations. In *Online Learning and Neural Networks*. Cambridge University Press, 1998.

[6] K. Chadha, G. Cheng, and J. Duchi. Accelerated, optimal and parallel: Some results on model-based stochastic optimization. In *International Conference on Machine Learning*, pages 2811–2827. PMLR, 2022.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[8] J. Chen, C. Wolfe, Z. Li, and A. Kyrillidis. Demon: Improved neural network training with momentum decay. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3958–3962, 2022.

[9] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36, 2024.

[10] D. Davis and D. Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

[11] A. Defazio. Momentum via primal averaging: theoretical insights and learning rate schedules for non-convex optimization. *arXiv preprint arXiv:2010.00406*, 2020.

[12] A. Defazio and K. Mishchenko. Learning-rate-free learning by d-adaptation. In *International Conference on Machine Learning*, pages 7449–7479. PMLR, 2023.

[13] Q. Deng and W. Gao. 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, 2021. Curran Associates Inc.

[14] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[15] P. Giselsson and S. Boyd. Monotonicity and restart in fast gradient methods. In *53rd IEEE Conference on Decision and Control*, pages 5058–5063. IEEE, 2014.

[16] I. Gitman, H. Lang, P. Zhang, and L. Xiao. Understanding the role of momentum in stochastic gradient methods. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[18] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[19] A. Hägele, E. Bakouch, A. Kosson, L. Von Werra, M. Jaggi, et al. Scaling laws and compute-optimal training beyond fixed training durations. *Advances in Neural Information Processing Systems*, 37:76232–76264, 2024.

[20] W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

[23] C. Hu, W. Pan, and J. Kwok. Accelerated gradient methods for stochastic optimization and online learning. *Advances in Neural Information Processing Systems*, 22, 2009.

[24] M. Ivgi, O. Hinder, and Y. Carmon. Dog is sgd's best friend: A parameter-free dynamic step size schedule. In *International Conference on Machine Learning*, pages 14465–14499. PMLR, 2023.

[25] S. Jelassi and Y. Li. Towards understanding how momentum improves generalization in deep learning. In *International Conference on Machine Learning*, pages 9965–10040. PMLR, 2022.

[26] K. Jordan, Y. Jin, V. Boza, J. You, F. Cesista, L. Newhouse, and J. Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[27] D. S. Kalra and M. Barkeshli. Why warmup the learning rate? underlying mechanisms and improvements. *Advances in Neural Information Processing Systems*, 37:111760–111801, 2024.

[28] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[29] J. E. Kelley, Jr. The cutting-plane method for solving convex programs. *Journal of the society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[30] J. L. Kim, P. Toulis, and A. Kyrillidis. Convergence and stability of the stochastic proximal point algorithm with momentum. In *Learning for Dynamics and Control Conference*, pages 1034–1047. PMLR, 2022.

[31] D. P. Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] K. C. Kiwiel. An aggregate subgradient method for nonsmooth convex minimization. *Mathematical Programming*, 27:320–341, 1983.

[33] K. C. Kiwiel. *Methods of descent for nondifferentiable optimization*, volume 1133. Springer, 2006.

[34] A. Kosson, B. Messmer, and M. Jaggi. Analyzing & reducing the need for learning rate warmup in gpt training. *Advances in Neural Information Processing Systems*, 37:2914–2942, 2024.

[35] H. Liu, Z. Li, D. L. W. Hall, P. Liang, and T. Ma. 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, 2024.

[36] J. Liu, J. Su, X. Yao, Z. Jiang, G. Lai, Y. Du, Y. Qin, W. Xu, E. Lu, J. Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025.

[37] Y. Liu, Y. Gao, and W. Yin. An improved analysis of stochastic gradient descent with momentum. *Advances in Neural Information Processing Systems*, 33:18261–18271, 2020.

[38] N. Loizou and P. Richtárik. Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *Computational Optimization and Applications*, 77(3):653–710, 2020.

[39] N. Loizou, S. Vaswani, I. H. Laradji, and S. Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. In *International Conference on Artificial Intelligence and Statistics*, pages 1306–1314. PMLR, 2021.

[40] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2017.

[41] I. Loshchilov and F. Hutter. 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, 2017.

[42] V. Mai and M. Johansson. Convergence of a stochastic gradient method with momentum for non-smooth non-convex optimization. In *International conference on machine learning*, pages 6630–6639. PMLR, 2020.

[43] Y. Malitsky and K. Mishchenko. Adaptive gradient descent without descent. *arXiv preprint arXiv:1910.09529*, 2019.

[44] Y. Nesterov. A method for solving the convex programming problem with convergence rate o (1/k2). In *Dokl akad nauk Sssr*, volume 269, page 543, 1983.

[45] D. Oikonomou and N. Loizou. Stochastic polyak step-sizes and momentum: Convergence guarantees and practical performance. *arXiv preprint arXiv:2406.04142*, 2024.

[46] A. Orvieto, S. Lacoste-Julien, and N. Loizou. Dynamics of sgd with stochastic polyak stepsizes: Truly adaptive variants and convergence to exact solution. *Advances in Neural Information Processing Systems*, 35:26943–26954, 2022.

[47] B. O'donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15:715–732, 2015.

[48] M. Pagliardini, P. Ablin, and D. Grangier. The ademamix optimizer: Better, faster, older. In *The Twelfth International Conference on Learning Representations, ICLR 2025*, 2025.

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

[50] B. T. Polyak. Introduction to optimization. 1987.

[51] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[52] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[53] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

[54] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[55] A. Ruszczyński. A linearization method for nonsmooth stochastic programming problems. *Mathematics of Operations Research*, 12(1):32–49, 1987.

[56] S. Saab Jr, S. Phoha, M. Zhu, and A. Ray. An adaptive polyak heavy-ball method. *Machine Learning*, 111(9):3245–3277, 2022.

[57] F. Schaipp, R. Ohana, M. Eickenberg, A. Defazio, and R. M. Gower. Momo: Momentum models for adaptive learning rates. *arXiv preprint arXiv:2305.07583*, 2023.

[58] O. Sebbouh, R. M. Gower, and A. Defazio. Almost sure convergence rates for stochastic gradient descent and stochastic heavy ball. In *Conference on Learning Theory*, pages 3935–3971. PMLR, 2021.

[59] N. Shazeer and M. Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR, 2018.

[60] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[61] L. N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[62] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[63] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023.

[64] P. Tseng. An incremental gradient (-projection) method with momentum term and adaptive stepsize rule. *SIAM Journal on Optimization*, 8(2):506–531, 1998.

[65] N. Vyas, D. Morwani, R. Zhao, M. Kwun, I. Shapira, D. Brandfonbrener, L. Janson, and S. Kakade. Soap: Improving and stabilizing shampoo using adam. In *The Twelfth International Conference on Learning Representations, ICLR 2025*, 2025.

[66] B. Wang, T. Nguyen, T. Sun, A. L. Bertozzi, R. G. Baraniuk, and S. J. Osher. Scheduled restart momentum for accelerated stochastic gradient descent. *SIAM Journal on Imaging Sciences*, 15(2):738–761, 2022.

[67] X. Wang, M. Johansson, and T. Zhang. Generalized polyak step size for first order optimization with momentum. In *International Conference on Machine Learning*, pages 35836–35863. PMLR, 2023.

