AdamS: Momentum Itself Can Be A Normalizer for LLM Pretraining and Post-training

Anonymous ACL submission

Abstract

We introduce *AdamS*, a simple yet effective alternative to Adam for large language model (LLM) pretraining and post-training. By leveraging a novel denominator, i.e., the root mean square of a properly weighted momentum and the current gradient, *AdamS* eliminates the need for second-moment estimates. Hence, *AdamS* is efficient, matching the memory and compute footprint of SGD with momentum while delivering superior optimization performance. Moreover, *AdamS* is easy to adopt: it can directly inherit hyperparameters of AdamW, and is entirely model-agnostic, integrating seamlessly into existing pipelines without modifications to optimizer APIs or architectures. The motivation behind *AdamS* stems from the observed (L_0, L_1) smoothness properties in transformer objectives, where local smoothness is governed by gradient magnitudes. In this setting, momentum offers a naturally smoothed gradient estimate. We establish rigorous theoretical convergence guarantees and provide practical guidelines for hyperparameter selection. Empirically, *AdamS* demonstrates strong performance across diverse tasks and architectures, including pretraining runs on GPT-2 and Llama2 (up to 13B parameters). It also excels in reinforcement learning post-training, particularly in the DeepSeek R1-Zero replication task, underscoring its versatility across training paradigms. With its efficiency, simplicity, and theoretical grounding, *AdamS* stands as a compelling alternative to existing optimizers.

1 Introduction

Due to the scaling law (Kaplan et al., 2020) of neural networks, it has been enthusiastic in the AI community to pre-train large foundation models with enormous data over the past years (Touvron et al., 2023a; Brown et al., 2020; Zhang et al., 2022; Rae et al., 2021; Chowdhery et al., 2022; Du et al., 2021; Liu et al., 2024; Dubey et al., 2024; Yang et al., 2024). Training such large foundation models become super challenging because of tremendous engineering efforts, computational cost (Rajbhandari et al., 2019; Guo et al., 2025), and potential training spikes (Zhang et al., 2022; Molybog et al., 2023; Chowdhery et al., 2022).

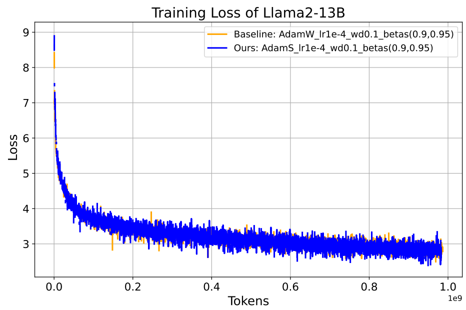

Figure 1: Training loss curves for pretraining Llama2-13B models. The proposed *AdamS* achieves convergence comparable to baseline methods—without the need to store AdamW's second-order estimates.

One reason for such high cost comes from the widely used optimizer *Adam* (Kingma and Ba, 2014) or *AdamW* (Loshchilov and Hutter, 2019): the optimizers require storing both the state of momentum and the state of second-moment estimates, which consumes $2 \sim 4$ times GPU memories of the model size, huge for models with hundreds of billions of parameters.

In this paper, we try to reduce such cost by proposing a simple yet effective optimizer *AdamS*, an alternative to AdamW. *AdamS* eliminates the need for second-moment estimates, by leveraging a novel denominator: the root mean square of a carefully weighted combination of momentum and the current gradient. As a consequence, *AdamS* matches the memory and compute footprint of stochastic gradient descent (SGD) with momentum while delivering superior performance as good as AdamW.

The design of *AdamS* is inspired by the observation that transformer-based models, which dominate modern large language models (LLMs), exhibit unique smoothness properties in their optimization landscapes. Specifically, the local smoothness of these objectives is governed by gradient magnitudes, which suggests that the learning rate should be proportional to the reciprocal of the gradient norm at each iteration, the core insight why Adam optimizer beats SGD on training transformer-like architectures (Wang et al., 2022, 2023b). We employ the fact that momentum, an exponential average of historical gradients, can provide a good and robust estimate of gradient magnitude (Cutkosky and Mehta, 2020; Zhang et al., 2020) without the need for complex second-moment computations. By leveraging this insight, *AdamS* reduces memory cost of the optimizer states by half. Such efficiency of *AdamS* is particularly attractive for large-scale training, where even small improvements in efficiency can translate into significant cost savings.

We note that there has been effort of designing new optimizers for less memory cost, i.e., Adafactor (Shazeer and Stern, 2018), Adam-mini (Zhang et al., 2024), Shampoo (Gupta et al., 2018), Lion (Chen et al., 2023), for better convergence either theoretical or practical, i.e, Sophia (Liu et al., 2023), NAdam (Dozat, 2016), AdaBound (Luo et al., 2019), AdaBelief (Zhuang et al., 2020), and RAdam (Liu et al., 2020)—Adam (Kingma and Ba, 2014) and its variant AdamW (Loshchilov and Hutter, 2019) remain the dominant choices in both academic and industrial deep learning implementations (Schneider et al., 2022). This reluctance to adopt new optimizers stems from the difficulty of systematically surpassing AdamW in large-scale learning (Kaddour et al., 2023) and the fundamental role optimizers play in training. Practitioners are hesitant to switch unless a new optimizer offers clear advantages, is easy to tune, and integrates seamlessly into existing workflows.

Recognizing this deep-rooted reliance on AdamW, we emphasize that *AdamS* is easy to adopt, and can serve as a drop-in replacement of AdamW for LLM pretraining and post-training tasks. *AdamS* is entirely model-agnostic, making it easy to integrate into existing pipelines without modifications to optimizer APIs or architectures. More importantly, it inherits AdamW's hyperparameter configuration, thereby mitigating the often prohibitive costs of hyperparameter re-tuning and minimizing the risk associated with deploying a new optimizer at scale.

Empirically, *AdamS* demonstrates strong performance across a wide range of tasks and architectures. In pretraining scenarios, it matches or exceeds the performance of AdamW on models ranging from GPT-2 to Llama2, with parameter counts up to 13B as shown in Figure 1. This scalability is particularly important given the growing trend toward even larger models and datasets. Additionally, *AdamS* excels in post-training tasks, including reinforcement learning (RL), where it achieves state-of-the-art results in tasks such as the DeepSeek R1-Zero replication. This versatility underscores its potential as a general-purpose optimizer for both pretraining and post-training paradigms.

On the theoretical side, we establish rigorous convergence guarantees that demonstrate the effectiveness of *AdamS* in optimizing non-convex objectives, which are typical in LLM training. These guarantees are derived under realistic assumptions about the smoothness and noise properties of the optimization landscape.

Our contributions can be summarized as follows:

- **Innovative Optimizer Design**: We introduce *AdamS*, which eliminates the need for second-moment estimates by leveraging a novel normalization strategy based on a weighted momentum-gradient combination. This approach reduce the memory footprint of optimizers' state by 50% while maintaining the ease of adoption.

- **Theoretical Grounding**: We rigorously analyze the convergence guarantees of *AdamS* for optimizing non-convex objectives under relaxed smoothness and weak noise assumptions, which matches the lower bounds of any gradient-based optimizers.

- **Empirical Validation**: Through extensive experiments, e.g., large-scale pretraining on models like GPT-2 and Llama2 (up to 13B parameters) and reinforcement learning post-training tasks such as DeepSeek R1-Zero replication, we demonstrate that *AdamS* consistently matches AdamW, underscoring its versatility across different training paradigms.

In the following sections, we detail the motivation and formulation of *AdamS*. We then present the theoretical analysis and convergence guarantees, followed by an extensive empirical study spanning a variety of tasks and architectures. Through this comprehensive exploration, we aim to establish *AdamS* as a compelling alternative in the evolving landscape of large language model pretraining and post-training optimization.

1.1 Related Works

The smoothness property of transformer-like architectures. The seminal work (Zhang et al., 2019) introduced the (L_0, L_1)-smooth condition that assumes local smoothness bounded by the local gradient norm, which is nicely verified by the optimization landscape of training transformer-like models. Under these assumptions, convergence properties of adaptive optimizers, AdaGrad (Faw et al., 2023; Wang et al., 2023b), Adam (Wang et al., 2022; He et al., 2023; Wang et al., 2023b; Li et al., 2023) are established and the benefit over SGD is demonstrated. Our design of *AdamS* is inspired by these local smoothness properties, and delivers robust empirical performance, where gradient magnitudes govern optimization dynamics particularly in transformer-like architectures.

Memory-efficient adaptive learning rate optimizers. In the development of memory-efficient adaptive learning rate optimizers, several notable methods have been proposed to address the challenges of high memory consumption in large-scale neural network training. Shazeer and Stern (2018) introduced Adafactor, which reduces memory usage by maintaining only per-row and per-column sums of the second-moment estimates for weight matrices. Anil et al. (2019) proposed SM3, a memory-efficient adaptive optimization method that approximates second-moment statistics with sublinear memory cost by partitioning parameters and sharing second-moment estimates among them. SM3 achieves per-parameter adaptivity with reduced memory overhead, facilitating the training of larger models and mini-batches. Luo et al. (2023) developed CAME to address the instability issues of existing memory-efficient optimizers via a confidence-guided adaptive strategy. Lv et al. (2023) introduced AdaLomo, which combines low-memory optimization techniques with adaptive learning rates by employing non-negative matrix factorization for second-order moment estimation. Zhao et al. (2024) proposed GaLore that projects weight gradients onto a low-rank subspace, and update the model in the low-rank subspace, enabling fine-tuning LLM with consumer-grade GPUs with 24GB memory, where the idea of low-rank projection has been initiated in (Yu et al., 2021). Recently, Zhang et al. (2024) proposed Adam-mini, an optimizer that reduces memory usage by partitioning model parameters into blocks based on the Hessian structure and assigning a single learning rate to each block, reducing memory consumption of optimizer state by approximately 45% to 50%.

