EMA-Nesterov: Stabilizing Nesterov's Lookahead for Accelerated Deep Learning Optimization

Chung-Yiu Yau[1] **Dawei Li**[1*] **Athanasios Glentis**[1*]

Valentyn Boreiko[2†] **Hoi-To Wai**[3]

Mingyi Hong[1]

[1]University of Minnesota [2]Amazon AGI [3]The Chinese University of Hong Kong
[1]{cyau,li004678,glent007,mhong}@umn.edu
[2]valentyn.boreiko@gmail.com [3]htwai@se.cuhk.edu.hk

Abstract

Lookahead-based acceleration methods, such as Nesterov's momentum, are widely used in optimization, but they often become unreliable in deep learning training mainly due to stochastic gradient noise and non-convex loss landscapes. In particular, standard lookahead relies on short-horizon update signals (e.g., differences between consecutive iterates), which are inherently noisy and can lead to unstable extrapolation directions. This work revisits Nesterov's acceleration from a trajectory perspective and argues that effective acceleration in deep learning should harness the low-frequency trends of optimization trajectories rather than extrapolating noisy one-step updates. Leveraging this insight, we propose `EMA-Nesterov`, a simple modification that replaces the standard Nesterov's lookahead direction with an exponential moving average (EMA) of parameter updates. This yields a stabilized lookahead direction that captures and harnesses the evolving trend of the training trajectory through a low-pass filter, while remaining adaptive to progressive changes via the geometric weighting structure of EMA. We show that `EMA-Nesterov` retains a theoretical accelerated convergence rate in convex problems that is analogous to Nesterov's accelerated gradient method. Furthermore, we provide empirical evidence on language model pre-training to verify that `EMA-Nesterov` is broadly applicable across a range of fine-tuned base optimizers, including Adam, SOAP, Muon, as well as complex optimizers that achieve state-of-the-art performance on optimization benchmarks (NanoGPT). Compared to prior lookahead methods, `EMA-Nesterov` achieves better performance by avoiding the instability of short-horizon lookahead and the non-adaptivity of long-horizon lookahead.

1 Introduction

The growing scale of deep learning training has motivated a new generation of optimizers, such as Adam [Kingma and Ba, 2014], SOAP [Vyas et al., 2024], and Muon [Jordan et al., 2024], which improve optimization through adaptive preconditioning and normalization mechanisms that regularize stochastic gradient updates and stabilize training in highly irregular non-convex loss landscapes. Beyond the (base) optimizer design, there has also been renewed interest in incorporating

*Equally contributing second author.
†This work is unrelated to Valentyn Boreiko's position at Amazon.

Preprint.

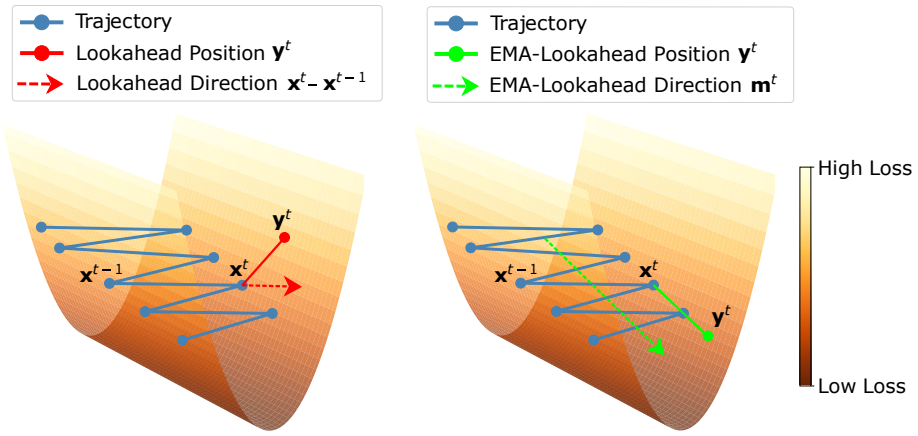


Figure 1: A hypothetical illustration of the training trajectory $(x_1, x_2, f(x_1, x_2)) \in \mathbb{R}^3$, assuming river valley loss landscape and a training trajectory under the WSD learning rate scheduling [Wen et al., 2024]. We compare the function value of the lookahead position at (Left) $\mathbf{y}^t = \mathbf{x}^t + \beta \cdot (\mathbf{x}^t - \mathbf{x}^{t-1})$ which arrives at a high loss position (i.e., $f(y_1^t, y_2^t) > f(x_1^t, x_2^t)$) and (Right) $\mathbf{y}^t = \mathbf{x}^t + \beta \cdot \mathbf{m}^t$ which progresses towards low loss position (i.e., $f(y_1^t, y_2^t) \leq f(x_1^t, x_2^t)$) where $\mathbf{m}^t = \gamma \mathbf{m}^{t-1} + (1 - \gamma)(\mathbf{x}^t - \mathbf{x}^{t-1})$.

trajectory-level momentum into deep learning optimization algorithms. In particular, recent works revisited Nesterov's accelerated gradient method [Nesterov, 1983], which extrapolates optimization trajectories using past iterate information (a.k.a. lookahead) to achieve accelerated convergence in convex optimization. Moreover, acceleration phenomena have also been theoretically established beyond convex settings: Hinder et al. [2020], Gupta and Wojtowytsch [2024] showed acceleration for certain non-convex problems, while Vaswani et al. [2019] proved accelerated convergence rates for SGD under over-parameterized regimes.

Generally speaking, lookahead acceleration constructs an extrapolated solution by moving beyond the current iterate along a direction induced by past optimization trajectories. In the classical Nesterov's accelerated gradient method, this extrapolation direction is given by the difference between consecutive iterates, i.e., $\mathbf{x}^t - \mathbf{x}^{t-1}$. One key insight behind why such lookahead direction can accelerate base optimizers is that extrapolating past iterates allows the algorithm to follow the persistent trend of the optimization trajectory, thereby moving more aggressively along stable descent directions. However, under the deep learning optimization trajectory, Nesterov's lookahead can be unstable due to oscillating optimizer update that overshoots the minima under pre-conditioning, as similarly observed in stochastic optimization [Wang et al., 2022]. In fact, such a lookahead has not been widely used in contemporary large-scale deep neural network training. Instead, exponential moving average of the past (stochastic) gradients has been a key component for modern optimizers such as Adam and Muon. Muon uses the name Nesterov in its implementation[3] but our careful inspection in Appdenix C.5 shows that it is actually an EMA of stochastic gradient with varying EMA rate, and has no connection to Nesterov's momentum or Nesterov's lookahead.

To gain intuition on the training trajectory of deep learning optimization and how lookahead directions affect such trajectories, in Figure 1 we consider a simplified river-valley landscape, in which during the stable phase of training, the iterate oscillates around the local minima, yet the training progresses along the "river" when using the WSD learning rate scheduling [Wen et al., 2024]. When Nesterov's lookahead is applied to this training trajectory, the extrapolation is steered toward the overshooting direction, causing the lookahead position to move into regions with higher objective values, see Figure 1 (left). This phenomenon is further validated in Figure 2, where we measure the validation loss at the lookahead positions during NanoGPT training for different lookahead step sizes β. We observe that increasing the lookahead step size leads to a sharp increase in the validation loss evaluated at the extrapolated positions. This suggests that short-horizon lookahead directions become unreliable in deep learning optimization and may produce extrapolated solutions that are unsuitable for constructing the next optimization step.

[3]According to Muon's open-sourced implementation at `https://github.com/KellerJordan/Muon`.

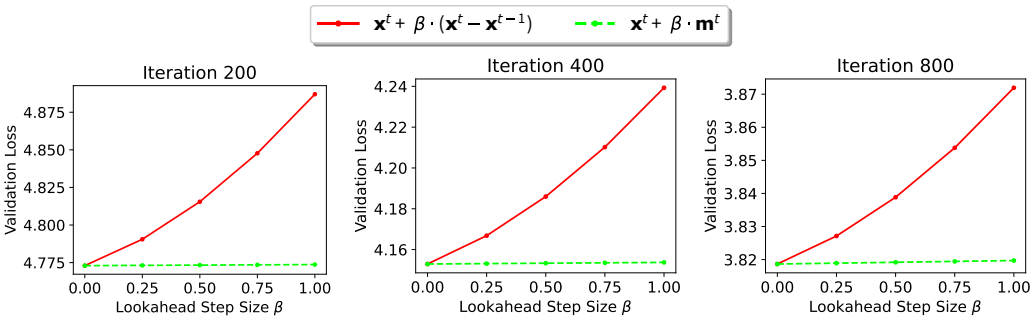


Figure 2: Under the NanoGPT setup, we measure the validation loss in different lookahead positions $\mathbf{y}^t$ at iterations $t = 200, 400, 800$ using (Solid Red) $\mathbf{y}^t = \mathbf{x}^t + \beta \cdot (\mathbf{x}^t - \mathbf{x}^{t-1})$ and (Dashed Green) $\mathbf{y}^t = \mathbf{x}^t + \beta \cdot \mathbf{m}^t$ where $\mathbf{m}^t = \gamma \mathbf{m}^{t-1} + (1-\gamma)(\mathbf{x}^t - \mathbf{x}^{t-1})$ with $\gamma = 0.995$. The usual lookahead leads to sharp increase in loss and EMA finds stabilized lookahead direction to avoid a sharp increase.

Prior Works	**Lookahead Direction**	**Lookahead Frequency**	**Adaptivity**
Nesterov's Accelerated Gradient [Nesterov, 1983]	$\mathbf{x}^t - \mathbf{x}^{t-1}$	1	✓
Pessimistic Lookahead [Zhang et al., 2019]	$-(\mathbf{x}^t - \mathbf{x}^{t-K})$	$1/K$	✗
NALA [Zuo et al., 2024]	$\mathbf{x}^t - \mathbf{x}^{t-K}$	$1/K$	✗
SNOO [Kallusky et al., 2025] (see App. A.1)	$\mathbf{x}^t - \mathbf{x}^{t-K}$ (Implicit)	$1/K$	✗
GPA [Defazio et al., 2025] (see App. A.2)	$\mathbf{z}^t - \mathbf{z}^{t-1}$ $+\beta(\mathbf{z}^t - \text{EMA}(\{\mathbf{z}^r\}_{r=0}^{t-1}))$ (Implicit)	1	✗
`EMA-Nesterov` **(Ours)**	$\text{EMA}(\{\mathbf{x}^r - \mathbf{x}^{r-1}\}_{r=1}^{t})$	1	✓

Table 1: Comparison between lookahead algorithms on the direction and frequency of lookahead. Adaptivity refers to whether the lookahead direction can adapt to changes in the trend of training trajectory, e.g., as illustrated in Figure 3.