[68] M. Wortsman, P. J. Liu, L. Xiao, K. Everett, A. Alemi, B. Adlam, J. D. Co-Reyes, I. Gur, A. Kumar, R. Novak, et al. Small-scale proxies for large-scale transformer training instabilities. In *The Twelfth International Conference on Learning Representations, ICLR 2024*, 2024.

[69] Y. Yan, T. Yang, Z. Li, Q. Lin, and Y. Yang. A unified analysis of stochastic momentum methods for deep learning. *arXiv preprint arXiv:1808.10396*, 2018.

[70] Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C.-J. Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

[71] H. Yuan, Y. Liu, S. Wu, X. Zhou, and Q. Gu. Mars: Unleashing the power of variance reduction for training large models. *arXiv preprint arXiv:2411.10438*, 2024.

[72] S. Zagoruyko. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[73] Z. Zhuang, M. Liu, A. Cutkosky, and F. Orabona. Understanding adamw through proximal methods and scale-freeness. *Transactions on machine learning research*, 2022.

A Derivation of Adaptive Memory Momentum

A cutting plane model of the function

$$f_m(x) = \max\left\{f\left(x_t\right) + g^\top\left(x - x_t\right), \hat{f}\left(x_t\right) + d_t^\top\left(x - x_t\right)\right\}$$

Tthe update is defined by:

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

Where P_t is a diagonal preconditioner and $\tilde{P}_t = (\lambda P_t + I)P_t$ which is also diagonal, is defined so that the derived update is properly scaled. The model is substituted as follows. Define $\tilde{x} = x - x_t$

$$\min_{\mathbf{x}\in\mathcal{X}}\left\{f_m(x) + \frac{1}{2\eta}\|\tilde{x}\|_{\tilde{P}_t}^2 + \lambda\langle d_t, \tilde{x}\rangle_{P_t} + \frac{\mu}{2}\|x\|_{\tilde{P}_t}^2\right\} =$$

$$\min_{\tilde{x},\zeta}\left\{\zeta + \frac{1}{2\eta}\|\tilde{x}\|_{\tilde{P}_t}^2 + \lambda\langle d_t, \tilde{x}\rangle_{P_t} + \frac{\mu}{2}\|x\|_{\tilde{P}_t}^2\right\} \text{ s.t } \begin{cases}\zeta \geq f\left(x_t\right) + g^\top\tilde{x} \\ \zeta \geq \hat{f}\left(x_t\right) + d^\top\tilde{x}\end{cases} =$$

$$\min_{\tilde{x},\zeta}\max_{a_1+a_2=1,\ a_1,a_2\geq 0}\Big\{\zeta + \frac{1}{2\eta}\|\tilde{x}\|_{\tilde{P}_t}^2 + \lambda\langle d_t, \tilde{x}\rangle_{P_t} + \frac{\mu}{2}\|x\|_{\tilde{P}_t}^2 \\ -a_1(\zeta - f\left(x_t\right) - g^\top\tilde{x}) - a_2(\zeta - \hat{f}\left(x_t\right) - d^\top\tilde{x})\Big\}$$

We write the KKT conditions:

$$\frac{\partial\cdot}{\partial\tilde{x}} = 0 \Rightarrow x_{t+1} = \frac{1}{1+\mu\eta}\left(x_t - \eta\tilde{P}_t^{-1}\left(a_1 g + a_2 d + \lambda P_t d\right)\right) \tag{19}$$

$$\text{or} \quad \frac{1}{\eta}\tilde{P}_t\tilde{x} = -\left(\lambda P_t d + \mu\tilde{P}_t\tilde{x} + \mu\tilde{P}_t x_t + a_1 g + a_2 d\right) \tag{20}$$

$$\frac{\partial\cdot}{\partial\zeta} = 0 \Rightarrow a_1 + a_2 = 1 \tag{21}$$

Multiplying 20 by $\tilde{x}$ yields

$$\tilde{x} = -\frac{\eta}{1+\mu\eta}\tilde{P}_t^{-1}\left((\lambda P_t + a_2 I)d + \mu\tilde{P}_t x_t + a_1 g\right) \tag{22}$$

Solving 20 w.r.t. $\tilde{x}$ yields

$$-\frac{1}{\eta}\|\tilde{x}\|_{\tilde{P}_t}^2 = \lambda\langle d_t, \tilde{x}\rangle_{P_t} + \mu\|x\|_{\tilde{P}_t}^2 + \mu\langle x_t, x\rangle_{P_t} + a_1 g^T\tilde{x} + a_2 d^T\tilde{x} \tag{23}$$

Substituting back the KKT conditions gives

$$\max_{a_1+a_2=1,\ a_1,a_2\geq 0}\left\{-\frac{1+\mu\eta}{2\eta}\|\tilde{x}\|_{\tilde{P}_t}^2 + \frac{\mu}{2}\|x_t\|_{\tilde{P}_t}^2 + a_1 f\left(x_t\right) + a_2\hat{f}\left(x_t\right)\right\}$$

Substituting (23) gives

$$\max_{a_1+a_2=1,\ a_1,a_2\geq 0}\Big\{-\frac{\eta}{2(1+\mu\eta)}\|(\lambda P_t + a_2 I)d + \mu\tilde{P}_t x_t + a_1 g\|_{\tilde{P}_t^{-1}}^2 \\ + \frac{\mu}{2}\|x_t\|_{\tilde{P}_t}^2 + a_1 f\left(x_t\right) + a_2\hat{f}\left(x_t\right)\Big\} =$$

$$\max_{0\leq\beta\leq 1}\Big\{-\frac{\eta}{2(1+\mu\eta)}\|(\lambda P_t+\beta I)d+\mu\tilde{P}_t x_t+(1-\beta)g\|^2_{\tilde{P}_t^{-1}} + \frac{\mu}{2}\|x_t\|^2_{\tilde{P}_t}+(1-\beta)f(x_t)+\beta\hat{f}(x_t)\Big\}$$

This is a concave quadratic program and admits a unique solution which can be analytically derived. Differentiating this whole expression, setting the derivative to 0, and projecting the solution to the feasible set yields.

$$\beta_t^* = \text{Clip}_{[0,1]}\left(\frac{\frac{(1+\mu\eta)(\hat{f}(x_t)-f(x_t))}{\eta}-\langle d_t-g_t, g_t+\lambda P_t d_t\rangle_{\tilde{P}_t^{-1}}}{\|d_t-g_t\|^2_{\tilde{P}_t^{-1}}}+\frac{\mu\langle x_t^\top, d_t-g_t\rangle}{\|d_t-g_t\|^2_{\tilde{P}_t^{-1}}}\right). \tag{24}$$

- 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 19 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 present a motivating analysis that supports the use of an *overestimating hyperplane* when modeling the objective function. To simplify the analysis, we restrict our attention to *non-overshooting* Heavy-Ball momentum, the class of momentum-based optimizers that, unlike classical momentum schemes, avoid overshooting the optimum. While momentum is often associated with acceleration through controlled overshoot, we consider this subclass to isolate key behaviors more cleanly. For completeness we also provide conditions under which heavy ball momentum converges to the optimum of a strongly convex and smooth quadratic without overshooting.

Within this setting, we show that if one were to optimize the distance to the optimum at step t with respect to the momentum parameter β, the optimal choice of β would implicitly correspond to constructing a model of the function that *overestimates* its value at x_t. Although the derivation is elementary, it serves as a useful illustrative case that highlights the potential benefits of overestimating models in guiding the optimization trajectory.