Despite the proliferation of all these advancements, practitioners often hesitate to move away from AdamW because they either need to tune more hyperparameters, or require to be aware of the model architecture, or do not systematically surpassing AdamW in large-scale learning (Kaddour et al., 2023; Hoffmann et al., 2022). In contrast, *AdamS* offers a model-agnostic solution that seamlessly integrates into existing workflows. It requires no additional hyperparameters beyond those used in AdamW, allowing for straightforward adoption and tuning. Moreover, *AdamS* matches the memory efficiency of vanilla SGD with momentum while delivering performance comparable to AdamW, making it a practical drop-in replacement that one can enjoy benefits with minimal effort.

2 Motivation and Design Choices of *AdamS*

This section outlines the motivation behind our optimizer design—specifically, the rationale for adopting the root mean square of a properly weighted momentum itself and the current gradient as an adaptive denominator. We then formalize the algorithm and analyze its properties.

2.1 Motivation and (L_0, L_1) smoothness

In classical optimization settings, gradient descent provably decreases the loss at each iteration—provided the learning rate is smaller than the inverse of the smoothness constant. However, this principle fails to hold for transformer-based models, where stochastic gradient descent (SGD) with momentum exhibits poor convergence empirically. Recent work by (Zhang et al., 2019) identifies a key observation: Transformer training objectives violate standard smoothness assumptions and instead obey a relaxed (L_0, L_1)-smoothness condition. Under this regime, the local smoothness depends on the gradient magnitude, enabling pathological curvature that can arbitrarily slow SGD's progress (Wang et al., 2023a). The (L_0, L_1)-smoothness assumption is as follows.

Assumption 2.1 ((L_0, L_1)-smooth condition)**.** Assuming that f is differentiable and lower bounded, there exist constants $L_0, L_1 > 0$, such that $\forall \boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^d$ satisfying $\|\boldsymbol{w}_1 - \boldsymbol{w}_2\| \leq \frac{1}{L_1}$,

$$\begin{aligned} &\|\nabla f(\boldsymbol{w}_1) - \nabla f(\boldsymbol{w}_2)\| \\ \leq &(L_0 + L_1\|\nabla f(\boldsymbol{w}_1)\|)\|\boldsymbol{w}_1 - \boldsymbol{w}_2\|. \end{aligned}$$

Assumption 2.1 is a general form of (L_0, L_1)-smooth condition, equivalent to the Hessian-bound form (Zhang et al., 2019) when Hessian exists.

When Assumption 2.1 holds, the local smoothness of the objective function is bounded by the the linear form of the gradient norm (i.e., $L(\boldsymbol{w}) \leq L_0 + L_1\|\nabla f(\boldsymbol{w})\|$. We know that the *smoothness constant* $L(\boldsymbol{w})$ governs how much the gradient can change locally. If $L(\boldsymbol{w})$ scales with $\|\nabla f(\boldsymbol{w})\|$, the curvature (and thus the risk of overshooting) increases with the gradient's magnitude. This necessitates a smaller learning rate when the gradient is large and allows a larger rate when the gradient is small.

A brief derivation (see details in Appendix B) gives a range of η_t that guarantees decreasing function value at each step, i.e., $\eta_t \leq 1/(L_0 + L_1\|\nabla f(\boldsymbol{w}_t)\|)$, which ensures convergence by balancing the descent and curvature terms. This adaptively scales η inversely with the grad's magnitude.

In practice, we do not know the exact values of L_0 and L_1, a typical choice of η_t should be

$$\eta_t = \frac{C}{\|\nabla f(\boldsymbol{w}_t)\| + \epsilon},$$

for some constant or scheduled constant C after taking account of avoiding explosion near minima. Such an argument can be extended to coordinate-wise sense, which necessitates per-coordinate adaptive learning rates.

We note that Adam adapts learning rates using second-moment estimates, i.e., the exponential average of of the square of historical gradients to approximate the gradient magnitude. We draw inspiration from (Zhang et al., 2020), which demonstrates that momentum—the exponential moving average of historical gradients—can itself serve as a robust proxy for gradient magnitudes. Building on this insight, we propose replacing second-moment estimation with a novel denominator derived from a weighted combination of momentum and the current mini-batch gradient. This approach retains the benefits of adaptive learning rate tuning while eliminating the computational overhead of tracking second moment statistics.

2.2 The Design of *AdamS*

The desing of *AdamS* is given by Algorithm 1. Specifically, the denominator is

$$\boldsymbol{\nu}_t \leftarrow \beta_2 \boldsymbol{m}_{t-1}^{\odot 2} + (1-\beta_2)\boldsymbol{g}_t^{\odot 2}.$$

Algorithm 1 AdamW v.s. *AdamS*

Input: momentum parameter β_1, denominator parameter β_2, weight decay λ, learning rate η, objective f, regularizer ϵ
Initialize: $\boldsymbol{w}_0$, $\boldsymbol{m}_0 \leftarrow 0, \boldsymbol{\nu}_0 \leftarrow 0, t \leftarrow 0$
while $\boldsymbol{w}_t$ not converged **do**
  $t \leftarrow t+1$
  $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_{t-1})$
  update state tracking
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$
  AdamW: $\boldsymbol{\nu}_t \leftarrow \beta_2 \boldsymbol{\nu}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^{\odot 2}$
  AdamS: $\boldsymbol{\nu}_t \leftarrow \beta_2 \boldsymbol{m}_{t-1}^{\odot 2} + (1-\beta_2)\boldsymbol{g}_t^{\odot 2}$
  update model parameters
  $\boldsymbol{w}_t \leftarrow (1-\eta_t\lambda)\boldsymbol{w}_{t-1} - \eta_t \left(\frac{1}{\sqrt{\boldsymbol{\nu}_t}+\epsilon} \odot \boldsymbol{m}_t\right)$
end while
return $\boldsymbol{w}_t$

2.3 The Properties of *AdamS*

We next compare the behavior of *AdamS* and that of AdamW.

Analytical comparison. The numerators of *AdamS* and AdamW are the same. For the denominator, we make some estimation as a thought verification. We consider the following sequence $\{X_t\}$, where $X_t \sim \mathcal{N}(\mu, \sigma^2)$ are independent. Then the distribution of the exponentially weighted moving average (EMA) of their squared values

$$S_t = (1-\beta_2)X_t^2 + \beta_2 S_{t-1}, \quad t = 1, 2, \dots, T.$$

follows a weighted sum of noncentral chi-squared distributions. As t becomes large, such a distribution tends to be a non-centered Gaussian distribution. We compute the mean and variance of S_t,

$$\mathbb{E}[S_t] = (\mu^2 + \sigma^2)(1-\beta_2^t),$$

$$\mathrm{Var}(S_t) = \left(2\sigma^4 + 4\mu^2\sigma^2\right)\frac{1-\beta_2}{1+\beta_2}(1-\beta_2^{2t}).$$

Consequently, $\mathbb{E}[S_\infty] = (\mu^2 + \sigma^2)$, and $\mathrm{Var}(S_\infty) = (2\sigma^4 + 4\mu^2\sigma^2)(1-\beta_2)/(1+\beta_2)$.

On the other side, the distribution of the exponential moving average of X_t, i.e., $M_t = (1 -$

$\beta_1)X_t + \beta_1 M_{t-1}, \quad t = 1, 2, \ldots,$ follows a Gaussian distribution.

The denominator of *AdamS* involves the following quantity, $V_t := \beta\, M_{t-1}^2 + (1-\beta)\, X_t^2$. Since X_t and M_{t-1} are independent, V_t is the sum of two independent scaled noncentral chi–squared random variables with one degree of freedom. We have

$$\mathbb{E}[V_\infty] = \mu^2 + \sigma^2 \left(1 - \frac{2\beta\beta_1}{1+\beta_1}\right),$$

$$\mathrm{Var}(V_\infty) = 2\sigma^4 \left[\beta^2 \left(\frac{1-\beta_1}{1+\beta_1}\right)^2 + (1-\beta)^2\right] + 4\mu^2\sigma^2 \left[\beta^2 \frac{1-\beta_1}{1+\beta_1} + (1-\beta)^2\right].$$

By comparing $\mathbb{E}[S_\infty]$ and $\mathbb{E}[V_\infty]$, we note that if $\mu \gg \sigma$, i.e., a regime achievable under large batch sizes where gradient noise becomes negligible, *AdamS*'s behavior increasingly resembles that of AdamW. This alignment with AdamW's dynamics under low-noise conditions mirrors practical LLM pretraining setups, where large batch sizes are standard.