Inspired by the above illustrations, we postulate that an appropriate way to accelerate deep learning model training is to *smooth* the lookahead directions through averaging the past lookahead directions. A line of recent works [Zhang et al., 2019, Douillard et al., 2023, Zuo et al., 2024, Kallusky et al., 2025] partially addressed this issue by incorporating an inner loop to accumulate (and thus average) the sequence of base optimizer updates and effectively performs long-horizon lookahead. These algorithms have demonstrated an empirical acceleration by adopting long-horizon lookahead. However, it is not fully understood how does the mechanism behind long-horizon lookahead enables acceleration. We draw insight from the perspective of low-pass filtering and provide a new understanding about long-horizon lookahead as a low-pass filter of the iterate updates.

This work aims at designing a stablized lookahead algorithm that adapts to the varying training trajectory by the use of exponential moving average (EMA) recursion, i.e., in the form of $\mathbf{m}^t = \gamma \mathbf{m}^{t-1} + (1-\gamma)(\mathbf{x}^t - \mathbf{x}^{t-1})$, $\gamma \in (0,1)$, to smooth the lookahead updates. As seen in Figure 1 (right), the EMA recursion finds the direction with less fluctuations; and the dashed green curves in Figure 2 shows that the lookahead position generated by the `EMA-Nesterov` algorithm does not suffer from the same performance degradation as the usual lookahead direction in solid red curves, therefore can be *reliably* used to build the subsequent steps. As a comparison, we summarize the existing lookahead algorithms according to their lookahead direction, lookahead frequency and adaptivity in Table 1 (also see Section 2 and 3). These recent approaches either suffer from non-adaptive lookahead or short-horizon oscillation due to their implicit algorithmic structure.

Our Contributions The contribution of this paper is summarized below.

- We introduce a trajectory-based perspective of lookahead acceleration for deep learning optimization. We show that classical short-horizon lookahead becomes unstable in deep learning optimization due to oscillatory high-frequency trajectory components, motivating a low-pass filtering view of acceleration.
- We propose `EMA-Nesterov`, a lightweight single-loop acceleration method that replaces long-horizon extrapolation with an exponential moving average of optimization trajectories, i.e., for a

given base optimizer function $\mathcal{A}_t : \mathbb{R}^d \to \mathbb{R}^d$, we perform

$$\mathbf{x}^{t+1} = \mathcal{A}_t(\mathbf{x}^t + \beta_t \mathbf{m}^t), \quad \mathbf{m}^{t+1} = \gamma \mathbf{m}^t + (1-\gamma)(\mathbf{x}^{t+1} - \mathbf{x}^t), \tag{1}$$

see Algorithm 1 for details. `EMA-Nesterov` exhibits a low-passed lookahead direction that can adapt to a varying trend in the training trajectory.

- We prove that `EMA-Nesterov` retains accelerated convergence rates in convex optimization. In particular, we establish accelerated rates of $O((1-\sqrt{(1-\gamma)/\kappa})^T)$ for strongly convex objectives with condition number $\kappa > 0$ and $O(1/T^2)$ for convex objectives.
- We demonstrate that `EMA-Nesterov` acts as a generic acceleration wrapper for modern neural network optimizers, consistently improving Adam, SOAP, Muon, and NorMuon on NanoGPT and Llama pre-training benchmarks. For example, in the NanoGPT benchmark[4], it improves the fine-tuned Muon optimizer on NanoGPT (162M) by achieving a 6% acceleration (for reaching the benchmark target of 3.28 validation loss). Our method outperforms existing lookahead algorithms under a comprehensive hyperparameter sweep for all baselines (see Figure 5).

More broadly, `EMA-Nesterov` can be viewed as a lightweight and optimizer-agnostic acceleration mechanism that can be combined with a broad class of modern neural network optimizers. Rather than redesigning optimizer-specific update rules, `EMA-Nesterov` operates directly at the trajectory level through stabilized lookahead extrapolation, making it naturally compatible with existing preconditioning, normalization, and momentum techniques.

2 Understanding Lookahead Acceleration in Deep Learning

This section studies the recent prior arts on lookahead acceleration and compares them with the classical Nesterov's lookahead algorithm from a trajectory perspective. We then identify key insights in the prior arts and suggest how they may inspire new algorithms.

To fix notations, we consider the stochastic optimization problem: for random sample ξ drawn from some distribution $\mathcal{D}$,

$$f^\star := \min_{\mathbf{x}\in\mathbb{R}^d} \; f(\mathbf{x}) := \mathbb{E}_{\xi\sim\mathcal{D}}[f(\mathbf{x};\xi)], \tag{2}$$

where $f(\mathbf{x};\xi)$ is differentiable w.r.t. $\mathbf{x}$. Here, we may denote $\mathcal{A}(\mathbf{x};\xi,\mathbf{s})$ as an optimizer function that takes the current model state $\mathbf{x}$ and returns the next model state, via consuming a sample of stochastic objective function gradient evaluated on $\mathbf{x}$ with randomness ξ, while the optimizer update is conditioned on its internal states $\mathbf{s}$. For example, $\mathcal{A}(\mathbf{x};\xi,\mathbf{s}) = \mathbf{x} - \alpha_t(\gamma\mathbf{s} + (1-\gamma)\nabla f(\mathbf{x};\xi))$ with learning rate $\alpha_t > 0$ produces the next model state using momentum SGD under gradient momentum rate $\gamma \in [0,1)$ and momentum buffer $\mathbf{s} \in \mathbb{R}^d$. For simplicity, in the rest of this paper, we shall use the notation $\mathcal{A}_t(\mathbf{x}) := \mathcal{A}(\mathbf{x};\xi^t,\mathbf{s}^t)$.

We begin our endeavor through studying the lookahead algorithm in Nichol et al. [2018], Zhang et al. [2019]. With a lookahead step size $\beta > 0$ and an optimizer $\mathcal{A}_t : \mathbb{R}^d \to \mathbb{R}^d$ (e.g., vanilla/momentum SGD, Muon, etc.) for (2), they proposed

$$\begin{aligned} &\textbf{for every } t = nK, n = 0, ..., \lfloor T/K \rfloor, \\ &\quad \tilde{\mathbf{x}}^{t+K} = \mathcal{A}_{t+K-1}(\cdots \mathcal{A}_{t+1}(\mathcal{A}_t(\mathbf{x}^t))), \quad \mathbf{x}^{t+K} = \mathbf{x}^t + \beta(\tilde{\mathbf{x}}^{t+K} - \mathbf{x}^t). \end{aligned} \tag{3}$$

For every K steps of optimizer update $\mathcal{A}_t$, the accumulated update $\tilde{\mathbf{x}}^{t+K} - \mathbf{x}^t$ is applied with a small step size $\beta \le 1$, which means that the algorithm's solution only interpolates between $\tilde{\mathbf{x}}^{t+K}$ and $\mathbf{x}^t$. The key property of (3) is that the long-horizon update direction $\tilde{\mathbf{x}}_{t+K} - \mathbf{x}_t$ averages optimization trajectories across multiple iterations, thereby suppressing high-frequency oscillations in stochastic optimization. As a result, the extrapolation direction becomes substantially more stable than the one-step lookahead direction used in classical Nesterov acceleration (similar to the case of Figure 1 (right)). Importantly, [Zhang et al., 2019, Proposition 2] demonstrated that the above scheme achieves a $\mathcal{O}(\beta)$-factor variance reduction. That being said, as demonstrated in [Zhang et al., 2019, Appendix A], (3) only exhibits a slightly improved empirical convergence speed over vanilla SGD in the quadratic setting, suggesting an 'unaccelerated' rate of $f(\mathbf{x}^T) - f^\star = \mathcal{O}((1-1/\kappa)^T)$, where κ is the condition number of $f(\mathbf{x})$.

[4] From `https://github.com/KellerJordan/modded-nanogpt`.

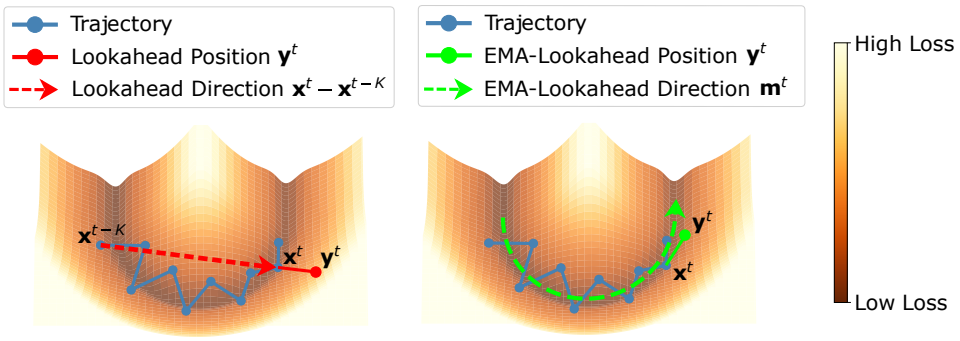


Figure 3: A hypothetical illustration of the training trajectory $(x_1, x_2, f(x_1, x_2)) \in \mathbb{R}^3$, assuming a U-shaped river valley loss landscape and a training trajectory under the WSD learning rate scheduling [Wen et al., 2024]. We compare the function value of the long-horizon lookahead position at (Left) $\mathbf{y}^t = \mathbf{x}^t + \beta \cdot (\mathbf{x}^t - \mathbf{x}^{t-K})$ which arrives at a high loss position and (Right) $\mathbf{y}^t = \mathbf{x}^t + \beta \cdot \mathbf{m}^t$ which progresses towards the trend of training trajectory.

For further insights, we compare (3) to the classical Nesterov's accelerated gradient method by reducing (3) to the special case with $K = 1$. Notice that in this case, (3) is equivalent to:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \beta(\mathcal{A}_t(\mathbf{x}^t) - \mathbf{x}^t) = \underbrace{\mathcal{A}_t(\mathbf{x}^t) + (\beta - 1)(\mathcal{A}_t(\mathbf{x}^t) - \mathbf{x}^t)}_{\text{lookahead position}}. \tag{4}$$

On the other hand, Nesterov's lookahead algorithm [Nesterov, 1983] can be written as:

$$\mathbf{x}^{t+1} = \mathcal{A}_t(\underbrace{\mathbf{x}^t + \beta'(\mathbf{x}^t - \mathbf{x}^{t-1})}_{\text{lookahead position}}), \quad \beta' > 0. \tag{5}$$

It is known that (5) achieves an accelerated convergence rate: with an appropriate β' and $\mathcal{A}_t$ utilizing deterministic gradient, one has $f(\mathbf{x}^T) - f^\star = \mathcal{O}((1 - \sqrt{1/\kappa})^T)$ for strongly convex f.