B.1 Non-overshooting optimizers

Definition B.1 (Coordinate-wise Monotonic Decrease Condition)**.** Let $\{x_t\}_{t\geq 0}$ be the iterates of a (momentum-based) method, and let $x_*\in\arg\min f$. We say that they satisfy the *coordinate-wise monotonic decrease condition* if for every $t\geq 0$ and each coordinate $i=1,\dots,d$,

$$|x_{t+1}^i-x_*^i| \;\leq\; |x_t^i-x_*^i|, \quad \text{sign}\big(x_{t+1}^i-x_*^i\big)=\text{sign}\big(x_t^i-x_*^i\big)$$

For any convex, L-smooth function $f\colon \mathbb{R}^d \to \mathbb{R}$, plain gradient descent with step-size $\eta \le \frac{1}{L}$ satisfies the coordinate-wise monotonic decrease condition for all $i = 1, \dots, d$. Moreover, when applying the heavy-ball method to a μ–strongly convex quadratic, one can choose the momentum parameter β and learning rate η in the *overdamped regime* (i.e. so as to avoid oscillations) and thereby enforce the same per-coordinate monotonicity of the iterates, gradients, and momentum buffer.

We restrict our attention to the strongly convex, L-smooth quadratic

$$f(x) \;=\; \tfrac{1}{2}\, x^\top A\, x, \quad A = \operatorname{diag}(a_1, \dots, a_d),\; a_i > 0, \quad x_* = 0,$$

which decouples across coordinates and permits fully quantitative analysis of momentum iterations. Under the coordinate-wise monotonic decrease condition, we obtain the following sign–preservation property.

Corollary B.2 (Sign preservation)**.** *Optimizing a strongly convex, L-smooth quadratic, under the* coordinate-wise monotonic decrease condition*, for every $t \ge 0$ and each coordinate $i = 1, \dots, d$ we have,*

$$\operatorname{sign}\big(x_t^i\big) = \operatorname{sign}\big(x_0^i\big), \quad \operatorname{sign}\big(g_t^i\big) = \operatorname{sign}\big(g_0^i\big), \quad \operatorname{sign}\big(d_t^i\big) = \operatorname{sign}\big(d_0^i\big).$$

Proof. We proceed by induction on t. At $t = 0$, since $d_0^i = g_0^i = a_i\, x_0^i$, all three quantities share the same sign $\operatorname{sign}(x_0^i)$.

Now assume for some $t \ge 0$ that

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

By the coordinate-wise monotonic decrease condition on the iterates, $\operatorname{sign}(x_{t+1}^i) = \operatorname{sign}(x_0^i)$. Since

$$g_{t+1}^i = a_i\, x_{t+1}^i,$$

we also get $\operatorname{sign}(g_{t+1}^i) = \operatorname{sign}(x_{t+1}^i) = \operatorname{sign}(x_0^i)$.

Finally, the heavy-ball update

$$d_{t+1}^i \;=\; \beta\, d_t^i \;+\; (1-\beta)\, g_t^i$$

is a convex combination of d_t^i and g_t^i, both of which by the induction hypothesis have sign $\operatorname{sign}(x_0^i)$. Therefore

$$\operatorname{sign}(d_{t+1}^i) = \operatorname{sign}(x_0^i).$$

□

Proposition B.3 (No overshoot dynamics of HB on a diagonal quadratic)**.** *Let*

$$f(x) = \tfrac{1}{2}\, x^\top A x, \qquad A = \operatorname{diag}(a_1, \dots, a_d),$$

where $0 < a_d \le \cdots \le a_1 = L$. Consider the heavy-ball iteration

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

with momentum parameter $\beta \in [0, 1)$ and learning rate $\eta > 0$. Then, if the step size satisfies

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

it holds that for each coordinate $i = 1, \dots, d$,

$$|x_{t+1}^i| \le |x_t^i| \quad \textit{and} \quad \operatorname{sign}(x_{t+1}^i) = \operatorname{sign}(x_t^i) = \operatorname{sign}(x_0^i),$$

i.e., the magnitude of the iterates decreases monotonically without overshooting.

Proof. Fix a coordinate i and write $a := a_i > 0$. The scalar recurrence becomes:

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

Defining the state vector $z_t = \begin{pmatrix} d_t \\ x_t \end{pmatrix}$ allows the update to be written as a linear system:

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

Let $r_1 > r_2 > 0$ be the (real) eigenvalues of M. These satisfy:

$$r_{1,2} = \frac{\tau \pm \sqrt{\tau^2 - 4\beta}}{2}, \qquad \tau = \beta + 1 - \eta(1-\beta)a.$$

Under the step size condition $\eta a \leq \frac{1-\sqrt{\beta}}{1+\sqrt{\beta}}$, it follows that the discriminant is nonnegative and both eigenvalues satisfy $0 < r_2 < r_1 < 1$; i.e., the system is overdamped.

From the eigen-decomposition of the coordinate-i update matrix M, one finds

$$x_t \;=\; c_1\, r_1^t \;+\; c_2\, r_2^t,$$

where the eigenvectors are

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

Imposing the initial condition $(d_0, x_0)^\top = (a\,x_0, x_0)^\top = c_1\,u_1 + c_2\,u_2$ gives

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

Since $0 < r_2 < r_1 < 1$ and $\beta = r_1 r_2$, we have $\beta - r_1 < \beta - r_2 < 0$ and $\beta < r_2 < r_1$, hence $\theta_2 < \theta_1 < 0$. Moreover

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

It follows that $\mathrm{sign}(c_1) = \mathrm{sign}(x_0)$ and $c_2 < 0$. Moreover,

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

which implies $c_1 > |c_2|$. Hence, for $x_0 > 0$,

$$x_t = r_1^t\Big(c_1 + c_2\,(r_2/r_1)^t\Big) = r_1^t\Big(c_1 - |c_2|\,(r_2/r_1)^t\Big) > r_1^t\,c_1\Big(1 - (r_2/r_1)^t\Big) \geq 0.$$

Thus, x_t remains nonnegative for all t and converges monotonically to 0 without overshooting.

□

B.2 Optimal one-step momentum

In the context of convex minimization one may instead the *learning rate*

$$x_{t+1} = x_t - \eta_t\, g_t$$

by minimizing the one-step distance to the optimum $J(\eta_t) = \big\|x_{t+1}(\eta_t) - x^*\big\|^2$ and then using convexity, $g_t^\top (x_t - x^*) \geq f(x_t) - f(x^*)$, to eliminate the unknown inner product. This yields the classic *Polyak step size*

$$\eta_t = \frac{f(x_t) - f(x^*)}{\|g_t\|^2}\,.$$

In a simillar manner we can obtain dynamic momentum parameters.