Empirical comparison between the update matrices of *AdamS* and AdamW. We analyze the update matrices of AdamW and *AdamS* along the training trajectory of a GPT-2 Small model. The detailed experimental setup is provided in Section 4.1.

To quantify the similarity between the updates, we compute the cosine similarity between the update matrices of *AdamS* and AdamW throughout the training process with AdamW. The results are presented in Figure 2. For comparison, we also include the cosine similarity between AdamW and the recently proposed Adam-mini (Zhang et al., 2024). The results show that *AdamS* exhibits a strong alignment with AdamW, closely matching its update direction.

The magnitude of *AdamS* update. For $\beta_1 = 0.9$, we plot the update magnitude of *AdamS* when the $gradient/momentum$ values span $[-13, 13]$, covering most values in practice, in Figure 3. We can see that overly large β_2 values can destabilize updates by inflating the denominator's sensitivity to outliers. To mitigate this, we recommend not setting β_2 too large, and a typical value of $\beta_2 = 0.95$ works well and aligns with empirical choice of AdamW for LLM pretraining.

Memory cost and throughput of *AdamS*. *AdamS* effectively reduces optimizer state memory usage by half. However, the extent of improvement in throughput and maximum batch size compared to the original AdamW depends on the model size and GPU type, as the primary bottleneck may be either memory or computation. Notably, as model size increases, the benefits of *AdamS* become more pronounced, aligning well with practical large language model (LLM) training scenarios. As shown in Table 1, *AdamS* can improve over AdamW in terms of throughput by almost 36%, i.e., reducing the time 6.9s to 4.4s of passing a batch of tokens, for GPT2-XL pretraining.

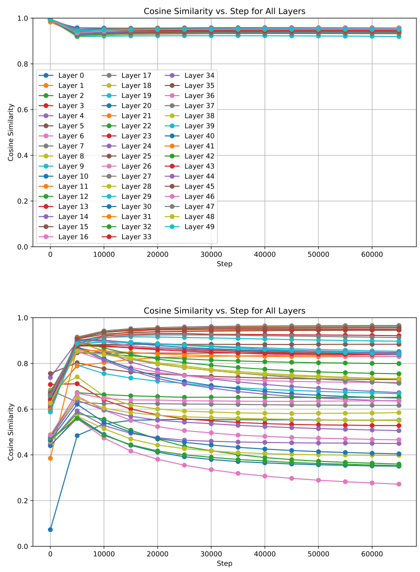

Figure 2: The cosine similarities between the update matrices of *AdamS* and AdamW (upper), Adam-mini and AdamW (lower) for all layers of GPT2-Small model . Across the training trajectory, the update direction of *AdamS* closely aligns with that of AdamW.

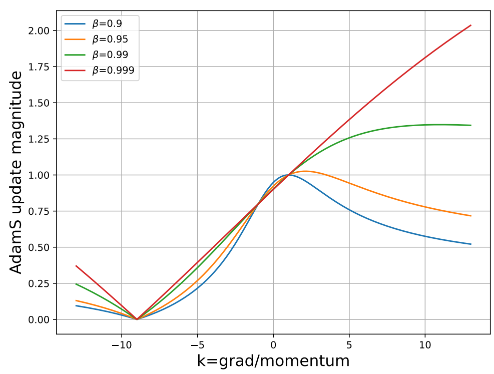

Figure 3: The update magnitude of *AdamS* when the grad/momentum changes with $\beta_1 = 0.1$ and $\beta = 0.9, 0.95, 0.99, 0.999$.

Model	Optimizer	Max batch	Throughput
774M	AdamW	10	2.0s
	AdamS	10	2.0s
1.5B	AdamW	1	6.9s
	AdamS	3	4.4s

Table 1: Memory cost and throughput comparison between AdamW and *AdamS*. The maximum batch size (Max batch) is the largest allowable batch without Out of Memory and the throughput (Throughput) is measured by the time (in seconds) for one iteration of passing 480K tokens with gradient accumulation. Experiment setup: 8 A100 GPUs with 40GB memory, training with GPT2-XL (1.5B) and GPT2-Large (774M).

3 Convergence of *AdamS*

This section establishes the theoretical convergence of *AdamS*. We first introduce another key assumption on the gradient noises.

Assumption 3.1 (Sub-gaussian noise)**.** We assume that the stochastic noise $\boldsymbol{g}_t$ is unbiased, i.e., $\mathbb{E}^{|\mathcal{F}_t}\boldsymbol{g}_t = \boldsymbol{G}_t$. We further assume $\boldsymbol{g}_t$ is centered with sub-gaussian norm, i.e., there exists some positive constant R, such that $\mathbb{P}^{|\mathcal{F}_t}(\|\boldsymbol{g}_t - \nabla f(\boldsymbol{w}_t)\| \geq s) \leq 2e^{-\frac{s^2}{2R^2}}$.

Assumption 3.1 is one of the weakest assumptions on the noise in existing literature, and generalizes bounded gradient assumption (Défossez et al., 2022) and bounded noise assumption (Li et al., 2023). Based on Assumption 2.1 and 3.1

Theorem 3.2. *Let Assumptions 2.1 and 3.1 hold. Then, setting $\eta_t = \tilde{\mathcal{O}}(\frac{1}{\sqrt{T}})$, $\beta_1 = 1 - \tilde{\Theta}(\frac{1}{\sqrt{T}})$, and $\beta_2 = 1 - \tilde{\Theta}(\frac{1}{T})$ with $\frac{1-\beta_1}{\eta} \geq C$, where C is some constant defined in Eq. (4) , we have that* AdamS *in Algorithm 1 satisfies*

$$\mathbb{E} \min_{t\in[1,T]} \|\nabla f(\boldsymbol{w}_t)\| \leq \tilde{\mathcal{O}}\left(\frac{1}{\sqrt[4]{T}}\right).$$

Proof. The proof is relegated to Appendix C due to space constraints. □

The derived convergence rate matches the known lower bound of $\Omega(1/\sqrt{T})$ for any gradient-based optimizer, including AdamW (Arjevani et al., 2022). This result not only demonstrates that the convergence rate in Theorem 3.2 is tight —- achieving the theoretically optimal bound —- but also provides a rigorous theoretical guarantee for AdamS's efficiency in optimizing Transformer architectures.

4 Empirical Performance of *AdamS*

In this section, we apply *AdamS* for large language model pretraining tasks and post-training tasks to demonstrate that *AdamS* can achieve performance comparable to AdamW with similar hyperparameters while requiring significantly less memory.

4.1 GPT2 experiments

In this experiment, we demonstrate that *AdamS* achieves performance comparable to AdamW for pretraining GPT2 (Radford et al., 2019) on the OpenWebText dataset (Gao et al., 2020) using the popular nanoGPT codebase[1]. We evaluate three variants: GPT2 Small (125M parameters), GPT2 Medium (355M parameters), and GPT2 Large (770M parameters).

Baselines. We primarily compare *AdamS* with AdamW (Loshchilov and Hutter, 2019), the most widely used optimizer in language modeling tasks, and Lion (Chen et al., 2023), a recently proposed optimizer that eliminates the need for second-moment estimates, discovered by symbolic search.

We adopt typical hyperparameter choices, following the settings used in (Zhang et al., 2024; Liu et al., 2023). For AdamW, we set $(\beta_1, \beta_2) = (0.9, 0.95)$ with a weight decay of 0.1, and we use a learning rate of 6×10^{-4} for the GPT2 Small model and $\mathrm{lr} = 3 \times 10^{-4}$ for the GPT2 Medium and GPT2 Large models. For Lion, as suggested by Chen et al. (2023), we use $(\beta_1, \beta_2) = (0.95, 0.98)$, set the learning rate to $0.1 \times \mathrm{lr}_{\mathrm{AdamW}}$, and choose a weight decay of $10 \times \mathrm{weight\_decay}_{\mathrm{AdamW}}$. For *AdamS*, we use the same hyperparameters as AdamW; that is, $\mathrm{lr} = \mathrm{lr}_{\mathrm{AdamW}}$, $(\beta_1, \beta_2) = (0.9, 0.95)$, and $\mathrm{weight\_decay} = \mathrm{weight\_decay}_{\mathrm{AdamW}}$.

Implementation. Following standard practices, for all GPT-2 models, we set the context length to be 1024 tokens. We use a batch size of 480 and employ a cosine learning rate schedule, setting the final learning rate to $0.1 \times \mathrm{lr}$ as suggested by Rae et al. (2021). We employ gradient clipping by norm with a threshold of 1.0, and we use a fixed warm-up period of 2,000 steps. The algorithms are implemented in PyTorch (Paszke et al., 2019), and training is conducted in float16 precision on clusters equipped with Nvidia Ampere or Hopper GPUs for the GPT2-Small, Medium, and Large models.