Comparing (4) and (5) on their corresponding lookahead position, we note that (4) can be regarded as a *pessimistic* lookahead algorithm since it uses the reversed Nesterov's lookahead direction when $\beta - 1$ becomes negative. This also implies that (4) may not be able to achieve acceleration when using a conservative lookahead step size $\beta \leq 1$. Meanwhile, Nesterov's lookahead (5) takes an *optimistic* lookahead direction as it fully trusts the update direction of past iteration with any $\beta' > 0$.

3 Stabilized Nesterov's Lookahead

We will develop `EMA-Nesterov` using the insights from the trajectory perspective, and to present the convergence properties of `EMA-Nesterov`. The discussion from the previous section motivates us to look at an optimistic lookahead algorithm with K-step long-horizon lookahead:

$$\begin{aligned} &\textbf{for every } t = nK, n = 0, ..., \lfloor T/K \rfloor, \\ &\quad \mathbf{x}^{t+K} = \mathcal{A}_{t+K-1}(\cdots \mathcal{A}_{t+1}(\mathcal{A}_t(\mathbf{x}^t + \beta' \mathbf{b}^t))), \quad \mathbf{b}^{t+K} = \mathbf{x}^{t+K} - \mathbf{x}^t. \end{aligned} \tag{6}$$

The same algorithm structure is used explicitly in NALA [Zuo et al., 2024] or implicitly in SNOO [Kallusky et al., 2025]. Particularly, Zuo et al. [2024] attempted to combine the advantages of Nesterov's accelerated gradient method and long-horizon lookahead, yielding an algorithm similar to (6) where $\mathcal{A}_t$ is gradient descent for $t \bmod K = 0$ and $\mathcal{A}_t$ is a pre-conditioned stochastic algorithm for $t \bmod K \neq 0$. Later, Kallusky et al. [2025] treated the long-horizon lookahead direction as a pseudo-gradient for Nesterov's momentum algorithm [Sutskever et al., 2013], yielding an algorithm that is fundamentally similar to (6) as discussed in Appendix A.1. Defazio et al. [2025] developed a single-loop algorithm under the primal-averaging framework which implicitly approximated a mixture of short-horizon and long-horizon lookahead as discussed in Section A.2, where such mixture implies unstable oscillation from short-horizon lookahead. We summarize the above algorithms from the lookahead perspective in Table 1.

To fully understand the long-horizon lookahead direction $\mathbf{b}^t$ in (6), we rewrite $\mathbf{b}^t$ in terms of one-step iterate update $\Delta \mathbf{x}^t := \mathbf{x}^t - \mathbf{x}^{t-1}$ as

$$\mathbf{b}^t = \mathbf{x}^t - \mathbf{x}^{t-K} = K \cdot \tfrac{1}{K} \textstyle\sum_{i=t-K+1}^{t} \Delta \mathbf{x}^i, \; t \geq K. \tag{7}$$

As seen, $\mathbf{b}^t$ is the unweighted combination of the past K iterations of (one-step) lookahead directions, i.e., it is the output of an order-K moving average *low-pass filter* over the sequence $\{\Delta\mathbf{x}^i\}_{i\geq 1}$. The low-pass filter suppresses high-frequency fluctuations in consecutive iterate updates. This aligns with our prior understanding that long-horizon lookahead can reduce fluctuation in the lookahead updates. However, we argue that such unweighted combination induces a lookahead direction that is not adaptive to the local geometry. In particular, when the trend of training trajectory evolves over time, the unweighted average (7) with large K can only capture a global trend, which is not suitable for local extrapolation, as illustrated in the left figure of Figure 3.

3.1 Exponential Moving Average Lookahead

Alternative to long-horizon lookahead, we replace the lookahead direction with a low-pass filter that is adaptive to the recent signals — the exponential moving average (EMA) filter: for $t \geq 1$ and let $\gamma \in [0, 1)$,

$$\mathbf{m}^{t+1} = \gamma\mathbf{m}^t + (1-\gamma)\Delta\mathbf{x}^{t+1} =: \mathrm{EMA}_\gamma(\{\Delta\mathbf{x}^i\}_{i=1}^{t+1}). \tag{8}$$

In particular, one has $\mathrm{EMA}_\gamma(\{\Delta\mathbf{x}^i\}_{i=1}^{t+1}) = (1-\gamma)\sum_{i=1}^{t+1}\gamma^{t+1-i}\Delta\mathbf{x}^i$. Notice that (8) is a low-pass filter with the geometric coefficients $\{\gamma^i\}_{i\geq 1}$ that is anticipated to filter out the high-frequency fluctuations while putting larger weights on recent lookahead signals $\Delta\mathbf{x}^i$, i.e., it aligns more with the recent trend of $\Delta\mathbf{x}^i$ as illustrated in the right figure of Figure 3. The low-pass behavior of EMA filter can be analyzed by its transfer function $H(z) = \frac{1-\gamma}{1-\gamma z^{-1}}$ on the input signal $z = \exp(i\omega)$ of frequency $\omega \in [0, \pi]$, where the output energy $|H(\exp(i\omega))|$ is decreasing as ω increases from 0 (low frequency) to π (high frequency). We replace the lookahead direction of Nesterov's algorithm by $\mathbf{m}^t$ and propose `EMA-Nesterov` in Algorithm 1.

Algorithm 1 `EMA-Nesterov` Algorithm

1: **Input:** Initial model $\mathbf{x}^0$, lookahead step size $\beta_t \geq 0$, exponential moving average rate $\gamma \in [0, 1)$, base optimizer $\mathcal{A}_t(\cdot)$, iterations $T > 0$.
2: **Initialization:** $\mathbf{m}^0 = \mathbf{0}$.
3: **for** $t = 0, ..., T-1$ **do**
4: $\quad\mathbf{x}^{t+1} = \mathcal{A}_t(\mathbf{x}^t + \beta_t\mathbf{m}^t)$
5: $\quad\mathbf{m}^{t+1} = \gamma\mathbf{m}^t + (1-\gamma)(\mathbf{x}^{t+1} - \mathbf{x}^t)$
6: **end for**
7: **Output:** Last iterate solution $\mathbf{x}^T$.

Line 4 of Algorithm 1 performs an optimization step from the lookahead position $\mathbf{x}^t + \beta_t\mathbf{m}^t$, using the EMA lookahead direction $\mathbf{m}^t$. With $\gamma > 0$, Algorithm 1 accumulates a stabilized lookahead direction in $\mathbf{m}^t$ that effectively follows the trend of optimization trajectory, without being perturbed by the oscillation that enters the filter as high frequency noise.

Comparison to Existing Algorithms. We can directly compare the differences between existing lookahead algorithms by reducing them into the lookahead form. For example, SNOO [Kallusky et al., 2025] can be reduced to the K-step long-horizon lookahead in (6) (see Appendix A.1 for the derivation) and GPA [Defazio et al., 2025] can be reduced to the following lookahead algorithm (see Appendix A.2 for the derivation):

$$\begin{aligned}&\textbf{for every } t = 0, ..., T-1,\\ &\quad\mathbf{z}^{t+1} = \mathbf{z}^t - \gamma_p\nabla f(\mathbf{y}^t; \xi^t), \quad \mathbf{x}^{t+1} = \beta\mathbf{x}^t + (1-\beta)\mathbf{z}^{t+1},\\ &\quad\mathbf{y}^{t+1} = \mathbf{y}^t + (1-\beta)(\mathbf{z}^{t+1} - \mathbf{z}^t) + \beta(1-\beta)(\mathbf{z}^{t+1} - \mathbf{x}^t).\end{aligned} \tag{9}$$

The lookahead direction of GPA is determined by $\mathbf{z}^{t+1} - \mathbf{z}^t$ and $\mathbf{z}^{t+1} - \mathbf{x}^t$, where the former is vulnerable to high-frequency fluctuations while the latter is effectively a long-horizon lookahead with $\mathbf{x}^t$ tracking the past trajectory of $\mathbf{z}^t$. A direct comparison on the numerical performance is provided in Figure 5, where our algorithm further improved GPA by 0.2 validation perplexity on NanoGPT. Theoretically, the convergence analysis of Defazio et al. [2025] does not guarantee accelerated convergence rate. In contrary, our Algorithm 1 adopts a matching design with Nesterov's accelerated algorithm which enables us to carry out convergence analysis with accelerated rate for convex optimization.

3.2 Convergence Analysis

In this subsection, we show that due to the optimistic design, `EMA-Nesterov` attains the same accelerated convergence as the classical Nesterov's accelerated gradient algorithm when specialized to (2) with (strongly) convex objective function and the optimizer utilizes exact gradients of $f(\mathbf{x})$. We focus on the special case of Algorithm 1 with the optimizer specified as the vanilla gradient descent:

$$\mathcal{A}(\mathbf{x}; \xi; \mathbf{s}) = \mathbf{x} - \alpha \nabla f(\mathbf{x}). \tag{10}$$

We notice that Algorithm 1 with (10) and EMA rate $\gamma = 0$ reduces to the classical Nesterov's accelerated gradient algorithm; yet when $\gamma > 0$, the two algorithms exhibit different lookahead behavior. Despite their similar algorithmic structure, we notice that extending the analysis in [Nesterov, 1983] to Algorithm 1 is highly non-trivial, especially for achieving the accelerated convergence rate(s). In this section, we proceed with the convergence guarantee of the above algorithm under a standard μ-strongly convex, L-smooth assumption for (2) as defined below.

Assumption 1. For any $L > \mu \geq 0$, we assume the objective function f is μ-strongly convex and has L-Lipschitz gradient such that $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L\|\mathbf{x} - \mathbf{y}\|_2 \ \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$. We define $\kappa := L/\mu$ as the condition number of f.