Proposition B.4 (Optimal one-step momentum parameter)**.** *Let $x^* \in \mathbb{R}^d$ be the (unknown) minimizer of f. At iteration t of the heavy-ball method, suppose for $\beta_t \in [0,1]$ we have*

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

Then the choice

$$\beta_t^* = \arg\min_{\beta \in \mathbb{R}} \big\|x_{t+1}(\beta_t) - x^*\big\|^2$$

is given in closed-form by

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

Proof. Define the quadratic objective

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

Set

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

so that

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

Thus

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

Differentiating with respect to β_t and setting to zero,

$$\frac{dJ}{d\beta_t} = -2\eta\, u^\top v + 2\beta_t\, \eta^2\, \|v\|^2 \implies \beta_t^* = \frac{u^\top v}{\eta\, \|v\|^2}.$$

Re-substituting $u = (x_t - x^*) - \eta\, g_t$ and $v = d_t - g_t$ and projecting to the feasible set gives

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

□

Note that our model based β for $\lambda = 0$ 24, reduces to the same formula, except $\frac{(d-g)^\top (x_t - x^*)}{\eta}$ is replaced with $\frac{\hat{f}(x_t) - f(x_t)}{\eta}$. in our case. To justify the introduction of an "overestimating hyperplane" (i.e. $\hat{f}(x_t) > f(x_t)$), we instead analyze the quantity $(d-g)^\top (x_t - x^*)$. Indeed, proving that the one-step optimal β_t satisfies $(d-g)^\top (x_t - x^*) > 0$ provides a direct validation of our choice.

Lemma B.5. *Under the Coordinate-wise Monotonic Decrease Condition for the heavy-ball method applied to the diagonal quadratic*

$$f(x) = \tfrac{1}{2}\, x^\top A x, \quad A = \mathrm{diag}(a_1, \ldots, a_d),\ a_i > 0,$$

with minimizer $x_* = 0$ (hence $x_*^i = 0$), it holds for every $t \geq 0$ and each coordinate $i \in \{1, \ldots, d\}$ that

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

Proof. We argue by induction on t.

Base case ($t = 0$). Since $d_0^i = g_0^i = a_i\, x_0^i$,

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

Inductive step. Assume $(d_t^i - g_t^i)\, x_t^i \geq 0$ and that coordinate-wise monotonicity holds, so

$$\mathrm{sign}(x_{t+1}^i) = \mathrm{sign}(x_t^i), \quad \mathrm{sign}(g_{t+1}^i) = \mathrm{sign}(g_t^i), \quad \mathrm{sign}(d_{t+1}^i) = \mathrm{sign}(d_t^i).$$

Recall

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

A short calculation gives

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

where $\beta \in [0, 1)$ and $\eta > 0$. By the induction hypothesis, $(d_t^i - g_t^i)$ has the same sign as x_t^i, and $g_t^i = a_i x_t^i$ also shares that sign. Hence each term on the right has sign $\mathrm{sign}(x_t^i)$, and thus

$$\mathrm{sign}\big(d_{t+1}^i - g_{t+1}^i\big) = \mathrm{sign}(x_t^i) = \mathrm{sign}(x_{t+1}^i).$$

Therefore

$$(d_{t+1}^i - g_{t+1}^i)\, x_{t+1}^i \ =\ \left|d_{t+1}^i - g_{t+1}^i\right| \left|x_{t+1}^i\right| \ \geq\ 0,$$

completing the induction. □

C Proofs of section 3.5

We provide convergence guarantees for our proposed methods 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 on a subset i of training samples , 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 $x, y \in \mathbb{R}^d$:

$$f(y) \le f(x) + \nabla f(x)^\top (y-x) + \frac{L}{2}\|y-x\|^2 \tag{25}$$

Assumption C.2 (Bounded Stochastic Gradient Norm)**.** There exists a G such that :

$$\mathbb{E}[\|\nabla f_{S_i}(x)\|_2^2] \le G^2 \tag{26}$$

Algorithm: We prove convergence for a simplified version of AM-SGD-M with $\lambda = 0$. Let $\eta > 0$, $0 \le \beta_t \le \beta_{\max} < 1$ be the learning rate, and initialize $d_0 = 0$. The updates are:

$$\begin{aligned}
\beta_t &= \operatorname*{Clip}_{[0,\beta_{\max}]}\Big(\frac{\|\nabla f_{S_t}\|_2^2}{\|d_t - \nabla f_{S_t}\|_2^2}\Big)\\
d_{t+1} &= \beta_t\, d_t \;+\; (1-\beta_t)\, g_t,\\
x_{t+1} &= x_t \;-\; \eta\, d_{t+1}.
\end{aligned}$$

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

Lemma C.3. *Consider the momentum iterates of C and assume that*

$$\mathbb{E}\big[\|\nabla f_{S_t}(x_t)\|_2^2\big] \;\le\; G^2.$$

Then, for every t,

$$\mathbb{E}\big[\|d_{t+1}\|_2^2\big] \;\le\; 2G^2.$$

Proof. We begin by expanding the definition of d_{t+1}:

$$d_{t+1} \;=\; \nabla f_{S_t}(x_t) + \beta_t\big(d_t - \nabla f_{S_t}(x_t)\big), \quad \text{where} \quad \beta_t \in [0,1).$$

Taking the squared norm and applying the triangle inequality,

$$\|d_{t+1}\|_2^2 \;=\; \big\|\nabla f_{S_t}(x_t) + \beta_t(d_t - \nabla f_{S_t}(x_t))\big\|_2^2 \;\le\; \|\nabla f_{S_t}(x_t)\|_2^2 + \beta_t^2\|d_t - \nabla f_{S_t}(x_t)\|_2^2.$$

Since $\beta_t < 1$, we can bound this more simply as

$$\|d_{t+1}\|_2^2 \;\le\; \|\nabla f_{S_t}(x_t)\|_2^2 + \beta_t\|d_t - \nabla f_{S_t}(x_t)\|_2^2.$$

Now, consider the adaptive choice

$$\beta_t \;=\; \operatorname*{Clip}_{[0,\,\beta_{\max}]}\left(\frac{\|\nabla f_{S_t}(x_t)\|_2^2}{\|d_t - \nabla f_{S_t}(x_t)\|_2^2}\right).$$

Under this definition, the second term is upper bounded by $\|\nabla f_{S_t}(x_t)\|_2^2$, leading to

$$\|d_{t+1}\|_2^2 \;\le\; 2\,\|\nabla f_{S_t}(x_t)\|_2^2.$$

Taking expectation and invoking the bounded gradient assumption $\mathbb{E}[\|\nabla f_{S_t}(x_t)\|_2^2] \le G^2$, we obtain

$$\mathbb{E}\big[\|d_{t+1}\|_2^2\big] \;\le\; 2G^2.$$

□

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 $d_t - \nabla f_{S_t}(x_t)$ to the stochastic gradient itself, and on Young's inequality to separate mixed inner products.