Results. The results are shown in Figure 4 and Table 2. As observed in Figure 4, the per-

[1] https://github.com/karpathy/nanoGPT

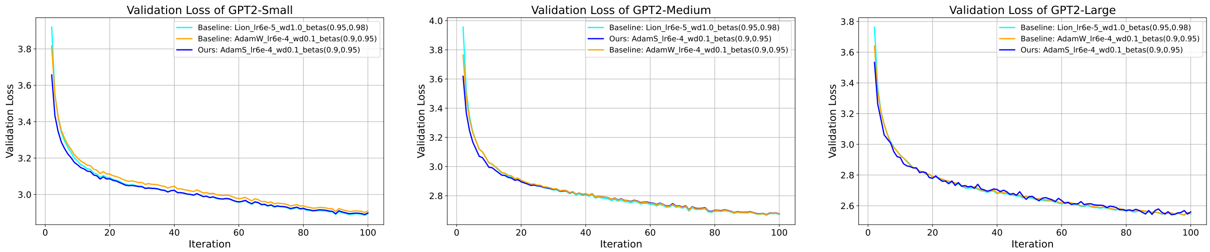

Figure 4: Validation loss curves for pretraining GPT-2 models. Across three different model sizes and with the same hyperparameters as AdamW, the proposed *AdamS* achieves convergence comparable to baseline methods—without the need to store AdamW's second-moment estimates.

formance of *AdamS* closely mirrors the AdamW curves across all three model sizes throughout the training process. This is achieved using the same hyperparameters as those for AdamW. Further details are provided in Table 2. Additionally, we confirm this behavior over 300K training iterations, which corresponds to processing over 14.4 billion tokens. Due to resource constraints, we were unable to train larger models or process more tokens.

4.2 Llama2 Pretraining Experiments

In this experiment, we confirm the behavior of AdamS for pretraining an even larger model Llama2-13B (Touvron et al., 2023b). It is trained with the well-known Torchtitan library[2] on the C4 dataset (Raffel et al., 2020).

Hyperparameter choice. For AdamW, we use $(\beta_1, \beta_2) = (0.9, 0.95)$, a peak learning rate of 10^{-4}, and a weight decay of 0.1. For *AdamS*, we use the same hyperparameters as AdamW. For Lion, we use the recommended settings: $\text{lr} = 0.1 \times \text{lr}_{\text{AdamW}}$ and $\text{weight\_decay} = 10 \times \text{weight\_decay}_{\text{AdamW}}$.

Implementation. The training setup involves a batch size of 2×8, a context length of 2048, and gradient clipping with a maximum norm of 1.0. The learning rate schedule includes a fixed 100-step warmup followed by linear decay. The training is conducted in bfloat16 precision on one node equipped with 8 Nvidia Hopper GPUs with 80G memory. Due to budget limitations, we train the model for 30K steps, which corresponds to processing over 0.96B tokens.

Results. The results are summarized in Figure 1. As shown in Figure 1, *AdamS* achieves performance nearly identical to AdamW across the training trajectory under the same hyperparameters. Notably, training with *AdamS* reduces memory consumption by 20% when using a popular training recipe, i.e., Fully Sharded Data Parallel (FSDP) technique (Paszke et al., 2019) on 4 NVIDIA Hopper GPUs. Additionally, by eliminating the need to communicate second-moment estimates across GPUs and nodes, *AdamS* alleviates communication bottlenecks, a critical advantage for low-end GPU clusters where inter-card bandwidth is often a limiting factor.

4.3 RL Post-training of LLMs

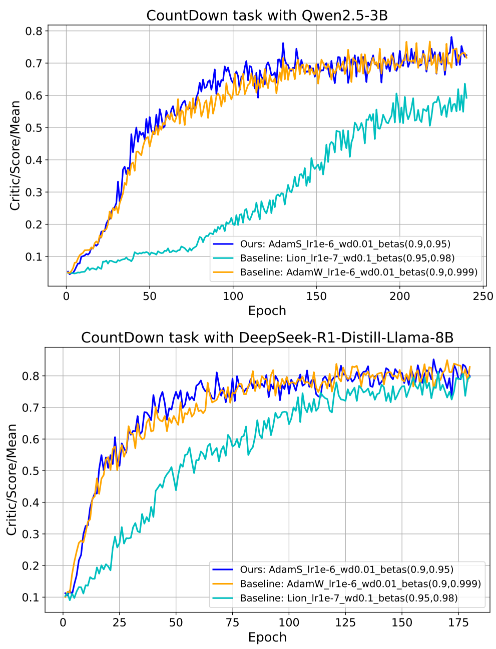

Figure 5: Mean critic scores for reinforcement learning (RL) post-training using the GRPO algorithm on the CountDown task are presented for the Qwen2.5-3B and DeepSeek-R1-Distill-Llama-8B models. The proposed *AdamS* closely resembles AdamW's performance trajectory, achieving similar convergence curves. In contrast, Lion with default hyperparameters demonstrates significantly slower convergence under the same conditions.

In this experiment, we leverage the TinyZero project (Pan et al., 2025) that provides a clean, min-

[2] https://github.com/pytorch/torchtitan

Table 2: Comparison of Lion, AdamW and *AdamS* on training GPT2 with the OpenWebText dataset.

Model size	Iteration Budget	Optimizer	Peak LR	Weight decay	(β_1, β_2)	Valid. PPL
124M	100K	AdamW	6e-4	0.1	$(0.9, 0.95)$	2.902
		Lion	6e-5	1.0	$(0.95, 0.98)$	2.886
		AdamS	6e-4	0.1	$(0.9, 0.95)$	2.890
	300K	AdamW	6e-4	0.1	$(0.9, 0.95)$	2.867
		Lion	6e-5	1.0	$(0.95, 0.98)$	2.847
		AdamS	6e-4	0.1	$(0.9, 0.95)$	2.866

imal, and accessible reproduction of the DeepSeek R1-Zero framework (Guo et al., 2025). We choose two models Qwen2.5-3B (Team, 2024) and R1-Distilled-Llama8B (Guo et al., 2025) and evaluate the DeepSeek R1-Zero method on the Countdown Numbers Game. In this task, the model is asked to use a set of randomly chosen numbers along with basic arithmetic operations $(+, -, \times, \div)$ to reach a specified target number, with each number used only once.

Hyperparameter choice. For the baseline AdamW setup, we use the default learning rate of 1×10^{-6}, $(\beta_1, \beta_2) = (0.9, 0.999)$, and a weight decay of 1×10^{-2}. We test the Group Relative Policy Optimization (GRPO) reinforcement learning algorithm (Shao et al., 2024; Guo et al., 2025) with all other hyperparameters maintained as in the original project. For *AdamS*, we adopt the same hyperparameters as AdamW, except that we set $\beta_2 = 0.95$ for good stability, as explained in Section 2.2 and Figure 3. For Lion, we follow the recommendations from the original paper by setting $\text{lr} = 0.1 \times \text{lr}_{\text{AdamW}}$, $\text{weight\_decay} = 10 \times \text{weight\_decay}_{\text{AdamW}}$, and $(\beta_1, \beta_2) = (0.95, 0.98)$.

Implementation. The TinyZero framework implements the DeepSeek R1-Zero reinforcement learning objective, which encourages the models to generate an extended chain-of-thought before producing a final answer. This approach aims to guide the models in developing a structured reasoning process for the Countdown Numbers Game.

Results. The results are shown in Figure 5. Across two distinct base models—Qwen2.5-3B and the distilled DeepSeek-R1-Distill-Llama-8B—the score curves of *AdamS* closely align with those of AdamW, even occasionally surpassing its validation performance. This consistency underscores *AdamS*'s ease of adoption across diverse tasks, requiring no specialized tuning. In contrast, Lion, when applied with its default hyperparameters, exhibits much slower convergence under identical experimental conditions.

This point holds significant practical value: while many optimizers excel in some specific scenarios with carefully tuned hyperparameters, *AdamS*'s robust performance easily generalizes to unseen tasks without much hyperparameter tuning, making it a scalable solution for both current and future applications.

5 Discussion and Conclusion

We have proposed a well-motivated design of LLM optimizer, *AdamS*, which can serve as the newly default optimizer for training large-scale language model training, because of its efficiency, simplicity, and theoretical rigor. By replacing second-moment estimation with a momentum-weighted root mean square denominator, the method achieves computational parity with SGD while matching the performance of Adam-family optimizers in both pre-training and post-training scenarios. Its seamless integration into existing frameworks—enabled by AdamW-compatible hyperparameters and model-agnostic design—removes adoption barriers, offering practitioners a "plug-and-play" upgrade.

The theoretical property of *AdamS* has also been extensively analyzed, including the update magnitude estimation and convergence under relaxed smoothness assumption. This theoretical insight, coupled with empirical validation across architectures (e.g., GPT-2, Llama2) and training paradigms (e.g., RL post-training), demonstrates robustness to scale and task diversity. Notably, *AdamS*'s elimination of communication overhead for second-moment statistics positions it as a scalable solution for communication-bounded environments.

Future work may explore *AdamS*'s applicability to emerging architectures and its synergies with advanced parallelism strategies for next-generation LLM development.

Limitations

While *AdamS* achieves promising performance across tasks and model scales, several limitations deserve discussion. First, our experiments were constrained by computational resources, particularly in pretraining scenarios (e.g., Llama2-13B). Validating *AdamS*'s efficacy at extreme scales—such as models beyond 100B parameters, datasets exceeding 1T tokens, or emerging architectures like Mixture of Experts (MoE)—remains critical for confirming its scalability in production-grade pipelines. Such studies would require computational resources far beyond our current capacity.