We remark that as the deep learning training problems are typically non-convex, the convexity assumption above may not be satisfied in practice. Nonetheless, they provide insights and theoretical support on the advantages of `EMA-Nesterov`, and the analysis can be of independent interests for studying perturbed Nesterov's accelerated gradient. Our results are further split into two cases.
Strongly Convex $f(\mathbf{x})$. When $\mu > 0$, we observe that

Theorem 1. *Under Assumption 1, by choosing $\alpha = 1/L$ and $\beta_t = \frac{\left(\sqrt{1-\gamma} - \sqrt{1/\kappa}\right)^2}{(1-\gamma)(1-1/\kappa)}$ for all $t = 0, ..., T-1$, the solution of Algorithm 1 with* (10) *satisfies*

$$f(\mathbf{x}^T) - f(\mathbf{x}^\star) = \mathcal{O}\left((1 - \sqrt{(1-\gamma)/\kappa})^T\right) \quad \textit{for any } \gamma \in [0, 1 - 1/\kappa),\ T > 0. \tag{11}$$

See Appendix B.1 for the proof. The above guarantee reduces to the classical accelerated rate of $(1 - \sqrt{1/\kappa})^T$ of Nesterov's algorithm when $\gamma = 0$. Our analysis generalizes to `EMA-Nesterov` when $\gamma > 0$ and the acceleration upon the gradient descent rate of $(1 - 1/\kappa)^T$ happens when

$$\sqrt{(1-\gamma)/\kappa} > 1/\kappa \Leftrightarrow \gamma < 1 - 1/\kappa. \tag{12}$$

Non-strongly Convex $f(\mathbf{x})$. When $\mu = 0$, we observe that

Theorem 2. *Under Assumption 1 with $\mu = 0$, by choosing $\alpha = 1/L$ and $\beta_t = \frac{c_t - \gamma c_{t+1}}{1 + (1-\gamma)c_{t+1}}$ for $c_t = 1 + \frac{\gamma}{4(1-\gamma)} + \frac{t}{4}$, the solution of Algorithm 1 with* (10) *satisfies*

$$f(\mathbf{x}^T) - f(\mathbf{x}^\star) = \mathcal{O}\left(L/((1-\gamma)^3 T^2)\right) \quad \textit{for any } \gamma \in [0, 1),\ T > 0. \tag{13}$$

See Appendix B.2 for the proof. Unlike the strongly convex case, Theorem 2 guarantees an accelerated convergence rate of $\mathcal{O}(1/T^2)$ for all $\gamma \in [0, 1)$, i.e., it converges faster than gradient descent of rate $\mathcal{O}(1/T)$ on convex objective function for large enough $T = \Omega(1/(1-\gamma)^3)$.

3.3 Practical Implementation of Lookahead Step Size Scheduling

It is generally understood that the early stage of deep learning optimization is unstable. As a remedy, we will schedule the lookahead step size β_t by three stages: 1) the warm-up stage where $\beta_t = 0$ for $t \leq T_w$; 2) the lookahead stage where $\beta_t = \beta \cdot \alpha_t / (\max_k \alpha_k)$ for $T_w < t \leq T_r$ such that the lookahead step size relatively follows $\alpha_t / (\max_k \alpha_k)$ the learning rate schedule; and 3) the early rest stage where $\beta_t = 0$ for $t > T_r$. The early rest stage is crucial to allow stable convergence to sharp local minima during the learning rate decay phase, as the EMA lookahead direction cannot quickly adapt to the local geometry near sharp minima. This is further validated by the observation in Figure 8, where the EMA lookahead direction leads to sharp increase in loss in later stage of the training. In our experiments, the scheduling of β_t (T_w and T_r) roughly follows the learning rate schedule as the learning rate schedule implicitly controls when stable training occurs. Details of the actual schedule used in our experiments can be found in Appendix C.4.

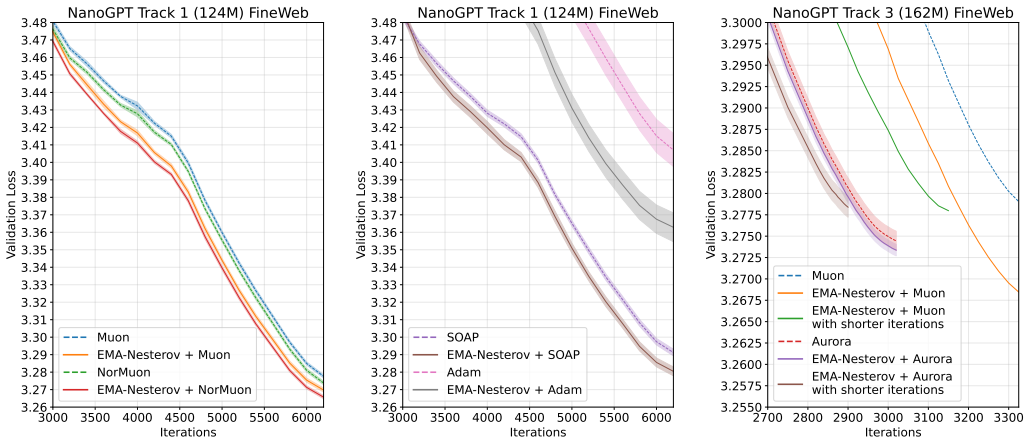


Figure 4: Training `EMA-Nesterov` with different base optimizers (Muon, NorMuon, SOAP and Adam) on NanoGPT Track 1 (124M) benchmark (Left, Middle) and with SOTA base optimizers (as of 05/18/2026) on NanoGPT Track 3 (162M) (Right). Results are reported by mean-variance plots of ± 1 standard deviation on multiple seeded runs for each algorithm.

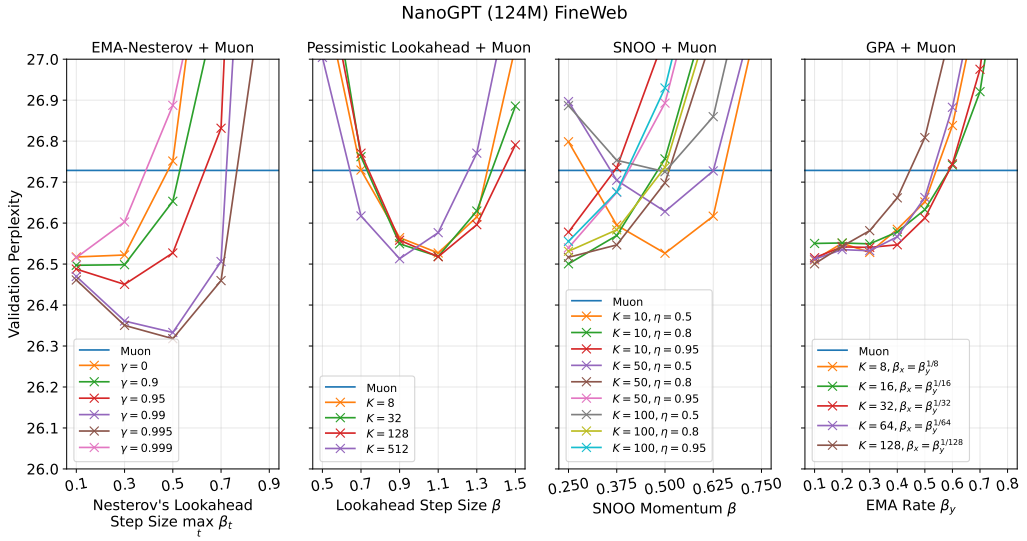


Figure 5: Comparison on accelerating Muon base optimizer using `EMA-Nesterov`, pessimistic lookahead, GPA and SNOO on NanoGPT (124M) benchmark. The same vertical axis spans across four plots horizontally. Among all hyperparameter configurations, `EMA-Nesterov` + Muon with $\gamma = 0.995$, $\max_t \beta_t = 0.5$ achieves the lowest validation perplexity.

4 Experiments

We validate the performance of Algorithm 1 on two language model pre-training settings: **(I)** training the NanoGPT models on FineWeb dataset[5] and **(II)** training Llama models on C4-en dataset. Unless otherwise specified, all settings use a training budget of around 20 tokens-per-parameter as suggested by Hoffmann et al. [2022]. The main experiments are performed on 8× NVIDIA H200 GPUs (141 GB memory). Our algorithm is implemented as a lightweight wrapper and does not impose extra overhead on the per-iteration time, we thus focus on comparing different algorithms' performance after a fixed number of training iterations. The base optimizers' hyperparameters are tuned without `EMA-Nesterov` and are unchanged when adding `EMA-Nesterov` externally. Our experiment is open-sourced at `https://github.com/OptimAI-Lab/ema-nesterov`.

[5]124M model is based on `https://github.com/KellerJordan/modded-nanogpt/blob/master/records/track_1_short/2024-10-10_Muon/eb5659d0-fb6a-49e5-a311-f1f89412f726.txt` while 162M model is based on `https://github.com/KellerJordan/modded-nanogpt/tree/master/records/track_3_optimization`.

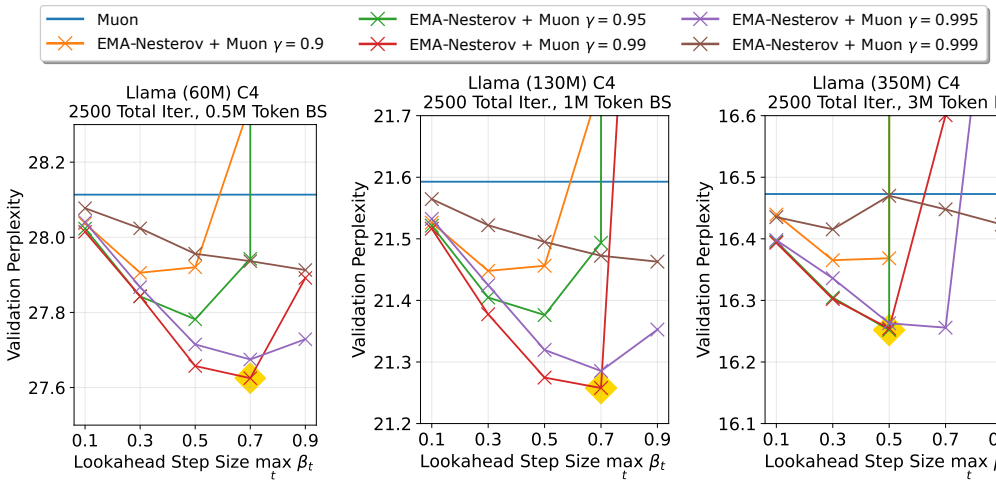


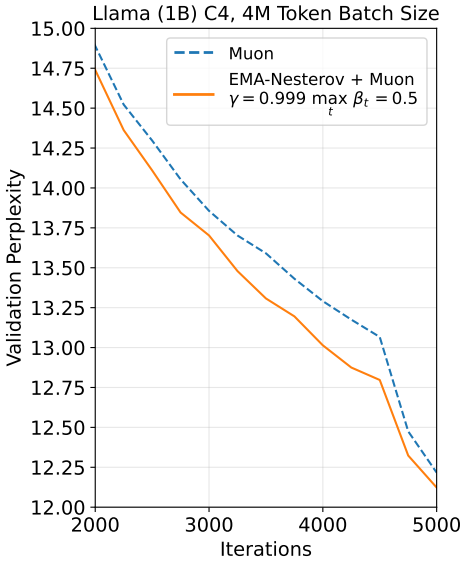
Figure 6: Sensitivity of hyperparameters for `EMA-Nesterov`-Muon across different model sizes on the Llama setup. The value of $\max_t \beta_t$ is searched in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ across the horizontal axis and the value of γ is searched in $\{0.9, 0.95, 0.99, 0.995, 0.999\}$ across different colors.