Lemma C.4. *Consider the momentum iterates of C, the following inequality holds for any $a > 0$:*

$$-2\eta\,\beta_t\,\langle x_t - x^\star,\, d_t - \nabla f_{S_t}(x_t)\rangle \;\le\; \beta_t^{\max} a\,\|x_t - x^\star\|_2^2 + \frac{\eta^2}{a}\,\|\nabla f_{S_t}(x_t)\|_2^2.$$

Proof. We begin by applying Young's inequality:

$$-2\eta\,\beta_t\,\langle x_t - x^\star,\, d_t - \nabla f_{S_t}(x_t)\rangle \;\le\; 2\eta\,\beta_t\,\|x_t - x^\star\|_2\,\|d_t - \nabla f_{S_t}(x_t)\|_2.$$

Using $2xy \le x^2/\alpha + \alpha y^2$ with $\alpha > 0$, we obtain

$$-2\eta\,\beta_t\,\langle x_t - x^\star,\, d_t - \nabla f_{S_t}(x_t)\rangle \;\le\; \beta_t\,a\,\|x_t - x^\star\|_2^2 + \frac{\eta^2\beta_t}{a}\,\|d_t - \nabla f_{S_t}(x_t)\|_2^2.$$

By the definition of β_t and the boundedness condition $\beta_t \le \beta_t^{\max} < 1$, we further bound $\|d_t - \nabla f_{S_t}(x_t)\|_2^2$ by $\|\nabla f_{S_t}(x_t)\|_2^2$, yielding

$$-2\eta\,\beta_t\,\langle x_t - x^\star,\, d_t - \nabla f_{S_t}(x_t)\rangle \;\le\; \beta_t^{\max}\,a\,\|x_t - x^\star\|_2^2 + \frac{\eta^2}{a}\,\|\nabla f_{S_t}(x_t)\|_2^2.$$

□

Lemma C.5. *For the momentum method, the following inequality holds for any $a_t > 0$:*

$$-\eta\,\beta_t\,\langle \nabla f(x_t),\, d_t - \nabla f_{S_t}(x_t)\rangle \;\le\; \beta_t^{\max}\,\frac{1}{2a_t}\|\nabla f(x_t)\|_2^2 + \frac{a_t\,\eta^2}{2}\,\|\nabla f_{S_t}(x_t)\|_2^2.$$

Proof. We again apply Young's inequality to the mixed term:

$$-\eta\,\beta_t\,\langle \nabla f(x_t),\, d_t - \nabla f_{S_t}(x_t)\rangle \;\le\; \eta\,\beta_t\,\|\nabla f(x_t)\|_2\,\|d_t - \nabla f_{S_t}(x_t)\|_2.$$

By Young's inequality, for any $a_t > 0$,

$$\eta\,\beta_t\,\|\nabla f(x_t)\|_2\,\|d_t - \nabla f_{S_t}(x_t)\|_2 \;\le\; \frac{\beta_t}{2a_t}\|\nabla f(x_t)\|_2^2 + \frac{a_t\eta^2\beta_t}{2}\,\|d_t - \nabla f_{S_t}(x_t)\|_2^2.$$

Bounding $\beta_t \le \beta_t^{\max}$ and $\|d_t - \nabla f_{S_t}(x_t)\|_2^2 \le \|\nabla f_{S_t}(x_t)\|_2^2$, we obtain the desired inequality:

$$-\eta\,\beta_t\,\langle \nabla f(x_t),\, d_t - \nabla f_{S_t}(x_t)\rangle \;\le\; \beta_t^{\max}\,\frac{1}{2a_t}\|\nabla f(x_t)\|_2^2 + \frac{a_t\,\eta^2}{2}\,\|\nabla f_{S_t}(x_t)\|_2^2.$$

□

Theorem C.6 (Convex Convergence)**.** *Let $f_i : \mathbb{R}^d \to \mathbb{R}$ convex, with stochastic gradients satisfying Assumption 3.2. Let $x^* \in \arg\min f > -\infty$ and define the averaged iterate $\bar{x}_T = \frac{1}{T}\sum_{t=1}^{T} x_t$. Then, for our method with $\beta_{\max} = 1/T$ and $\eta = 1/\sqrt{T}$ we have:*

$$\mathbb{E}\big[f(\bar{x}_T) - f(x^*)\big] \le \frac{1}{\sqrt{T}}\left(\frac{1}{2}\|x_0 - x^*\|^2 + G^2\right). \tag{27}$$

Proof.

$$\begin{aligned}\|x_{t+1} - x^\star\|_2^2 &= \|x_t - x^\star\|_2^2 - 2\eta\,\langle x_t - x^\star,\, d_{t+1}\rangle + \eta^2\|d_{t+1}\|_2^2 \\ &= \|x_t - x^\star\|_2^2 - 2\eta\,\langle x_t - x^\star,\, \nabla f_{S_t}(x_t)\rangle - 2\eta\,\beta_t\,\langle x_t - x^\star,\, d_t - \nabla f_{S_t}(x_t)\rangle + \eta^2\|d_{t+1}\|_2^2.\end{aligned}$$

By convexity of f,

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

Applying Lemma C.4 with $a = 1$ to the cross term $-2\eta\,\beta_t\,\langle x_t - x^\star,\, d_t - \nabla f_{S_t}(x_t)\rangle$, we obtain

$$\|x_{t+1} - x^\star\|_2^2 \le (1 + \beta_t^{\max})\|x_t - x^\star\|_2^2 - 2\eta\big(f_{S_t}(x_t) - f_{S_t}(x^\star)\big) + \eta^2\|\nabla f_{S_t}(x_t)\|_2^2 + \eta^2\|d_{t+1}\|_2^2.$$

Taking expectations and applying Lemma C.3 and the bounded gradient norm assumption, we obtain

$$\mathbb{E}[\|x_{t+1} - x^\star\|_2^2] \le (1 + \beta_t^{\max})\mathbb{E}[\|x_t - x^\star\|_2^2] - 2\eta\big(f(x_t) - f(x^\star)\big) + 2\eta^2 G^2.$$

Define the scaling sequence $\{a_t\}$ re cursively as

$$a_{t-1} = (1 + \beta_t^{\max})a_t, \quad a_{-1} = 1,$$

and multiply both sides of the above inequality by a_t:

$$a_t\,\mathbb{E}[\|x_{t+1} - x^\star\|_2^2] \le a_{t-1}\,\mathbb{E}[\|x_t - x^\star\|_2^2] - 2\eta\,a_t\,\mathbb{E}[f(x_t) - f(x^\star)] + 2\eta^2\,a_t\,G^2.$$

Rearranging and telescoping the inequality from $t = 0$ to $T - 1$, we obtain

$$2\eta \sum_{t=0}^{T-1} a_t\,\mathbb{E}[f(x_t) - f(x^\star)] \le a_{-1}\|x_0 - x^\star\|_2^2 - a_{T-1}\,\mathbb{E}[\|x_T - x^\star\|_2^2] + 2\eta^2 G^2 \sum_{t=0}^{T-1} a_t.$$

Since all terms are nonnegative, we can drop the last negative term to get

$$2\eta \sum_{t=0}^{T-1} a_t\,\mathbb{E}[f(x_t) - f(x^\star)] \le \|x_0 - x^\star\|_2^2 + 2\eta^2 G^2 \sum_{t=0}^{T-1} a_t.$$

Dividing both sides by $2\eta \sum_{t=0}^{T-1} a_t$ and applying Jensen's inequality (by convexity of f), we obtain

$$\mathbb{E}\big[f(\bar{x}_T) - f(x^\star)\big] \le \mathbb{E}[\sum_{t=0}^{T-1} \frac{a_t}{\sum_{t=0}^{T-1} a_t}(f(x_t) - f(x^\star))] \le \frac{\|x_0 - x^\star\|_2^2}{2\eta \sum_{t=0}^{T-1} a_t} + \eta\,G^2.$$

Since $\beta_t^{\max} = 1/T$, we have

$$a_{t-1} = \big(1 + \tfrac{1}{T}\big)a_t \quad \Rightarrow \quad a_t = \big(1 + \tfrac{1}{T}\big)^{T-t}.$$