Second, fairly benchmarking optimizers has inherent challenges due to confounding variables like learning rate schedules, weight decay policies, optimizer-specific hyperparameters (e.g., *AdamS*'s momentum weighting), and implementation efficiency. While our work compares *AdamS* against strong baselines (AdamW, Lion) using established hyperparameters, we limited exhaustive hyperparameter searches across all optimizers to maintain parity.

These limitations underscore the need for community-driven standardization of optimizer evaluations and deeper exploration of *AdamS*'s behavior in extreme-scale regimes. To foster reproducibility, we will release all code, configurations, and training protocols to facilitate reproducibility and encourage broader investigation.

References

Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. 2019. Memory efficient adaptive optimization. *Advances in Neural Information Processing Systems*, 32.

Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, and Blake Woodworth. 2022. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, pages 1–50.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *arXiv preprint*.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Yao Liu, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, et al. 2023. Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint*.

Ashok Cutkosky and Harsh Mehta. 2020. Momentum improves normalized SGD. In *International conference on machine learning*, pages 2260–2268. PMLR.

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

Timothy Dozat. 2016. Incorporating Nesterov Momentum into Adam. In *Proceedings of the 4th International Conference on Learning Representations*, pages 1–4.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2021. All NLP tasks are generation tasks: A general pretraining framework. *CoRR*, abs/2103.10360.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Matthew Faw, Litu Rout, Constantine Caramanis, and Sanjay Shakkottai. 2023. Beyond uniform smoothness: A stopped analysis of adaptive sgd. *arXiv preprint arXiv:2302.06570*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Vineet Gupta, Tomer Koren, and Yoram Singer. 2018. Shampoo: Preconditioned stochastic tensor optimization. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1842–1850. PMLR.

Meixuan He, Yuqing Liang, Jinlan Liu, and Dongpo Xu. 2023. Convergence of adam for non-convex objectives: Relaxed hyperparameters and non-ergodic case. *arXiv preprint arXiv:2307.11782*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Jean Kaddour, Oscar Key, Piotr Nawrot, Pasquale Minervini, and Matt Kusner. 2023. No train no gain: Revisiting efficient training algorithms for transformer-based language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint*.

Haochuan Li, Ali Jadbabaie, and Alexander Rakhlin. 2023. Convergence of Adam under relaxed assumptions. *arXiv preprint arXiv:2304.13972*.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. 2023. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *arXiv preprint arXiv:2305.14342*.

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

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

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. 2019. Adaptive gradient methods with dynamic bound of learning rate. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, Louisiana.

Yang Luo, Xiaozhe Ren, Zangwei Zheng, Zhuo Jiang, Xin Jiang, and Yang You. 2023. Came: Confidence-guided adaptive memory efficient optimization. *arXiv preprint arXiv:2307.02047*.

Kai Lv, Hang Yan, Qipeng Guo, Haijun Lv, and Xipeng Qiu. 2023. Adalomo: Low-memory optimization with adaptive learning rate. *arXiv preprint arXiv:2310.10195*.

Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, Binh Tang, Diana Liskovich, Puxin Xu, Yuchen Zhang, Melanie Kambadur, Stephen Roller, and Susan Zhang. 2023. A theory on adam instability in large-scale machine learning. *Preprint*, arXiv:2304.09871.

Jiayi Pan, Junjie Zhang, Xingyao Wang, Lifan Yuan, Hao Peng, and Alane Suhr. 2025. Tinyzero. https://github.com/Jiayi-Pan/TinyZero. Accessed: 2025-01-24.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. volume 32.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, F. Song, John Aslanides, Sarah Henderson, R. Ring, S. Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

Colin Raffel, 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. *Journal of machine learning research*, 21(140):1–67.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. Zero: Memory optimization towards training A trillion parameter models. *CoRR*, abs/1910.02054.

Frank Schneider, Zachary Nado, Naman Agarwal, George E. Dahl, and Philipp Hennig. 2022. HITY workshop poll, NeurIPS 2022. `https://github.com/fsschneider/HITYWorkshopPoll`.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Hugo Touvron, Thibault Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothee Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Bohan Wang, Jingwen Fu, Huishuai Zhang, Nanning Zheng, and Wei Chen. 2023a. Closing the gap between the upper bound and lower bound of adam's iteration complexity. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Bohan Wang, Huishuai Zhang, Zhiming Ma, and Wei Chen. 2023b. Convergence of adagrad for nonconvex objectives: Simple proofs and relaxed assumptions. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 161–190. PMLR.

Bohan Wang, Yushun Zhang, Huishuai Zhang, Qi Meng, Zhi-Ming Ma, Tie-Yan Liu, and Wei Chen. 2022. Provable adaptivity in Adam. *arXiv preprint arXiv:2208.09900*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Da Yu, Huishuai Zhang, Wei Chen, Jian Yin, and Tie-Yan Liu. 2021. Large scale private learning via low-rank reparametrization. In *International Conference on Machine Learning*, pages 12208–12218. PMLR.

Bohang Zhang, Jikai Jin, Cong Fang, and Liwei Wang. 2020. Improved analysis of clipping algorithms for non-convex optimization. *Advances in Neural Information Processing Systems*, 33:15511–15521.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. 2019. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Ming-Wei Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xiaodong Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. 2024. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. 2020. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. In *Advances in Neural Information Processing Systems*, volume 33, pages 18795–18806. Curran Associates, Inc.

A Comparison with Lion

Algorithm 2 Lion Optimizer (Chen et al., 2023)

Input: momentum parameters β_1, β_2, weight decay λ, learning rate η, objective function f
Initialize starting point $\boldsymbol{w}_0$, initial $\boldsymbol{m}_0 \leftarrow 0, t \leftarrow 0$
while $\boldsymbol{w}_t$ not converged **do**
  $t \leftarrow t + 1$
  $\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_{t-1})$
  update model parameters
  $\boldsymbol{u}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$
  $\boldsymbol{w}_t \leftarrow \boldsymbol{w}_{t-1} - \eta_t(\text{sign}(\boldsymbol{u}_t) + \lambda \boldsymbol{w}_{t-1})$
  update momentum tracking
  $\boldsymbol{m}_t \leftarrow \beta_2 \boldsymbol{m}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t$
end while
return $\boldsymbol{w}_t$

B Derivation of the Learning Rate under (L_0, L_1) Smoothness

The *smoothness constant* $L(\boldsymbol{w})$ governs how much the gradient can change locally. If $L(\boldsymbol{w})$ scales with $\|\nabla f(\boldsymbol{w})\|$, the curvature (and thus the risk of overshooting) increases with the gradient's magnitude. This necessitates a smaller learning rate when the gradient is large and allows a larger rate when the gradient is small.

Here is a brief derivation for the above intuition.

Descent Lemma: For $L(\boldsymbol{w})$-smooth f, the update $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla f(\boldsymbol{w}_t)$ satisfies:

$$f(\boldsymbol{w}_{t+1}) \leq f(\boldsymbol{w}_t) - \eta \|\nabla f(\boldsymbol{w}_t)\|^2 + \frac{\eta^2 L(\boldsymbol{w}_t)}{2} \|\nabla f(\boldsymbol{w}_t)\|^2.$$

Substitute $L(\boldsymbol{w}_t) \leq L_0 + L_1 \|\nabla f(\boldsymbol{w}_t)\|$:

$$f(\boldsymbol{w}_{t+1}) \leq f(\boldsymbol{w}_t) - \eta \|\nabla f(\boldsymbol{w}_t)\|^2 + \frac{\eta^2 (L_0 + L_1 \|\nabla f(\boldsymbol{w}_t)\|)}{2} \|\nabla f(\boldsymbol{w}_t)\|^2.$$

Ensure Decrease: For $f(\boldsymbol{w}_{t+1}) \leq f(\boldsymbol{w}_t)$, require:

$$-\eta \|\nabla f(\boldsymbol{w}_t)\|^2 + \frac{L_0 + L_1 \|\nabla f(\boldsymbol{w}_t)\|}{2} \eta^2 \|\nabla f(\boldsymbol{w}_t)\|^2 \leq 0.$$

Factor out $\eta \|\nabla f(\boldsymbol{w}_t)\|^2$:

$$\eta \|\nabla f(\boldsymbol{w}_t)\|^2 \left(-1 + \eta \frac{L_0 + L_1 \|\nabla f(\boldsymbol{w}_t)\|}{2} \right) \leq 0.$$

This implies:

$$\eta \leq \frac{2}{L_0 + L_1 \|\nabla f(\boldsymbol{w}_t)\|}.$$

C Proof of Theorem 3.2

This section collects the proof of Theorem 3.2. Overall, the proof is inspired by the proof of Theorem 4.2 in Li et al. (2023), which utilizes stopping time to bound the norm of stochastic gradients.