Accelerating Common Optimizers. The first experiment aims to present the acceleration effects offered by applying `EMA-Nesterov` on common pre-conditioned optimizers as $\mathcal{A}_t$ — including Adam [Kingma and Ba, 2014], SOAP [Vyas et al., 2024], Muon [Jordan et al., 2024] and NorMuon [Li et al., 2025]. In Figure 4, we apply `EMA-Nesterov` with the best pair of hyperparameters $(\gamma, \max_t \beta_t)$ (selected from Figure 5) onto different choices of base optimizer $\mathcal{A}_t$. Also from the figure, `EMA-Nesterov` improves the performances of these common optimizers ranging from 0.208 perplexity drop (on Muon) to 1.305 perplexity drop (on Adam). For the larger model size of NanoGPT, we similarly apply `EMA-Nesterov` onto the SOTA optimization algorithms such as Muon and Aurora [Dewulf et al., 2026] and observe acceleration upon the base optimizers. Note that to achieve the maximal acceleration for arriving at a target loss value (e.g., the benchmark target of 3.28 validation loss in Figure 4), we adjust the total iterations accordingly as shown in the right figure of Figure 4. These evidences show that `EMA-Nesterov` can offer acceleration to existing fine-tuned optimizers.

Comparison with Existing Lookahead Algorithms. Our second experiment compares `EMA-Nesterov` to existing baseline lookahead algorithms. For fairness, we also perform a hyperparameter search on the NanoGPT setting for pessimistic lookahead [Zhang et al., 2019], SNOO [Kallusky et al., 2025] and GPA [Defazio et al., 2025] according to the corresponding suggested range of hyperparameters. As seen in Figure 5, `EMA-Nesterov` achieved the largest improvement among the other lookahead algorithms under the Muon base optimizer $\mathcal{A}_t$.

Model Scaling. This experiment investigates how sensitive the performance of `EMA-Nesterov` is on choosing the hyperparameters γ and $\max_t \beta_t$ for different model sizes. We perform a set of hyperparameter sweep over 60M, 130M and 350M parameter Llama models on C4 pre-training. The batch sizes are scaled up accordingly to match the total training budget of $20\times$ tokens-per-parameter. From Figure 6, when scaling to larger models, the optimal hyperparameters shift predictably. As the model size increases, the optimal γ tends to increase and the optimal $\max_t \beta_t$ tends to decrease. By this set of results, default choices of $\max_t \beta_t = 0.5$, $\gamma = 0.99$ for model size $\leq$ 350M and $\gamma = 0.995$ for model size $>$ 350M are recommended for large batch setting. In Appendix C.3, we provide more hyperparameter sweeps to illustrate the hyperparameter sensitivity in other settings and our default hyperparameters deliver near-optimal performance. Finally, we scale up to 1B model in the left figure of Table 2, and summarizes the last iteration validation perplexity for different Llama models in Table 2.

Data / Batch Size Scaling. To stress test the performance of `EMA-Nesterov` on more realistic setting, we increase the training budget beyond $20\times$ tokens-per-parameter in Figure 7. Under the optimal hyperparameters from Figure 6, we observe that `EMA-Nesterov` consistently improved the base optimizer in larger batch size setting, validating the practical advantage of `EMA-Nesterov` for large-scale data-parallel pre-training. We also observe that a $(\cdot)^{1/4}$ learning rate scaling rule delivers better performance than the square root scaling rule (see comparison in Figure 9).



Algorithm	Val. Ppl.	Iter.	Token Batch Size	γ	$\max_t \beta_t$
60M Model					
Muon	28.11	2500	0.5M	-	-
EMA-Nesterov-Muon	27.62			0.99	0.7
Muon	24.09	2500	4M	-	-
EMA-Nesterov-Muon	23.67			0.99	0.7
Muon	28.21	5000	0.26M	-	-
EMA-Nesterov-Muon	27.27			0.995	0.7
130M Model					
Muon	21.59	2500	1M	-	-
EMA-Nesterov-Muon	21.26			0.99	0.7
Muon	18.39	2500	8M	-	-
EMA-Nesterov-Muon	18.11			0.99	0.7
Muon	21.15	5000	0.5M	-	-
EMA-Nesterov-Muon	20.85			0.995	0.5
Muon	17.78	5000	4M	-	-
EMA-Nesterov-Muon	17.59			0.995	0.5
350M Model					
Muon	16.12	2500	3M	-	-
EMA-Nesterov-Muon	15.86			0.99	0.5
Muon	14.01	2500	25M	-	-
EMA-Nesterov-Muon	13.82			0.99	0.5
1B Model					
Muon	12.22	5000	4M	-	-
EMA-Nesterov-Muon	12.12			0.999	0.5

Table 2: (Left) Training curve of Muon and EMA-Nesterov-Muon on Llama 1B for 5000 iterations. (Right) Validation perplexities of different sizes of Llama models.

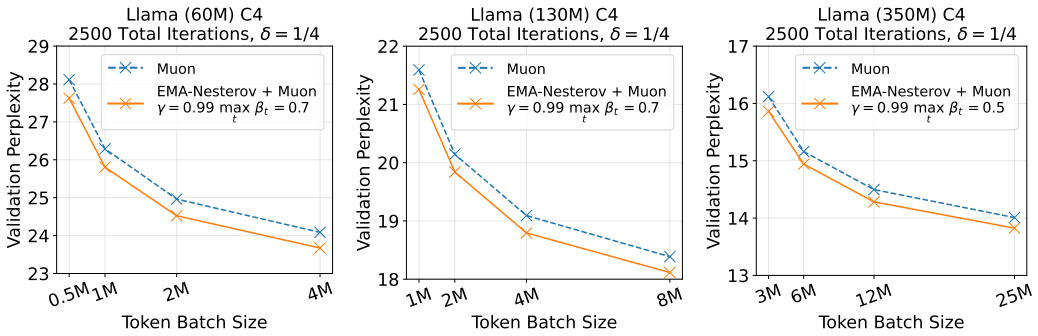


Figure 7: Data scaling to total training budgets of $\geq \{20\times, 40\times, 80\times, 160\times\}$ tokens-per-parameter with increased token batch size and fixed 2500 total iterations. The learning rates are adapted under the power scaling rule: for tuned learning rate α on batch size B, the adapted learning rate α' for larger batch size $B' > B$ is $\alpha' = \alpha \cdot (B'/B)^{\delta}$ with the learning rate scaling exponent $\delta = 1/4$.

5 Conclusion

We proposed a lightweight wrapper algorithm called EMA-Nesterov that is motivated by the low-pass filtering of lookahead directions in deep learning setting. EMA-Nesterov utilizes a stabilized lookahead direction by filtering the high frequency oscillation in the training trajectory. We show that EMA-Nesterov achieved acceleration in practical deep learning problems and theoretical convex problems.

References

Aaron Defazio, Konstantin Mishchenko, Parameswaran Raman, Hao-Jun Michael Shi, and Lin Xiao. Smoothing diloco with primal averaging for faster training of llms. *arXiv preprint arXiv:2512.17131*, 2025.

Alec Dewulf, Dhruv Pai, Li Yang, Ashley Zhang, and Ben Keigwin. Aurora: A leverage-aware optimizer for rectangular matrices. 2026. URL https://tilderesearch.com/blog/aurora.

Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

Kanan Gupta and Stephan Wojtowytsch. Nesterov acceleration in benignly non-convex landscapes. *arXiv preprint arXiv:2410.08395*, 2024.

Oliver Hinder, Aaron Sidford, and Nimit Sohoni. Near-optimal methods for minimizing star-convex functions and beyond. In *Conference on learning theory*, pages 1894–1938. PMLR, 2020.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *Advances in neural information processing systems*, 30, 2017.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan. github. io/posts/muon*, 6(3):4, 2024.

Dominik Kallusky, Vinay Rao, Vishal Nandavanam, and Hao-Jun Michael Shi. Snoo: Step-k nesterov outer optimizer-the surprising effectiveness of nesterov momentum applied to pseudo-gradients. *arXiv preprint arXiv:2510.15830*, 2025.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Yurii Nesterov. A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. In *Dokl akad nauk Sssr*, volume 269, page 543, 1983.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. pmlr, 2013.

Sharan Vaswani, Francis Bach, and Mark Schmidt. Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *The 22nd international conference on artificial intelligence and statistics*, pages 1195–1204. PMLR, 2019.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

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

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*, 2024.

Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, 32, 2019.

Xuan Zuo, Hui-Yan Li, Shan Gao, Pu Zhang, and Wan-Ru Du. Nala: a nesterov accelerated look-ahead optimizer for deep learning. *PeerJ Computer Science*, 10:e2167, 2024.

A Reduction of Existing Algorithms into Lookahead Form

In this section, we include the missing proofs for reducing the existing algorithms into its lookahead form.

A.1 SNOO

SNOO [Kallusky et al., 2025] (originated from DiLoCo [Douillard et al., 2023] in federated learning) utilizes the K-step base optimizer update as a pseudo-gradient oracle for Nesterov's algorithm. In particular, for any base optimizer $\mathcal{A}_t(\cdot)$ and $K > 0$ inner steps, we can reduce the SNOO algorithm as the following lookahead algorithm with lookahead step size $\beta > 0$ by the derivation in Proof 1.

$$\begin{aligned}
&\textbf{for every } t = nK, n = 0, ..., \lfloor T/K \rfloor, \\
&\quad \tilde{\theta}^{t+K} = \mathcal{A}_{t+K-1}(\cdots \mathcal{A}_{t+1}(\mathcal{A}_t(\mathbf{x}^t - \beta \mathbf{b}^t))) \\
&\quad \mathbf{b}^{t+K} = \mathbf{x}^t - \tilde{\theta}^{t+K} \\
&\quad \mathbf{x}^{t+K} = \tilde{\theta}^{t+K} \\
&\textbf{end for;} \quad \textbf{return } \mathbf{x}^T - \beta \mathbf{b}^T.
\end{aligned} \tag{14}$$

Proof 1. The original SNOO algorithm can be summarized as follows:

$$\begin{aligned}
&\textbf{for every } t = nK, n = 0, ..., \lfloor T/K \rfloor, \\
&\quad \tilde{\theta}^{t+K} = \mathcal{A}_{t+K-1}(\cdots \mathcal{A}_{t+1}(\mathcal{A}_t(\theta^t))) \\
&\quad \mathbf{b}^{t+K} = \beta \mathbf{b}^t + (\theta^t - \tilde{\theta}^{t+K}) \\
&\quad \theta^{t+K} = \theta^t - \eta(\beta \mathbf{b}^{t+K} + (\theta^t - \tilde{\theta}^{t+K})) \\
&\textbf{end for;} \quad \textbf{return } \theta^T.
\end{aligned} \tag{15}$$

with momentum $\beta > 0$ and Nesterov's learning rate $\eta > 0$. The base optimizer $\mathcal{A}_t$ is applied for K times consecutively from θ^t to produce $\tilde{\theta}^{t+K}$. However, it is non-trivial to understand the effect of Nesterov's momentum that is updated only after every K stochastic gradient samples. To simplify the algorithm, we apply a change of variable $\theta^t = \mathbf{x}^t - \beta \mathbf{b}^t$ to revert the transformation of Sutskever et al. [2013] into the following lookahead formulation:

$$\begin{aligned}
&\textbf{for every } t = nK, n = 0, ..., \lfloor T/K \rfloor, \\
&\quad \tilde{\theta}^{t+K} = \mathcal{A}_{t+K-1}(\cdots \mathcal{A}_{t+1}(\mathcal{A}_t(\mathbf{x}^t - \beta \mathbf{b}^t))) \\
&\quad \mathbf{b}^{t+K} = \mathbf{x}^t - \tilde{\theta}^{t+K} \\
&\quad \mathbf{x}^{t+K} = \mathbf{x}^t + (\beta - \eta\beta - \eta)\mathbf{b}^{t+K} - (\beta - \eta\beta)\mathbf{b}^t \\
&\textbf{end for;} \quad \textbf{return } \mathbf{x}^T - \beta \mathbf{b}^T.
\end{aligned} \tag{16}$$

Then, when $\eta = 1$, we have $\mathbf{x}^{t+K} = \mathbf{x}^t - \mathbf{b}^{t+K}$ and SNOO is reduced to (14).