Hence

$$\sum_{t=0}^{T-1} a_t = \sum_{k=1}^{T} \big(1 + \tfrac{1}{T}\big)^k \ge T,$$

since $\big(1 + \frac{1}{T}\big)^k \ge 1$ for all k. Using this lower bound and substituting $\eta = 1/\sqrt{T}$, we obtain

$$\mathbb{E}\big[f(\bar{x}_T) - f(x^\star)\big] \le \frac{\|x_0 - x^\star\|_2^2}{2\eta T} + \eta\,G^2 \le \frac{1}{\sqrt{T}}\left(\frac{1}{2}\|x_0 - x^\star\|_2^2 + G^2\right).$$

□

Theorem C.7 (Non-convex Convergence)**.** *Let f be L-smooth , with stochastic gradients satisfying Assumption 3.2. Then, for our method with $\eta = 1/\sqrt{T}$ and $\beta_{\max} = c\eta$ with $0 < c < 1$, we have:*

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

Proof. By L-smoothness of f, we have

$$\begin{aligned} f(x_{t+1}) &\le f(x_t) + \langle \nabla f(x_t),\, x_{t+1} - x_t\rangle + \frac{L}{2}\|x_{t+1} - x_t\|_2^2 \\ &\le f(x_t) - \eta\,\langle \nabla f(x_t),\, d_{t+1}\rangle + \frac{L\eta^2}{2}\|d_{t+1}\|_2^2 \\ &\le f(x_t) - \eta\,\langle \nabla f(x_t),\, \nabla f_{S_t}(x_t)\rangle - \eta\,\langle \nabla f(x_t),\, d_t - \nabla f_{S_t}(x_t)\rangle + \frac{L\eta^2}{2}\|d_{t+1}\|_2^2 \end{aligned}$$

Applying Lemma C.5 to the second inner product, we obtain

$$\begin{aligned} f(x_{t+1}) \le f(x_t) - \eta\,\langle \nabla f(x_t),\, \nabla f_{S_t}(x_t)\rangle + \frac{\beta_t^{\max}}{2a_t}\|\nabla f(x_t)\|_2^2 \\ + \frac{a_t\eta^2}{2}\|\nabla f_{S_t}(x_t)\|_2^2 + \frac{L}{2}\eta^2\|d_{t+1}\|_2^2. \end{aligned}$$

Taking expectations, rearranging and applying Lemma C.3 and the bounded gradient norm assumption, we obtain

$$\left(\eta - \frac{\beta_t^{\max}}{2a_t}\right)\mathbb{E}\big[\|\nabla f(x_t)\|_2^2\big] \leq \mathbb{E}\big[f(x_t) - f(x_{t+1})\big] + \frac{\eta^2}{2}\,G^2\,(a_t + 2L),$$

Setting $a_t = \frac{1}{2}$ and summing over t telescopes the right-hand side:

$$\begin{aligned}\sum_{t=0}^{T-1}(\eta - \beta_t^{\max})\mathbb{E}\big[\|\nabla f(x_t)\|_2^2\big] &\leq \mathbb{E}[f(x_0) - f(x_T)] + \frac{\eta^2 G^2}{4}\sum_{t=0}^{T-1}(1+4L) \\ &\leq \mathbb{E}[f(x_0) - f(x_T)] + \frac{\eta^2 G^2}{4}(1+4L)T.\end{aligned}$$

Using $\eta = 1/\sqrt{T}$ and $\beta_t^{\max} = c\eta$ with $0 < c < 1$, we have

$$\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(x_t)\|_2^2\big] \;\leq\; \frac{1}{\eta(1-c)}\left(\mathbb{E}[f(x_0) - f(x_T)] + \frac{\eta^2 G^2}{4}(1+4L)T\right).$$

Dividing both sides by T and using $\eta = 1/\sqrt{T}$, we obtain

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\|\nabla f(x_t)\|_2^2\big] \;\leq\; \frac{1}{\sqrt{T}(1-c)}\left(\mathbb{E}[f(x_0) - f(x_T)] + \frac{(1+4L)G^2}{4}\right).$$

Finally, since $\min_t \mathbb{E}[\|\nabla f(x_t)\|_2^2] \leq \frac{1}{T}\sum_t \mathbb{E}[\|\nabla f(x_t)\|_2^2]$, we conclude that

$$\min_{0\leq t\leq T-1}\mathbb{E}\big[\|\nabla f(x_t)\|_2^2\big] \;\leq\; \frac{1}{\sqrt{T}(1-c)}\left(\mathbb{E}[f(x_0) - f(x_T)] + \frac{(1+4L)G^2}{4}\right).$$

□

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: $P_t = (1 - \beta_{1\max} \prod_t \beta_{1t}) \mathrm{diag}\left(\epsilon + \sqrt{\hat{v}_t}\right)$
  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:

$$\begin{aligned} f(x_{t-1}, s_t) - f(x_t, s_t) &\approx \nabla f(x_t, s_t)^\top (x_{t-1} - x_t) \\ &= \eta_{t-1} g_t^T (P_{t-1}^{-1} d_t + \mu x_t) && \text{(From equation 22)} \\ &\approx \eta_{t-1} g_t^T (P_t^{-1} d_t + \mu x_t) && \text{(Assuming } P_{t-1}^{-1} \approx P_t^{-1}\text{)} \end{aligned}$$

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 5.

Table 1: 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 2 shows that the adaptive momentum mechanism introduces negligible overhead, below 0.5% even for small models, while consistently accelerating convergence by up to $1.5\times$ for larger models.

Table 2: 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 3: 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 4: 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 5: 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 6: Hyper-parameter settings for the LLM experiments.

Hyper-parameter	**Value**
Iterations	1000
Learning Rate	0.001
Weight Decay	0.0001
Batch Size	1024
Model Precision	BF16
Mix Precision	BF16 & TF32
Scheduler	Cosine with warm-up
Warm-up Iterations	100
Grad Clipping	1.0
β_1	0.9
β_2	0.99
ϵ	1e-8
Seq-len	256
AM-AdamW λ	0.1
$\beta_{1\,\max}$	$0.9 - 0.1\lambda$
Eval Precision	BF16
Eval Seq-len	256

Table 7: 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 ($\epsilon = 1\text{e}{-6}$)
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

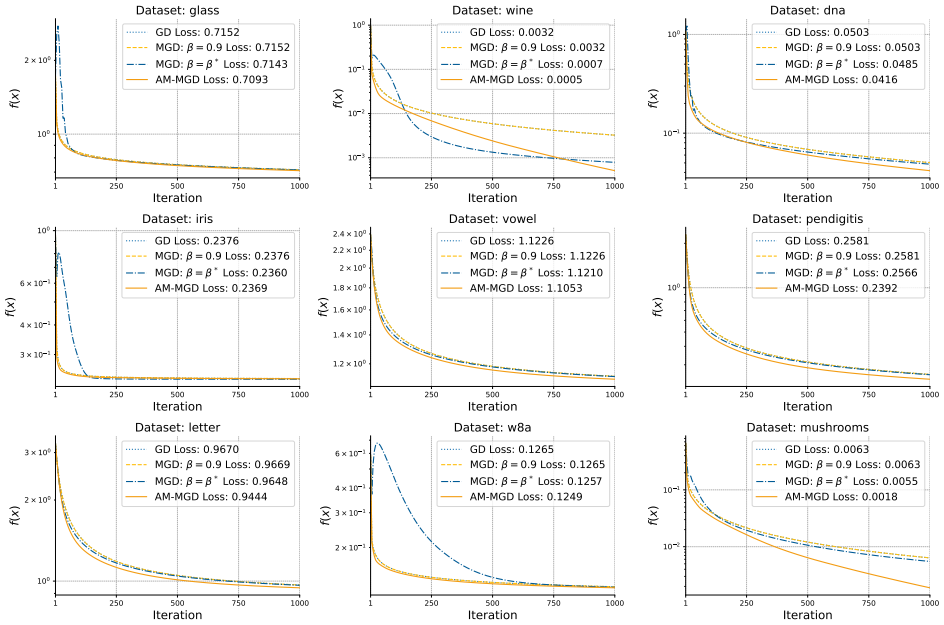

Figure 8: Logistic Loss over time, AM-MGD (red curve) outperforms fixed momentum on all but one experiments. GD and MGD with $\beta = 0.9$ practically overlap. We run all algorithms for 10000 iterations with a learning rate $\eta = 1/L$ where L is the smoothness