In the following proof, we define

$$\sigma \triangleq \max\left\{\sqrt{2R^2\log\frac{T}{\delta}}, L\frac{\eta_t}{1-\beta_1}\max\{\frac{\beta_1}{\sqrt{\beta_2}}, \frac{1-\beta_1}{\sqrt{1-\beta_2}}\}, \frac{3L_0}{4L_1}\right\}, \tag{1}$$

$$G \triangleq \max\{\frac{3L_0}{4L_1}, 72L_1(f(\boldsymbol{w}_1)-f^*), \sqrt{72L_1\sigma^2\eta_t((1-\beta_1)T+1)}, 60\sqrt{L_1R^2\sigma^2\eta_t\sqrt{2T\log(1/\delta)}}\}, \tag{2}$$

$$F \triangleq \frac{G^2}{3(3L_0+4L_1G)}, \tag{3}$$

$$C \triangleq \sqrt{\frac{4L^2}{\varepsilon^4}}(G+\sigma+\varepsilon). \tag{4}$$

We consider the following stopping time:

$$\tau := \min\{t \mid f(\boldsymbol{w}_t)-f^* > F\} \wedge \min\{t \mid \|\nabla f(\boldsymbol{w}_t)-\boldsymbol{g}_t\| > \sigma\} \wedge (T+1). \tag{5}$$

Due to Lemma C.2 and the definition of F (Eq. (3)), one can easily see that for any $t<\tau$, $\|\nabla f(\boldsymbol{w}_t)\| \le G$.

Also, as we are dealing with optimizers with coordinate-wise learning rates, we introduce the following norm to ease the burden of writing. Specifically, let $\boldsymbol{b}\in\mathbb{R}^d$ be a vector with each coordinate positive. For any $\boldsymbol{a}\in\mathbb{R}^d$, we define

$$\|\boldsymbol{a}\|_{\boldsymbol{b}} = \sqrt{\langle \boldsymbol{a}\odot\boldsymbol{b}, \boldsymbol{a}\rangle}.$$

C.1 Useful Lemmas

The following lemma bounds the change of f through its local second-order expansion.

Lemma C.1. *Let Assumption 2.1 holds. Then, for any three points $\boldsymbol{w}^1, \boldsymbol{w}^2\in\mathbb{R}^d$ satisfying $\|\boldsymbol{w}^1-\boldsymbol{w}^2\|\le\frac{1}{L_1}$, we have*

$$f(\boldsymbol{w}^2) \le f(\boldsymbol{w}^1) + \langle\nabla f(\boldsymbol{w}^1), \boldsymbol{w}^2-\boldsymbol{w}^1\rangle + \frac{1}{2}(L_0+L_1\|\nabla f(\boldsymbol{w}^1)\|)\|\boldsymbol{w}^2-\boldsymbol{w}^1\|^2.$$

Proof. By the Fundamental Theorem of Calculus, we have

$$\begin{aligned}
&f(\boldsymbol{w}^2)\\
=&f(\boldsymbol{w}^1)+\int_0^1\langle\nabla f(\boldsymbol{w}^1+a(\boldsymbol{w}^2-\boldsymbol{w}^1)), \boldsymbol{w}^2-\boldsymbol{w}^1\rangle \mathrm{d}a\\
=&f(\boldsymbol{w}^1)+\langle\nabla f(\boldsymbol{w}^1), \boldsymbol{w}^2-\boldsymbol{w}^1\rangle+\int_0^1\langle\nabla f(\boldsymbol{w}^1+a(\boldsymbol{w}^2-\boldsymbol{w}^1))-\nabla f(\boldsymbol{w}^1), \boldsymbol{w}^2-\boldsymbol{w}^1\rangle \mathrm{d}a\\
\le&f(\boldsymbol{w}^1)+\langle\nabla f(\boldsymbol{w}^1), \boldsymbol{w}^2-\boldsymbol{w}^1\rangle+\int_0^1\|\nabla f(\boldsymbol{w}^1+a(\boldsymbol{w}^2-\boldsymbol{w}^1))-\nabla f(\boldsymbol{w}^1)\|\|\boldsymbol{w}^2-\boldsymbol{w}^1\|\mathrm{d}a\\
\overset{(\star)}{\le}&f(\boldsymbol{w}^1)+\langle\nabla f(\boldsymbol{w}^1), \boldsymbol{w}^2-\boldsymbol{w}^1\rangle+\int_0^1(L_0+L_1\|\nabla f(\boldsymbol{w}^1)\|)\|a(\boldsymbol{w}^2-\boldsymbol{w}^1)\|\|\boldsymbol{w}^2-\boldsymbol{w}^1\|\mathrm{d}a\\
\le&f(\boldsymbol{w}^1)+\langle\nabla f(\boldsymbol{w}^1), \boldsymbol{w}^2-\boldsymbol{w}^1\rangle+\frac{1}{2}(L_0+L_1\|\nabla f(\boldsymbol{w}^1)\|)\|\boldsymbol{w}^2-\boldsymbol{w}^1\|^2,
\end{aligned}$$

where Inequality $(\star)$ uses the fact $\|\boldsymbol{w}^2-\boldsymbol{w}^1\|\le\frac{1}{L_1}$, so that Assumption 2.1 can be applied.

The proof is completed. $\square$

The following lemma bounds the gradient norm through the function value when Assumption 2.1 holds.

Lemma C.2. *Under Assumptions 2.1, we have $\|\nabla f(\boldsymbol{w})\|^2\le 3(3L_0+4L_1\|\nabla f(\boldsymbol{w})\|)(f(\boldsymbol{w})-f^*)$.*

Proof. Denot $L := 3L_0+4L_1\|\nabla f(\boldsymbol{w})\|$. Let $\boldsymbol{v} := \boldsymbol{w}-\frac{1}{2L}\nabla f(\boldsymbol{w})$. Then one can easily see

$$\|\boldsymbol{v}-\boldsymbol{w}\|\le\frac{1}{2L_1},$$

and thus Lemma C.1 can be applied. Therefore, we have

$$f^* - f(\boldsymbol{w}) \le f(\boldsymbol{v}) - f(\boldsymbol{w}) \le \langle \nabla f(\boldsymbol{w}), \boldsymbol{v} - \boldsymbol{w}\rangle + \frac{L}{2}\|\boldsymbol{v} - \boldsymbol{w}\|^2 = -\frac{3L\|\nabla f(\boldsymbol{w})\|^2}{8} \le -\frac{L\|\nabla f(\boldsymbol{w})\|^2}{3}.$$

The proof is completed. □

The following lemma bounds the update of AdamS:

Lemma C.3. *For any t, let $\boldsymbol{w}_t$ be the parameter of AdamS after the t-th iteration. Then,*

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| \le \eta_t \sqrt{d} \max\{\frac{\beta_1}{\sqrt{\beta_2}}, \frac{1-\beta_1}{\sqrt{1-\beta_2}}\}.$$

Therefore, under the hyperparameter selection of Theorem 3.2, we have $\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| = \mathcal{O}(\frac{1}{\sqrt{T}})$.

Proof. We have

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| = \eta_t \left\| \frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon} \odot \boldsymbol{m}_t \right\| = \eta_t \left\| \frac{1}{\sqrt{\beta_2 \boldsymbol{m}_{t-1}^{\odot 2} + (1-\beta_2)\boldsymbol{g}_t^{\odot 2}} + \varepsilon} \odot \boldsymbol{m}_t \right\|.$$

On the other hand, by Young's inequality, we have that coordinate-wisely

$$\boldsymbol{m}_t^{\odot 2} \le \beta_1^2 \boldsymbol{m}_{t-1}^{\odot 2} + (1-\beta_1)^2 \boldsymbol{g}_t^{\odot 2}.$$

The proof is completed. □

The following lemma bounds the adaptive conditioner $\boldsymbol{\nu}_t$.

Lemma C.4. *If $t < \tau$, we have the i-th coordinate $\boldsymbol{\nu}_{t,i}$ of $\boldsymbol{\nu}_t$ satisfies*

$$0 \le \sqrt{\boldsymbol{\nu}_{t,i}} \le G + \sigma.$$

Proof. The first inequality is obvious.

For the second inequality, one can easily see that $\boldsymbol{g}_{t,i}$ satisfies the same inequality according to the definition of τ. According to the definition of $\boldsymbol{\nu}_t$, we have

$$\boldsymbol{\nu}_{t,i} = (1-\beta_2)\boldsymbol{g}_{t,i}^2 + \beta_2((1-\beta_1)\sum_{s=0}^{t-1} \beta_1^{t-1-s} \boldsymbol{g}_{s,i})^2.$$

Applying the estimation of $\boldsymbol{g}_{s,i}$ completes the proof. □

The following lemma provides a rough bound of the gap between $\nabla f(\boldsymbol{w}_t)$ and $\boldsymbol{m}_t$.

Lemma C.5. *Let $\Delta_t = \boldsymbol{m}_t - \nabla f(\boldsymbol{w}_t)$. If $t \le \tau$, we have $\|\Delta_t\| \le 2\sigma$.*

Proof. We prove this claim by induction. First, note that for $t = 1$, we have

$$\|\Delta_1\| = \|\boldsymbol{g}_1 - \nabla f(\boldsymbol{w}_1)\| \le \sigma \le 2\sigma.$$

Now suppose $\|\Delta_t\| \le 2\sigma$ for some $2 \le t \le \tau$. According to the update rule of $\boldsymbol{m}_t$, we have

$$\Delta_t = \beta_1(\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t)) + (1-\beta_1)(\boldsymbol{g}_t - \nabla f(\boldsymbol{w}_t)),$$

which implies

$$\|\Delta_t\| \le (1+\beta_1)\sigma + \|\nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t)\| \le (1+\beta_1)\sigma + L\eta_t \max\{\frac{\beta_1}{\sqrt{\beta_2}}, \frac{1-\beta_1}{\sqrt{1-\beta_2}}\}\sqrt{d} \le 2\sigma,$$

where in the second inequality, we use $\|\boldsymbol{w}_{t-1} - \boldsymbol{w}_t\| \le \frac{1}{L_1}$ when T is large enough and thus Assumption 2.1 can be applied, and Lemma C.3, and in the last inequality, we use the definition of σ (Eq. 1).