□

A.2 GPA

GPA [Defazio et al., 2025] adopts the primal-averaging framework which approximate Nesterov's lookahead. By the derivation in Proof 2, we can reduce GPA into the following lookahead algorithm for learning rate $\gamma_p > 0$ and EMA rate $\beta > 0$:

$$\begin{aligned}
&\textbf{for every } t = 0, ..., T-1, \\
&\quad \mathbf{z}^{t+1} = \mathbf{z}^t - \gamma_p \nabla f(\mathbf{y}^t; \xi^t) \\
&\quad \mathbf{x}^{t+1} = \beta \mathbf{x}^t + (1-\beta)\mathbf{z}^{t+1} \\
&\quad \mathbf{y}^{t+1} = \mathbf{y}^t + (1-\beta)(\mathbf{z}^{t+1} - \mathbf{z}^t) + \beta(1-\beta)(\mathbf{z}^{t+1} - \mathbf{x}^t) \\
&\textbf{end for;} \quad \textbf{return } \mathbf{x}^T.
\end{aligned} \tag{17}$$

Proof 2. We summarize the GPA algorithm as follows:

$$\begin{aligned}
&\textbf{for every } t = 0, ..., T-1, \\
&\quad \mathbf{y}^t = \beta_y \mathbf{x}^t + (1-\beta_y)\mathbf{z}^t \\
&\quad \mathbf{z}^{t+1} = \mathbf{z}^t + \mathcal{A}_t(\mathbf{y}^t) - \mathbf{y}^t \\
&\quad \mathbf{x}^{t+1} = \beta_x \mathbf{x}^t + (1-\beta_x)\mathbf{z}^{t+1} \\
&\textbf{end for;} \quad \textbf{return } \mathbf{x}^T.
\end{aligned} \tag{18}$$

for momentum $\beta_x \in [0,1), \beta_y \in [0,1]$. Now apply [Defazio et al., 2025, Proposition 1] to obtain the following equivalent algorithm when $\mathcal{A}_t(\mathbf{y}) = \mathbf{y} - \gamma_p \nabla f(\mathbf{y}; \xi^t)$:

$$\begin{aligned}
&\textbf{for every } t = 0, ..., T-1, \\
&\quad \mathbf{z}^{t+1} = \mathbf{z}^t - \gamma_p \nabla f(\mathbf{y}^t; \xi^t) \\
&\quad \mathbf{x}^{t+1} = \beta \mathbf{x}^t + (1-\beta)\mathbf{z}^{t+1} \\
&\quad \mathbf{b}^t = \frac{1}{\gamma_m}(\mathbf{x}^t - \mathbf{x}^{t+1}) = \beta \mathbf{b}^{t-1} + \nabla f(\mathbf{y}^t; \xi^t) \\
&\quad \mathbf{y}^{t+1} = \mathbf{y}^t - \gamma_m(\beta \mathbf{b}^t + \nabla f(\mathbf{y}^t; \xi^t)) \\
&\textbf{end for;} \quad \textbf{return } \mathbf{x}^T.
\end{aligned} \tag{19}$$

with $\beta = \beta_x = \beta_y$ and $\gamma_m = (1-\beta)\gamma_p$. To obtain (17), we first prove that $\mathbf{x}^t = \mathbf{z}^t + \gamma_p \beta \mathbf{b}^{t-1}$ because

$$\mathbf{z}^t + \gamma_p \beta \mathbf{b}^{t-1} = \mathbf{z}^t + \frac{\beta}{1-\beta}(\mathbf{x}^{t-1} - \mathbf{x}^t) \tag{20}$$

$$= \mathbf{z}^t + \frac{\beta}{1-\beta}(\mathbf{x}^{t-1} - \beta \mathbf{x}^{t-1} - (1-\beta)\mathbf{z}^t) \tag{21}$$

$$= \beta \mathbf{x}^{t-1} + (1-\beta)\mathbf{z}^t = \mathbf{x}^t. \tag{22}$$

Then, to simplify $\mathbf{y}^{t+1}$, we observe that

$$\mathbf{b}^t = \beta \mathbf{b}^{t-1} + \nabla f(\mathbf{y}^t; \xi^t) = \beta \mathbf{b}^{t-1} + \frac{1}{\gamma_p}(\mathbf{z}^t - \mathbf{z}^{t+1}) = \frac{1}{\gamma_p}(\mathbf{x}^t - \mathbf{z}^{t+1}), \tag{23}$$

therefore,

$$\mathbf{y}^{t+1} = \mathbf{y}^t - \gamma_m \beta \mathbf{b}^t - \gamma_m \nabla f(\mathbf{y}^t; \xi^t) \tag{24}$$

$$= \mathbf{y}^t - \frac{\gamma_m \beta}{\gamma_p}(\mathbf{x}^t - \mathbf{z}^{t+1}) - \frac{\gamma_m}{\gamma_p}(\mathbf{z}^t - \mathbf{z}^{t+1}) \tag{25}$$

$$= \mathbf{y}^t + (1-\beta)(\mathbf{z}^{t+1} - \mathbf{z}^t) + \beta(1-\beta)(\mathbf{z}^{t+1} - \mathbf{x}^t). \tag{26}$$

□

B Convergence Proof

Notation. We denote $\|\cdot\| = \|\cdot\|_2$ as the vector 2-norm.

B.1 Strongly Convex

Under an appropriate choice of Lyapunov function $\mathcal{E}_t$ where

$$\mathcal{E}_t := f(\mathbf{x}^t) - f(\mathbf{x}^\star) + \frac{\mu}{2}\|\mathbf{v}^t - \mathbf{x}^\star\|^2 \tag{27}$$

where $\mathbf{v}^t := \mathbf{x}^t + c\mathbf{m}^t$ for some $c > 0$, we provide a proof of linear convergence towards the optimal solution as follows.

First, we expand the one step evolution of every variable as follows:

$$\mathbf{x}^{t+1} = \mathbf{y}^t - \alpha \nabla f(\mathbf{y}^t) = \mathbf{x}^t + \beta \mathbf{m}^t - \alpha \nabla f(\mathbf{y}^t), \tag{28}$$

$$\mathbf{m}^{t+1} = \gamma\mathbf{m}^t + (1-\gamma)(\mathbf{x}^{t+1} - \mathbf{x}^t) \tag{29}$$

$$\overset{(28)}{=} \gamma\mathbf{m}^t + (1-\gamma)(\beta\mathbf{m}^t - \alpha\nabla f(\mathbf{y}^t)) \tag{30}$$

$$= (\gamma + (1-\gamma)\beta)\mathbf{m}^t - (1-\gamma)\alpha\nabla f(\mathbf{y}^t). \tag{31}$$

Then, the one step evolution of $\mathbf{v}^t$ can be derived as

$$\mathbf{v}^{t+1} = \mathbf{x}^{t+1} + c\mathbf{m}^{t+1} \tag{32}$$

$$= \mathbf{x}^t + \beta\mathbf{m}^t - \alpha\nabla f(\mathbf{y}^t) + c\left[(\gamma + (1-\gamma)\beta)\mathbf{m}^t - (1-\gamma)\alpha\nabla f(\mathbf{y}^t)\right] \tag{33}$$

$$= \mathbf{x}^t + [\beta + c(\gamma + (1-\gamma)\beta)]\mathbf{m}^t - \alpha[1 + c(1-\gamma)]\nabla f(\mathbf{y}^t). \tag{34}$$

Now we wish to find some $c > 0$ such that the above equation can be equivalently reduced to

$$\mathbf{v}^{t+1} = (1-r)\mathbf{v}^t + r\left(\mathbf{y}^t - \frac{1}{\mu}\nabla f(\mathbf{y}^t)\right) \tag{35}$$

for some $r > 0$. In this regard, we expand the R.H.S. of (35) and obtain

$$(1-r)(\mathbf{x}^t + c\mathbf{m}^t) + r\left(\mathbf{x}^t + \beta\mathbf{m}^t - \frac{1}{\mu}\nabla f(\mathbf{y}^t)\right) \tag{36}$$

$$= \mathbf{x}^t + [(1-r)c + r\beta]\mathbf{m}^t - \frac{r}{\mu}\nabla f(\mathbf{y}^t). \tag{37}$$

Then we are left with matching the coefficients between (34) and (37), i.e., we find a set of c, r, β such that

$$\beta + c(\gamma + (1-\gamma)\beta) = (1-r)c + r\beta \tag{38}$$

and

$$\alpha[1 + c(1-\gamma)] = \frac{r}{\mu} \Leftrightarrow r = \alpha\mu[1 + c(1-\gamma)]. \tag{39}$$