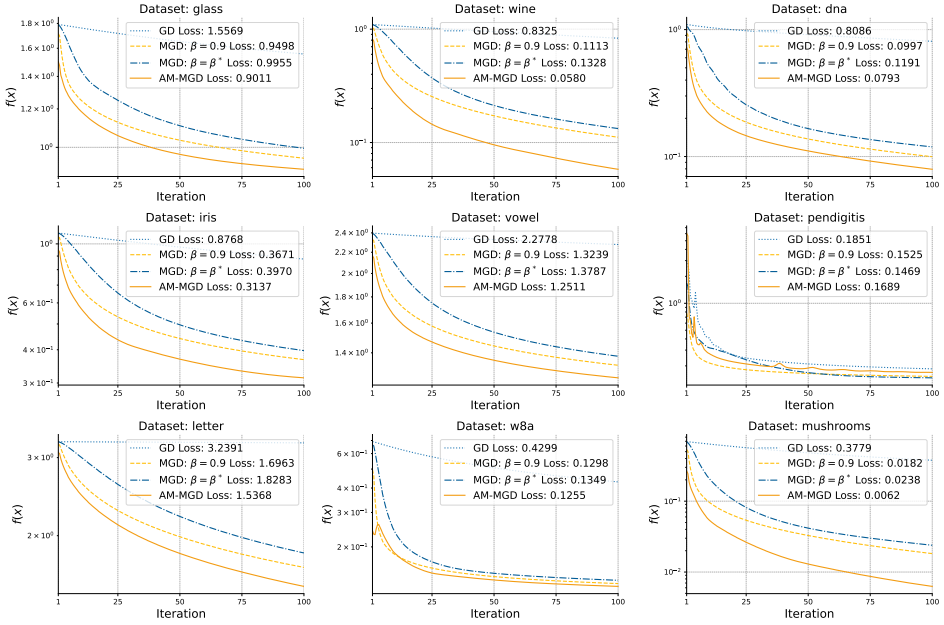

Figure 9: Logistic Loss over time, AM-Adam (purple curve) outperforms fixed momentum on all but one experiments. We run all algorithms for 1000 iterations with a learning rate of η=0.001

F.2 Detailed Curves for Figure 7c

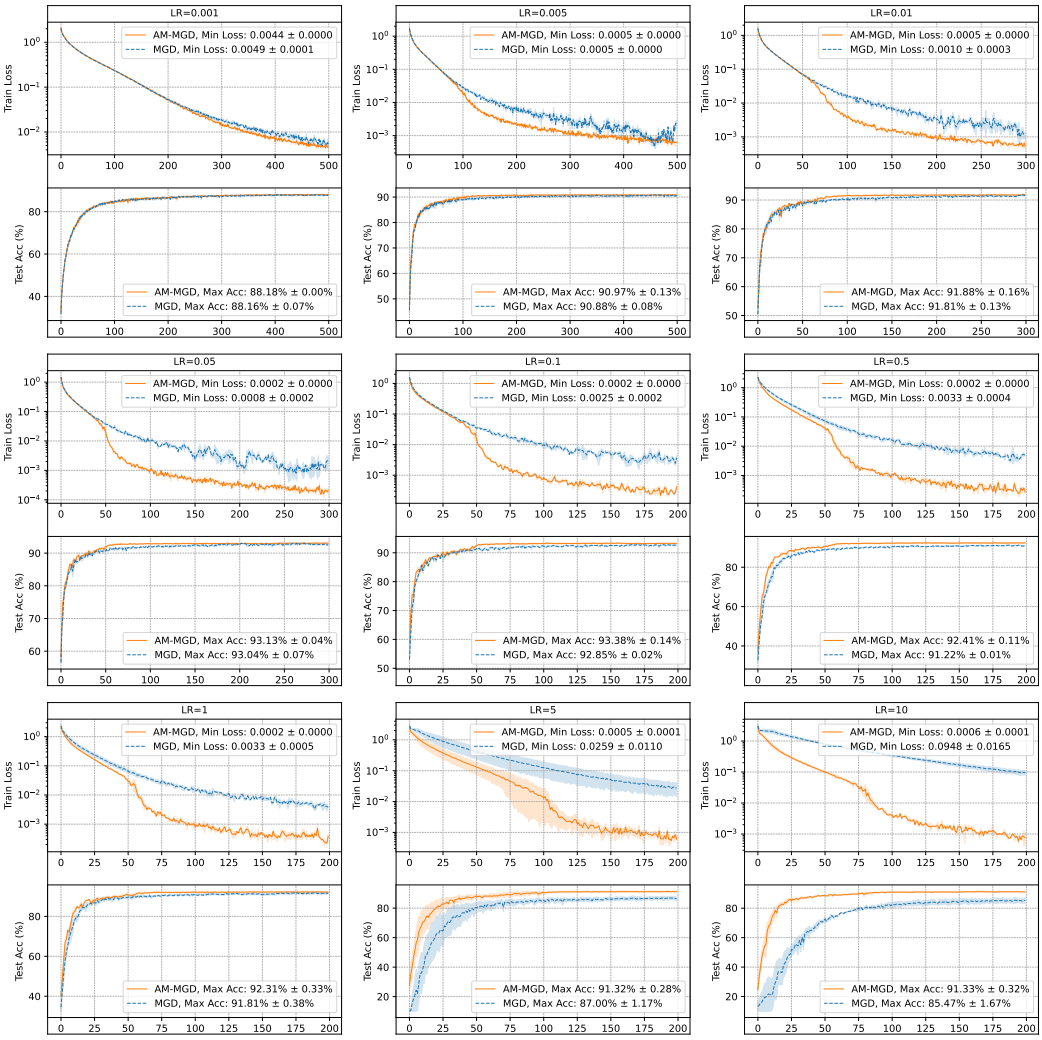

Figure 10: Training curves for different learning rates on ResNet18 trained on CIFAR10