As $(1-\beta_1)\sigma = \Theta(\log T/\sqrt{T})$, which is large than $\mathcal{O}(1/\sqrt{T})$ when T is large enough. The proof is completed. □

The following lemma bounds the gap between $\nabla f(\boldsymbol{w}_t)$ and $\boldsymbol{m}_t$ recursively.

Lemma C.6. *Let $\Delta_t = \boldsymbol{m}_t - \nabla f(\boldsymbol{w}_t)$. With probability $1-\delta$,*

$$\sum_{t=1}^{\tau-1}\left(\frac{4(G+\sigma+\varepsilon)}{\varepsilon^2}\|\Delta_t\|^2 - \|\nabla f(\boldsymbol{w}_t)\|^2\right) \leq 4\sigma^2((1-\beta_1)T+1) + 20R^2\sigma^2\sqrt{2\sum_{t=2}^{T}\log(1/\delta)}$$
$$= \mathcal{O}(\sigma^2\sqrt{T\log(1/\delta)}).$$

Proof. According to the definition of $\boldsymbol{m}_t$, we have

$$\Delta_t = \beta_1(\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t)) + (1-\beta_1)(g_t - \nabla f(\boldsymbol{w}_t)). \tag{6}$$

As T is large enough, by Lemma C.3, we have $\|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\| \leq \frac{1}{L_1}$. Therefore by Assumption 2.1,

$$\|\nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t)\| \leq L\|\boldsymbol{w}_t - \boldsymbol{w}_{t-1}\| \leq \frac{\eta L}{\varepsilon}\|\boldsymbol{m}_{t-1}\| \leq \frac{\eta L}{\varepsilon}\left(\|\nabla f(\boldsymbol{w}_{t-1})\| + \|\Delta_{t-1}\|\right), \tag{7}$$

Therefore,

$$\begin{aligned}
&\|(\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t))\|^2 \\
\leq&\frac{1}{\beta_1}\|\Delta_{t-1}\|^2 + \frac{1}{1-\beta_1}\|\nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t)\|^2 \\
\leq&\frac{1}{\beta_1}\|\Delta_{t-1}\|^2 + \frac{1}{1-\beta_1}\frac{4\eta^2L^2}{\varepsilon^2}(\|\nabla f(\boldsymbol{w}_{t-1})\|^2 + \|\Delta_{t-1}\|^2)
\end{aligned}$$

where the first inequality uses Young's inequality, and the second inequality uses Eq. (7).

Due to our choice of β_1 and η, we have $\frac{\beta_1^2}{1-\beta_1}\frac{4\eta^2L^2}{\varepsilon^2} = \mathcal{O}(1/\sqrt{T})$, which is smaller than $1-\frac{1}{2}(1-\beta_1)$ when T is large enough. Therefore,

$$\beta_1^2\|(\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t))\|^2 \leq \left(\frac{1}{2} + \frac{\beta}{2}\right)\|\Delta_t\|^2 + \frac{\beta_1^2}{1-\beta_1}\frac{4\eta^2L^2}{\varepsilon^2}\|\nabla f(\boldsymbol{w}_{t-1})\|^2.$$

Therefore, applying the above inequality back to Eq. (6), we have if $t \leq \tau$,

$$\begin{aligned}
&\|\Delta_t\|^2 \\
=&\beta_1^2\|\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t)\|^2 + 2\beta_1(1-\beta_1)\langle\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t), g_t - \nabla f(\boldsymbol{w}_t)\rangle \\
&+ (1-\beta_1)^2\|g_t - \nabla f(\boldsymbol{w}_t)\|^2 \\
\leq&\frac{1+\beta_1}{2}\|\Delta_{t-1}\|^2 + \frac{\beta_1^2}{1-\beta_1}\frac{4\eta^2L^2}{\varepsilon^2}\|\nabla f(\boldsymbol{w}_{t-1})\|^2 + (1-\beta_1)^2\|g_t - \nabla f(\boldsymbol{w}_t)\|^2 \\
&+ 2\beta_1(1-\beta_1)\langle\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t), g_t - \nabla f(\boldsymbol{w}_t)\rangle,
\end{aligned} \tag{8}$$

where in the last equation we use Young's inequality.

On the other hand, note that

$$\begin{aligned}
&\beta_1(1-\beta_1)\sum_{t=2}^{\tau}\langle\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t), g_t - \nabla f(\boldsymbol{w}_t)\rangle \\
=&\beta_1(1-\beta_1)\sum_{t=2}^{T}1_{\tau\geq t}\langle\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t), g_t - \nabla f(\boldsymbol{w}_t)\rangle.
\end{aligned}$$

As $\mathbb{E}^{|\mathcal{F}_t}[1_{\tau\geq t}\langle\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t), g_t - \nabla f(\boldsymbol{w}_t)\rangle] = 0$, we have that

$$V_t \triangleq 1_{\tau\geq t}\langle\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t), g_t - \nabla f(\boldsymbol{w}_t)\rangle$$

is a martingale difference sequence. Also, according to Lemma C.5, we have when T is large enough, $\|\Delta_{t-1} + \nabla f(\boldsymbol{w}_{t-1}) - \nabla f(\boldsymbol{w}_t)\| \leq 3\sigma$, thus by Assumption 3.1, we have V_t is subgaussian with constant $3\sigma R$. Then by the Azuma-Hoeffding inequality, we have with probability at least $1-\delta/2$,

$$\left|\sum_{t=2}^{T} V_t\right| \leq 5R^2\sigma^2\sqrt{2\sum_{t=2}^{T}\log(1/\delta)}.$$

Also, due to Assumption 3.1, we have with probability at least $1-\delta/2T$,

$$\|g_t - \nabla f(\boldsymbol{w}_t)\|^2 \leq \sqrt{2R^2\log\frac{T}{\delta}} \leq \sigma.$$

Applying the above inequalities back to Eq. (8),

$$\begin{aligned}\frac{1-\beta_1}{2}\|\Delta_{t-1}\|^2 \leq \frac{1-\beta_1}{2}\|\Delta_{t-1}\|^2 \leq \|\Delta_{t-1}\|^2 - \|\Delta_t\|^2 + \frac{\beta_1^2}{1-\beta_1}\frac{4\eta^2L^2}{\varepsilon^2}\|\nabla f(\boldsymbol{w}_{t-1})\|^2 \\ + (1-\beta_1)^2\|g_t - \nabla f(\boldsymbol{w}_t)\|^2 + 2\beta_1(1-\beta_1)V_t.\end{aligned}$$

Taking a summation over t from 2 to τ, we have with probability at least $1-\delta$,

$$\begin{aligned}&\frac{1-\beta_1}{2}\sum_{t=1}^{\tau-1}\left(\|\Delta_t\|^2 - \frac{\varepsilon^2}{4(G+\sigma+\varepsilon)}\|\nabla f(\boldsymbol{w}_t)\|^2\right)\\ \leq&\sum_{t=2}^{\tau}\frac{1-\beta_1}{2}\|\Delta_{t-1}\|^2 - \frac{\beta_1^2}{1-\beta_1}\frac{4\eta^2L^2}{\varepsilon^2}\|\nabla f(\boldsymbol{w}_{t-1})\|^2\\ \leq&\|\Delta_1\|^2 - \|\Delta_\tau\|^2 + (1-\beta_1)^2\sigma^2T + 10(1-\beta_1)R^2\sigma^2\sqrt{2\sum_{t=2}^{T}\log(1/\delta)}\\ \leq&2\sigma^2((1-\beta_1)^2T+1) + 10(1-\beta_1)R^2\sigma^2\sqrt{2\sum_{t=2}^{T}\log(1/\delta)},\end{aligned}$$

where the first inequality is due to the assumption in Theorem 3.2 that $\frac{\eta}{1-\beta_1} \geq C$, where C is defined in Eq. (4).