By choosing

$$c = \frac{1-\gamma-r}{(1-\gamma)r}, \tag{40}$$

we deduce that

$$r = \alpha\mu\left(1 + \frac{1-\gamma-r}{r}\right) = \alpha\mu\frac{1-\gamma}{r} \tag{41}$$

$$\Leftrightarrow r = \sqrt{\alpha\mu(1-\gamma)} \tag{42}$$

will match the gradient term coefficient as written in (39). Meanwhile, back to the coefficient of $\mathbf{m}^t$ term in (38), we get

$$\beta + \frac{1-\gamma-r}{(1-\gamma)r}(\gamma + (1-\gamma)\beta)) = (1-r)\frac{1-\gamma-r}{(1-\gamma)r} + r\beta \tag{43}$$

$$\Leftrightarrow \beta(1 + \frac{1-\gamma-r}{r} - r) = \frac{1-r}{r}\cdot\frac{1-\gamma-r}{1-\gamma} - \gamma\cdot\frac{1-\gamma-r}{(1-\gamma)r} \tag{44}$$

$$\Leftrightarrow \beta(\frac{1-\gamma}{r} - r) = \frac{(1-\gamma-r)^2}{r(1-\gamma)} \tag{45}$$

$$\Leftrightarrow \beta(1-\gamma-r^2) = \frac{(1-\gamma-r)^2}{(1-\gamma)} \tag{46}$$

$$\Leftrightarrow \beta = \frac{(1-\gamma-r)^2}{(1-\gamma)(1-\gamma-r^2)} \tag{47}$$

will match the $\mathbf{m}^t$ term coefficient as well. Next, using (35) and the equality $\|(1-\theta)\mathbf{u} + \theta\mathbf{w}\|^2 = (1-\theta)\|\mathbf{u}\|^2 + \theta\|\mathbf{w}\|^2 - \theta(1-\theta)\|\mathbf{u}-\mathbf{w}\|^2$ for any $\mathbf{u}, \mathbf{w} \in \mathbb{R}^d$, $\theta \in [0,1]$, we observe the evolution of the Lyapunov function by expanding

$$\|\mathbf{v}^{t+1} - \mathbf{x}^\star\|^2 \tag{48}$$

$$= \|(1-r)(\mathbf{v}^t - \mathbf{x}^\star) + r(\mathbf{y}^t - \mathbf{x}^\star - \frac{1}{\mu}\nabla f(\mathbf{y}^t))\|^2 \tag{49}$$

$$= (1-r)\|\mathbf{v}^t - \mathbf{x}^\star\|^2 + r\|\mathbf{y}^t - \mathbf{x}^\star - \frac{1}{\mu}\nabla f(\mathbf{y}^t)\|^2 - r(1-r)\|\mathbf{v}^t - \mathbf{y}^t + \frac{1}{\mu}\nabla f(\mathbf{y}^t)\|^2. \tag{50}$$

Notice that

$$\|\mathbf{y}^t - \mathbf{x}^\star - \frac{1}{\mu}\nabla f(\mathbf{y}^t)\|^2 = \|\mathbf{y}^t - \mathbf{x}^\star\|^2 + \frac{1}{\mu^2}\|\nabla f(\mathbf{y}^t)\|^2 - \frac{2}{\mu}\left\langle \nabla f(\mathbf{y}^t) \,\middle|\, \mathbf{y}^t - \mathbf{x}^\star \right\rangle \tag{51}$$

Under μ-strong convexity, we can combine the above with

$$f(\mathbf{x}^\star) \geq f(\mathbf{y}^t) + \left\langle \nabla f(\mathbf{y}^t) \,\middle|\, \mathbf{x}^\star - \mathbf{y}^t \right\rangle + \frac{\mu}{2}\|\mathbf{y}^t - \mathbf{x}^\star\|^2 \tag{52}$$

$$\Leftrightarrow -\left\langle \nabla f(\mathbf{y}^t) \,\middle|\, \mathbf{y}^t - \mathbf{x}^\star \right\rangle \leq f(\mathbf{x}^\star) - f(\mathbf{y}^t) - \frac{\mu}{2}\|\mathbf{y}^t - \mathbf{x}^\star\|^2 \tag{53}$$

and obtain

$$\|\mathbf{y}^t - \mathbf{x}^\star - \frac{1}{\mu}\nabla f(\mathbf{y}^t)\|^2 \leq \frac{1}{\mu^2}\|\nabla f(\mathbf{y}^t)\|^2 - \frac{2}{\mu}(f(\mathbf{y}^t) - f(\mathbf{x}^\star)). \tag{54}$$

Putting together (50) and (54) yields

$$\|\mathbf{v}^{t+1} - \mathbf{x}^\star\|^2 \leq (1-r)\|\mathbf{v}^t - \mathbf{x}^\star\|^2 + \frac{r}{\mu^2}\|\nabla f(\mathbf{y}^t)\|^2 - \frac{2r}{\mu}(f(\mathbf{y}^t) - f(\mathbf{x}^\star)) \tag{55}$$

$$- r(1-r)\|\mathbf{v}^t - \mathbf{y}^t + \frac{1}{\mu}\nabla f(\mathbf{y}^t)\|^2. \tag{56}$$

Meanwhile, under L-smooth assumption,

$$f(\mathbf{x}^{t+1}) - f(\mathbf{x}^\star) \leq f(\mathbf{y}^t) + \left\langle \nabla f(\mathbf{y}^t) \,\middle|\, \mathbf{x}^{t+1} - \mathbf{y}^t \right\rangle + \frac{L}{2}\|\mathbf{x}^{t+1} - \mathbf{y}^t\|^2 - f(\mathbf{x}^\star) \tag{57}$$

$$= f(\mathbf{y}^t) - \alpha\|\nabla f(\mathbf{y}^t)\|^2 + \frac{\alpha^2 L}{2}\|\nabla f(\mathbf{y}^t)\|^2 - f(\mathbf{x}^\star) \tag{58}$$

$$\leq f(\mathbf{y}^t) - f(\mathbf{x}^\star) - \frac{\alpha}{2}\|\nabla f(\mathbf{y}^t)\|^2 \tag{59}$$

when $\alpha^2 L/2 \leq \alpha/2 \Leftrightarrow \alpha \leq 1/L$. Combining (56) and (59) together gives us the recursion of the Lyapunov function as

$$\mathcal{E}_{t+1} \leq (1-r)(f(\mathbf{y}^t) - f(\mathbf{x}^\star)) + (\frac{r}{2\mu} - \frac{\alpha}{2})\|\nabla f(\mathbf{y}^t)\|^2 + \frac{\mu(1-r)}{2}\|\mathbf{v}^t - \mathbf{x}^\star\|^2 \tag{60}$$

$$- \frac{\mu r(1-r)}{2}\|\mathbf{v}^t - \mathbf{y}^t + \frac{1}{\mu}\nabla f(\mathbf{y}^t)\|^2 \tag{61}$$

$$= (1-r)\mathcal{E}_t + (1-r)(f(\mathbf{y}^t) - f(\mathbf{x}^t)) + (\frac{r}{2\mu} - \frac{\alpha}{2})\|\nabla f(\mathbf{y}^t)\|^2 \tag{62}$$

$$- \frac{\mu r(1-r)}{2}\|\mathbf{v}^t - \mathbf{y}^t + \frac{1}{\mu}\nabla f(\mathbf{y}^t)\|^2 \tag{63}$$

$$\leq (1-r)\mathcal{E}_t + (1-r)\left\langle \nabla f(\mathbf{y}^t) \,\middle|\, \mathbf{y}^t - \mathbf{x}^t \right\rangle - \frac{\mu(1-r)}{2}\|\mathbf{y}^t - \mathbf{x}^t\|^2 \tag{64}$$

$$+ (\frac{r}{2\mu} - \frac{\alpha}{2})\|\nabla f(\mathbf{y}^t)\|^2 - \frac{\mu r(1-r)}{2}\|\mathbf{v}^t - \mathbf{y}^t + \frac{1}{\mu}\nabla f(\mathbf{y}^t)\|^2, \tag{65}$$

where the last inequality is due to μ-strong convexity. Now recall by the definition of $\mathbf{v}^t = \mathbf{x}^t + c\mathbf{m}^t$, we have

$$\mathbf{y}^t = \mathbf{x}^t + \beta\mathbf{m}^t = (1 - \beta/c)\,\mathbf{x}^t + \beta/c \cdot \mathbf{v}^t \tag{66}$$

$$\Leftrightarrow \mathbf{v}^t = c/\beta \cdot \mathbf{y}^t - (c/\beta - 1)\,\mathbf{x}^t \tag{67}$$

$$\Leftrightarrow \mathbf{v}^t - \mathbf{y}^t = (c/\beta - 1)\,(\mathbf{y}^t - \mathbf{x}^t). \tag{68}$$

This allows us to expand

$$\|\mathbf{v}^t - \mathbf{y}^t + \frac{1}{\mu}\nabla f(\mathbf{y}^t)\|^2 = (c/\beta - 1)^2\,\|\mathbf{y}^t - \mathbf{x}^t\|^2 + \frac{1}{\mu^2}\|\nabla f(\mathbf{y}^t)\|^2 \tag{69}$$

$$+ \frac{2}{\mu}\,(c/\beta - 1)\left\langle \mathbf{y}^t - \mathbf{x}^t \,\middle|\, \nabla f(\mathbf{y}^t) \right\rangle. \tag{70}$$

Putting it back to the recursion of the Lyapunov function, we get

$$\mathcal{E}_{t+1} \leq (1-r)\mathcal{E}_t + [(1-r) - (c/\beta - 1)r(1-r)] \left\langle \nabla f(\mathbf{y}^t) \,\middle|\, \mathbf{y}^t - \mathbf{x}^t \right\rangle \tag{71}$$

$$- \left(\frac{\mu(1-r)}{2} + (c/\beta - 1)^2 \frac{\mu r(1-r)}{2} \right) \|\mathbf{y}^t - \mathbf{x}^t\|^2 \tag{72}$$

$$- \left(-\frac{r}{2\mu} + \frac{\alpha}{2} + \frac{r(1-r)}{2\mu} \right) \|\nabla f(\mathbf{y}^t)\|^2. \tag{73}$$

By $\langle \nabla f(\mathbf{y}^t) \mid \mathbf{y}^t - \mathbf{x}^t \rangle \leq \|\nabla f(\mathbf{y}^t)\| \|\mathbf{y}^t - \mathbf{x}^t\|$, we rewrite the residual by defining $C := (1-r)(1 - r(c/\beta - 1))$, $A := \frac{\mu(1-r)}{2}(1 + r(c/\beta - 1)^2)$, $B := \left(-\frac{r}{2\mu} + \frac{\alpha}{2} + \frac{r(1-r)}{2\mu} \right)$ so that

$$\mathcal{E}_{t+1} \leq (1-r)\mathcal{E}_t + |C| \|\nabla f(\mathbf{y}^t)\| \cdot \|\mathbf{y}^t - \mathbf{x}^t\| - A\|\mathbf{y}^t - \mathbf{x}^t\|^2 - B\|\nabla f(\mathbf{y}^t)\|^2. \tag{74}$$