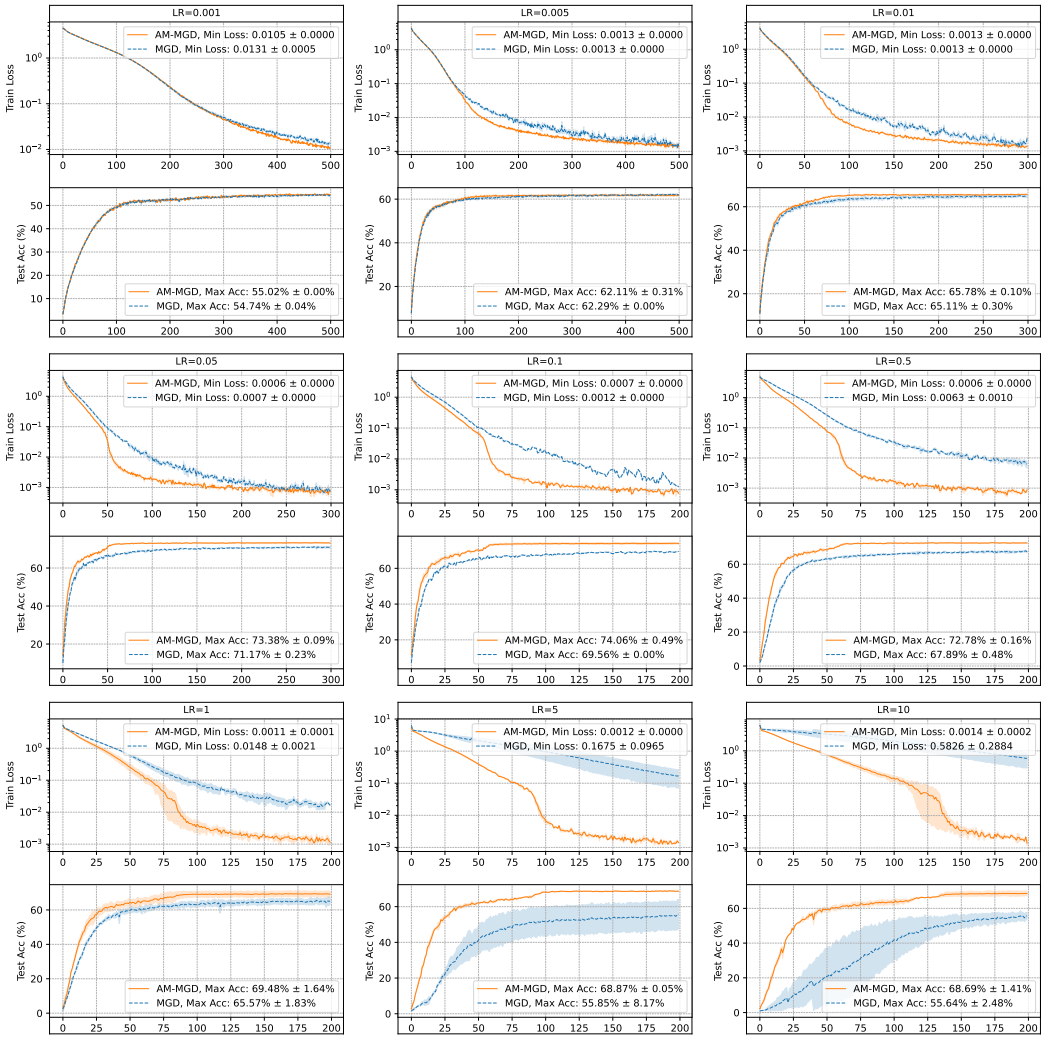

Figure 11: Training curves for different learning rates on ResNet50 trained on CIFAR100

F.3 More plots on the behaviour of β_t

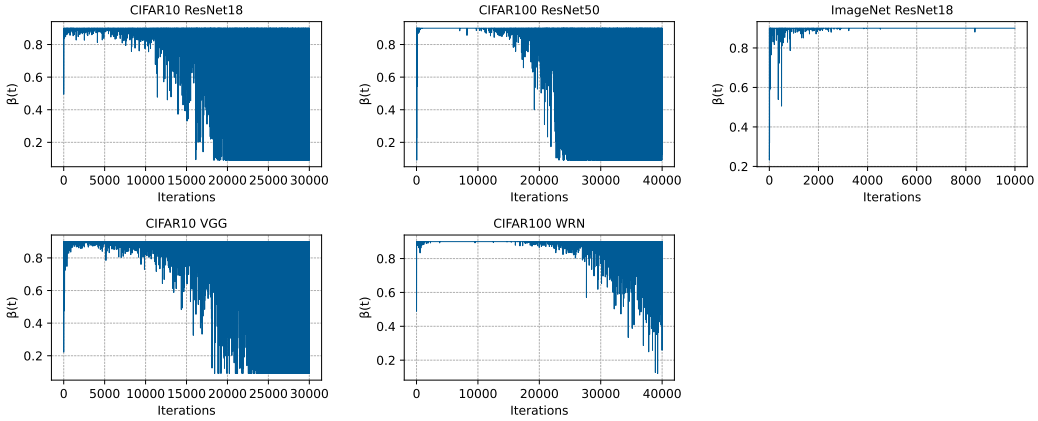

Figure 12: The adaptive β_t across different experiments

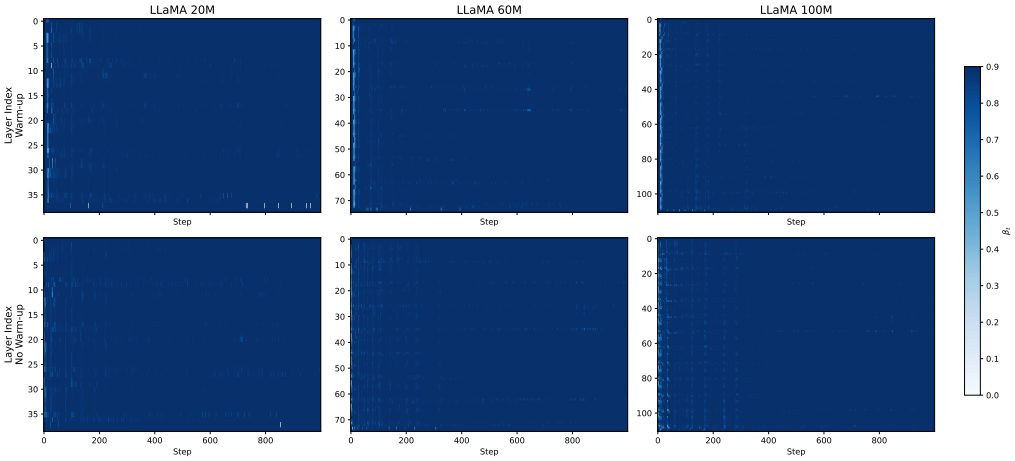

Figure 13: The layer-wise adaptive β_t across different Llama experiments

F.4 Imagenet Experiment

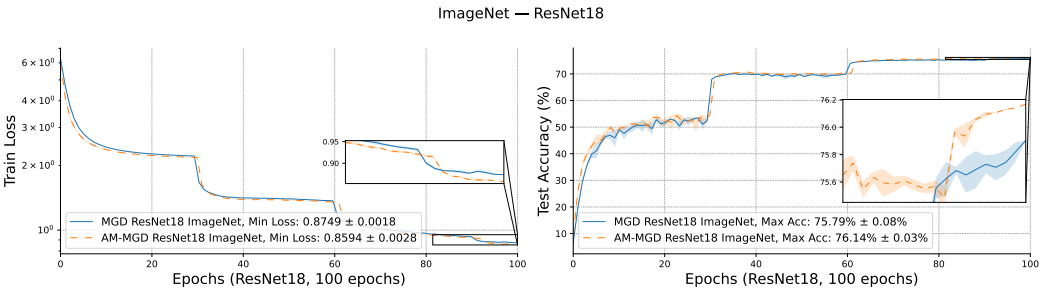

Figure 14: ResNet18 on Imagenet