The proof is completed. $\square$

C.2 Proof of the full theorem

Proof of Theorem 3.2. Recall that by Lemma C.3

$$\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\| = \mathcal{O}(\frac{1}{\sqrt{T}}).$$

When T is large enough, $\boldsymbol{w}_t$ and $\boldsymbol{w}_{t+1}$ will fulfill the requirement of Lemma C.1, which gives

$$f(\boldsymbol{w}_{t+1}) - f(\boldsymbol{w}_t) \leq \langle\nabla f(\boldsymbol{w}_t), \boldsymbol{w}_{t+1} - \boldsymbol{w}_t\rangle + \frac{L_0 + L_1\|\nabla f(\boldsymbol{w}_t)\|}{2}\|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2.$$

If $t < \tau$, we further have $\|\nabla f(\boldsymbol{w}_t)\| \leq G$. Therefore, if $t < \tau$, the above inequality can be further bounded by

$$
\begin{aligned}
& f(\boldsymbol{w}_{t+1}) - f(\boldsymbol{w}_t) \\
\le & \langle \nabla f(\boldsymbol{w}_t), \boldsymbol{w}_{t+1} - \boldsymbol{w}_t \rangle + \frac{L_0 + L_1 G}{2} \|\boldsymbol{w}_{t+1} - \boldsymbol{w}_t\|^2 \\
= & - \langle \nabla f(\boldsymbol{w}_t), \eta_t \frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon} \odot \nabla f(\boldsymbol{w}_t) \rangle + \langle \nabla f(\boldsymbol{w}_t), \eta_t \frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon} \odot (\nabla f(\boldsymbol{w}_t) - \boldsymbol{m}_t) \rangle \\
& + \frac{L_0 + L_1 G}{2} \eta_t^2 \left\| \frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon} \odot \boldsymbol{m}_t \right\|^2 \\
= & - \eta_t \|\nabla f(\boldsymbol{w}_t)\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} + \langle \nabla f(\boldsymbol{w}_t), \eta_t \frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon} \odot (\nabla f(\boldsymbol{w}_t) - \boldsymbol{m}_t) \rangle \\
& + \frac{L_0 + L_1 G}{2} \eta_t^2 \|\boldsymbol{m}_t\|^2_{\frac{1}{(\sqrt{\boldsymbol{\nu}_t} + \varepsilon)^2}} \\
\overset{(\circ)}{\le} & - \eta_t \|\nabla f(\boldsymbol{w}_t)\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} + \frac{1}{4}\eta_t \|\nabla f(\boldsymbol{w}_t)\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} + \eta_t \|\Delta_t\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} \\
& + (L_0 + L_1 G)\eta_t^2 \|\Delta_t\|^2_{\frac{1}{(\sqrt{\boldsymbol{\nu}_t} + \varepsilon)^2}} + (L_0 + L_1 G)\eta_t^2 \|\nabla f(\boldsymbol{w}_t)\|^2_{\frac{1}{(\sqrt{\boldsymbol{\nu}_t} + \varepsilon)^2}} \\
= & - \frac{3}{4}\eta_t \|\nabla f(\boldsymbol{w}_t)\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} + \eta_t \|\Delta_t\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} \\
& + (L_0 + L_1 G)\eta_t^2 \|\Delta_t\|^2_{\frac{1}{(\sqrt{\boldsymbol{\nu}_t} + \varepsilon)^2}} + (L_0 + L_1 G)\eta_t^2 \|\nabla f(\boldsymbol{w}_t)\|^2_{\frac{1}{(\sqrt{\boldsymbol{\nu}_t} + \varepsilon)^2}}
\end{aligned}
$$

where Δ_t is defined as $\Delta_t = \boldsymbol{m}_t - \nabla f(\boldsymbol{w}_t)$ and inequality $(\circ)$ uses Young's inequality. According to Lemma C.4, we further have

$$
\begin{aligned}
& f(\boldsymbol{w}_{t+1}) - f(\boldsymbol{w}_t) \\
\le & - \frac{3}{4}\eta_t \|\nabla f(\boldsymbol{w}_t)\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} + \eta_t \|\Delta_t\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} \\
& + \frac{(L_0 + L_1 G)\eta_t^2}{\varepsilon} \|\Delta_t\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} + \frac{(L_0 + L_1 G)\eta_t^2}{\varepsilon} \|\nabla f(\boldsymbol{w}_t)\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} .
\end{aligned}
$$

With large enough T, we have $\eta_t \le \frac{\varepsilon}{4(L_0 + L_1 G)}$, and thus

$$
\begin{aligned}
& f(\boldsymbol{w}_{t+1}) - f(\boldsymbol{w}_t) \\
\le & - \frac{1}{2}\eta_t \|\nabla f(\boldsymbol{w}_t)\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} + 2\eta_t \|\Delta_t\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} \\
\le & - \frac{1}{2(G + \sigma + \varepsilon)}\eta_t \|\nabla f(\boldsymbol{w}_t)\|^2 + 2\frac{\eta_t}{\varepsilon} \|\Delta_t\|^2_{\frac{1}{\sqrt{\boldsymbol{\nu}_t} + \varepsilon}} \\
\le & - \frac{1}{2(G + \sigma + \varepsilon)}\eta_t \|\nabla f(\boldsymbol{w}_t)\|^2 + 2\frac{\eta_t}{\varepsilon^2} \|\Delta_t\|^2 .
\end{aligned}
$$

After taking sum over t and rearranging, we have

$$
\sum_{t=1}^{\tau-1} \left(\|\nabla f(\boldsymbol{w}_t)\|^2 - \frac{2(G + \sigma + \varepsilon)}{\varepsilon^2} \|\Delta_t\|^2 \right) \le \frac{2(G + \sigma + \varepsilon)}{\eta_t} \left(f(\boldsymbol{w}_1) - f(\boldsymbol{w}_\tau) \right) .
$$

Multiplying both sides of the above inequality by 2 and adding the inequality in Lemma C.6, we obtain with probability at least $1 - \delta$,

$$
\begin{aligned}
\sum_{t=1}^{\tau-1} \|\nabla f(\boldsymbol{w}_t)\|^2 \le & \frac{2(G + \sigma + \varepsilon)}{\eta_t} (f(\boldsymbol{w}_1) - f(\boldsymbol{w}_\tau)) + 4\sigma^2((1 - \beta_1)T + 1) + 20R^2\sigma^2 \sqrt{2 \sum_{t=2}^{T} \log(1/\delta)} \qquad (9) \\
= & \tilde{\mathcal{O}}(1/\sqrt{T}).
\end{aligned}
$$

In the following proof, we will bound the probability of the event $\{\tau \leq T\}$. Note if we can show $\mathbb{P}(\tau > T) \geq 1-\delta$, the proof is completed, as conditional on $\{\tau > T\}$, $\sum_{t=1}^{\tau-1} \|\nabla f(\boldsymbol{w}_t)\|^2$ in the above inequality will become $\sum_{t=1}^{T} \|\nabla f(\boldsymbol{w}_t)\|^2$.

Obviously, the stopping time τ (eq. (5)) can be decomposed as $\tau := \min\{\tau_1, \tau_2\}$, where τ_1 and τ_2 are two stopping times defined as

$$\tau_1 := \min\{t \mid f(\boldsymbol{w}_t) - f^* > F\} \wedge (T+1),$$
$$\tau_2 := \min\{t \mid \|\nabla f(\boldsymbol{w}_t) - \boldsymbol{g}_t\| > \sigma\} \wedge (T+1),$$

We then bound $\mathbb{P}(\tau_1 \leq T)$ and $\mathbb{P}(\tau_2 \leq T)$ respectively.

Bound of $\mathbb{P}(\tau_2 \leq T)$. We bound this term by a similar practice as Lemma C.6. According to the definition of τ_2

$$\begin{aligned}
\mathbb{P}(\tau_2 \leq T) =& \mathbb{P}\left(\bigcup_{1\leq t\leq T} \{\|\nabla f(\boldsymbol{w}_t) - \boldsymbol{g}_t\| > \sigma\}\right) \\
\leq& \sum_{1\leq t\leq T} \mathbb{P}\left(\|\nabla f(\boldsymbol{w}_t) - \boldsymbol{g}_t\| > \sigma\right) \\
\leq& 2Te^{-\frac{\sigma^2}{2R^2}} \\
\leq& \frac{\delta}{2},
\end{aligned}$$

where the last inequality uses the definition of σ.

Bound of $\mathbb{P}(\tau_1 \leq T)$. Simple rearranging of Eq. (9) gives that, with probability $1-\frac{\delta}{2}$,

$$\begin{aligned}
&\frac{2(G+\sigma+\varepsilon)}{\eta_t}(f(\boldsymbol{w}_\tau) - f^*) \\
\leq& \sum_{t=1}^{\tau-1} \|\nabla f(\boldsymbol{w}_t)\|^2 + \frac{2(G+\sigma+\varepsilon)}{\eta_t}(f(\boldsymbol{w}_\tau) - f^*) \\
\leq& \frac{2(G+\sigma+\varepsilon)}{\eta_t}(f(\boldsymbol{w}_1) - f^*) + 4\sigma^2((1-\beta_1)T+1) + 20R^2\sigma^2\sqrt{2\sum_{t=2}^{T}\log(1/\delta)}.
\end{aligned}$$

Therefore, by dividing both sides of the above inequality, we obtain

$$\begin{aligned}
&f(\boldsymbol{w}_\tau) - f^* \\
\leq& (f(\boldsymbol{w}_1) - f^*) + \frac{\eta_t}{2(G+\sigma+\varepsilon)}4\sigma^2((1-\beta_1)T+1) + \frac{\eta_t}{2(G+\sigma+\varepsilon)}20R^2\sigma^2\sqrt{2\sum_{t=2}^{T}\log(1/\delta)} \\
\leq& \frac{G^2}{3(3L_0+4L_1G)} \\
=& F,
\end{aligned}$$

where the last inequality uses the definition of G.

Therefore, we have that

$$\mathbb{P}(\tau_1 \leq T) \leq \mathbb{P}(\text{Eq. 9 fails to hold}) \leq \frac{\delta}{2}.$$

The proof is completed by $\mathbb{P}(\tau \leq T) \leq \mathbb{P}(\tau_1 \leq T) + \mathbb{P}(\tau_2 \leq T) \leq \delta$.

□