To achieve $\mathcal{E}_{t+1} \leq (1-r)\mathcal{E}_t$, our last step is to show

$$|C| \|\nabla f(\mathbf{y}^t)\| \cdot \|\mathbf{y}^t - \mathbf{x}^t\| - A\|\mathbf{y}^t - \mathbf{x}^t\|^2 - B\|\nabla f(\mathbf{y}^t)\|^2 \leq 0 \tag{75}$$

$$\Leftrightarrow A \left(\|\mathbf{y}^t - \mathbf{x}^t\|^2 - \frac{|C|}{A} \cdot \|\nabla f(\mathbf{y}^t)\| \cdot \|\mathbf{y}^t - \mathbf{x}^t\| + (\frac{|C|}{2A} \|\nabla f(\mathbf{y}^t)\|)^2 \right) \tag{76}$$

$$+ (-\frac{C^2}{4A} + B)\|\nabla f(\mathbf{y}^t)\|^2 \geq 0 \tag{77}$$

$$\Leftrightarrow A \left(\|\mathbf{y}^t - \mathbf{x}^t\| - \frac{|C|}{2A} \|\nabla f(\mathbf{y}^t)\| \right)^2 + (B - \frac{C^2}{4A})\|\nabla f(\mathbf{y}^t)\|^2 \geq 0. \tag{78}$$

Since it is obvious that $A \geq 0$, we are left with verifying $B - C^2/(4A) \geq 0 \Leftrightarrow 4AB - C^2 \geq 0$. We begin by simplifying each constant as

$$B = \frac{\alpha}{2} - \frac{r^2}{2\mu} \overset{(42)}{=} \frac{\alpha}{2} - \frac{\alpha\mu(1-\gamma)}{2\mu} = \frac{\alpha\gamma}{2} \overset{(42)}{=} \frac{r^2\gamma}{2(1-\gamma)\mu}, \tag{79}$$

$$c/\beta \overset{(40),(47)}{=} \frac{1-\gamma-r}{(1-\gamma)r} \cdot \frac{(1-\gamma)(1-\gamma-r^2)}{(1-\gamma-r)^2} = \frac{1-\gamma-r^2}{r(1-\gamma-r)} \tag{80}$$

$$\Rightarrow c/\beta - 1 = \frac{(1-r)(1-\gamma)}{r(1-\gamma-r)} \tag{81}$$

$$C \overset{(81)}{=} (1-r)(1 - \frac{(1-r)(1-\gamma)}{1-\gamma-r}) \tag{82}$$

$$= \frac{1-r}{1-\gamma-r} \cdot (1-\gamma-r-1+r+\gamma-r\gamma) \tag{83}$$

$$= -\frac{\gamma r(1-r)}{1-\gamma-r}, \tag{84}$$

$$A \overset{(81)}{=} \frac{\mu(1-r)}{2}(1 + r \cdot \frac{(1-r)^2(1-\gamma)^2}{r^2(1-\gamma-r)^2}) \tag{85}$$

$$= \frac{\mu(1-r)}{2r(1-\gamma-r)^2}(r(1-\gamma-r)^2 + (1-r)^2(1-\gamma)^2). \tag{86}$$

Therefore,

$$4AB - C^2 = \frac{r\gamma(1-r)}{(1-\gamma)(1-\gamma-r)^2}(r(1-\gamma-r)^2 + (1-r)^2(1-\gamma)^2) - \frac{\gamma^2 r^2(1-r)^2}{(1-\gamma-r)^2} \tag{87}$$

$$= \frac{r\gamma(1-r)}{(1-\gamma-r)^2} \left[\frac{r(1-\gamma-r)^2}{1-\gamma} + (1-r)^2(1-\gamma) - \gamma r(1-r) \right]. \tag{88}$$

Using

$$r(1-\gamma-r)^2+(1-r)^2(1-\gamma)^2-(1-\gamma)\gamma r(1-r) \tag{89}$$
$$=r(1-\gamma-r)^2+(1-r)(1-\gamma)[(1-r)(1-\gamma)-\gamma r] \tag{90}$$
$$=r(1-\gamma-r)^2+(1-r)(1-\gamma)(1-r-\gamma) \tag{91}$$
$$=(1-\gamma-r)(1-\gamma-r^2), \tag{92}$$

to c we obtain

$$4AB-C^2=\frac{r\gamma(1-r)(1-\gamma-r^2)}{(1-\gamma-r)(1-\gamma)}. \tag{93}$$

Since $0\le r\le 1, 0\le\gamma\le 1$, by plugging in $r=\sqrt{\alpha\mu(1-\gamma)}$ we have $1-\gamma-r^2\ge 0\Leftrightarrow\gamma\le 1$. Meanwhile, $1-\gamma-r\ge 0\Leftrightarrow\gamma\le 1-\alpha\mu$. Therefore, we have $4AB-C^2\ge 0$ and this completes the proof. □

B.2 General Convex

For general convex objective function, we aim to adopt another choice of Lyapunov function Φ_t where

$$\Phi_t:=A_t(f(\mathbf{x}^t)-f(\mathbf{x}^\star))+\frac{1}{2}\|\mathbf{x}^t+c_t\mathbf{m}^t-\mathbf{x}^\star\|_2^2 \tag{94}$$

with some $A_t, c_t>0$ determined later. Then, we can guarantee a sublinear convergence. Below we present the proof.

Define the auxiliary sequence $\mathbf{v}^t:=\mathbf{x}^t+c_t\mathbf{m}^t$. We first expand the one step evolution of every variable as follows:

$$\mathbf{x}^{t+1}=\mathbf{x}^t+\beta_t\mathbf{m}^t-\frac{1}{L}\nabla f(\mathbf{y}^t), \tag{95}$$
$$\mathbf{m}^{t+1}\overset{(95)}{=}\gamma\mathbf{m}^t+(1-\gamma)(\mathbf{x}^{t+1}-\mathbf{x}^t) \tag{96}$$
$$=\big(\gamma+(1-\gamma)\beta_t\big)\mathbf{m}^t-\frac{1-\gamma}{L}\nabla f(\mathbf{y}^t). \tag{97}$$

Then, the one step evolution of $\mathbf{v}^t$ can be derived as

$$\mathbf{v}^{t+1}=\mathbf{x}^{t+1}+c_{t+1}\mathbf{m}^{t+1} \tag{98}$$
$$=\mathbf{x}^t+\beta_t\mathbf{m}^t-\frac{1}{L}\nabla f(\mathbf{y}^t)+c_{t+1}\left[\big(\gamma+(1-\gamma)\beta_t\big)\mathbf{m}^t-\frac{1-\gamma}{L}\nabla f(\mathbf{y}^t)\right] \tag{99}$$
$$=\mathbf{x}^t+\big[\beta_t+c_{t+1}\big(\gamma+(1-\gamma)\beta_t\big)\big]\,\mathbf{m}^t-\frac{1+(1-\gamma)c_{t+1}}{L}\nabla f(\mathbf{y}^t). \tag{100}$$

By the definition of β_t,

$$\beta_t\big(1+(1-\gamma)c_{t+1}\big)+\gamma c_{t+1}=c_t. \tag{101}$$

Thus

$$\mathbf{v}^{t+1}=\mathbf{x}^t+c_t\mathbf{m}^t-\frac{1+(1-\gamma)c_{t+1}}{L}\nabla f(\mathbf{y}^t) \tag{102}$$
$$=\mathbf{v}^t-\eta_t\nabla f(\mathbf{y}^t), \tag{103}$$

where $\eta_t:=(1+(1-\gamma)c_{t+1})/L$.

Since $\mathbf{v}^t-\mathbf{x}^t=c_t\mathbf{m}^t$, we have

$$\mathbf{y}^t=\mathbf{x}^t+\beta_t\mathbf{m}^t=\mathbf{x}^t+\frac{\beta_t}{c_t}(\mathbf{v}^t-\mathbf{x}^t)=\left(1-\frac{\beta_t}{c_t}\right)\mathbf{x}^t+\frac{\beta_t}{c_t}\mathbf{v}^t. \tag{104}$$

We now find a sequence A_t so that the following holds:

$$A_{t+1}\left(1-\frac{\beta_t}{c_t}\right)\le A_t,\qquad A_{t+1}\frac{\beta_t}{c_t}=\eta_t. \tag{105}$$

In fact, define a sequence $A_t = a_t/L$ as follows:

$$a_{t+1} := \frac{c_t q_t^2}{c_t - \gamma c_{t+1}}, \qquad t \geq 0, \tag{106}$$

where

$$q_t := 1 + (1-\gamma)c_{t+1}. \tag{107}$$

Specifically, for $t = 0$, choose $a_0 \geq a_1 - q_0$ (for example, choose $a_0 = a_1$). For the given choice of c_t, one can easily verify

$$c_t - \gamma c_{t+1} = (1-\gamma)\left(1 + \frac{t}{4}\right), \qquad q_t = 1 + (1-\gamma)c_{t+1} = \frac{5}{4} + (1-\gamma)\left(1 + \frac{t}{4}\right). \tag{108}$$

Therefore, by (106), (107) as well as the definition of β_t and η_t,

$$A_{t+1}\frac{\beta_t}{c_t} = \frac{a_{t+1}}{L} \cdot \frac{\beta_t}{c_t} = \frac{1}{L}\frac{c_t q_t^2}{c_t - \gamma c_{t+1}} \cdot \frac{c_t - \gamma c_{t+1}}{c_t q_t} = \frac{q_t}{L} = \eta_t. \tag{109}$$

We now show $A_{t+1}\left(1 - \frac{\beta_t}{c_t}\right) \leq A_t$, or equivalently, $a_{t+1} - q_t \leq a_t$. When $t = 0$, the inequality holds due to our choice of a_0. For $t \geq 1$, put $s_t := 1 + \frac{t}{4}$. Then a direct calculation gives

$$a_t - (a_{t+1} - q_t) = \frac{8(1-\gamma)^3 s_t(16s_t^2 - 1) + 36(1-\gamma)^2 s_t(4s_t - 1) + 25\gamma}{64(1-\gamma)^2 s_t(4s_t - 1)}. \tag{110}$$

Since $t \geq 1$, we have $s_t \geq 5/4$. Also $0 < 1 - \gamma \leq 1$. Therefore every term in the numerator is nonnegative and the denominator is positive. Hence $a_t - (a_{t+1} - q_t) \geq 0$. Thus (105) holds.

Given the choice of A_t, we now prove the Lyapunov function Φ_t decreases.

Since f is L-smooth and $\mathbf{x}^{t+1} = \mathbf{y}^t - \frac{1}{L}\nabla f(\mathbf{y}^t)$, we have

$$f(\mathbf{x}^{t+1}) \leq f(\mathbf{y}^t) - \frac{1}{2L}\|\nabla f(\mathbf{y}^t)\|^2. \tag{111}$$

Furthermore, since

$$\mathbf{y}^t = \left(1 - \frac{\beta_t}{c_t}\right)\mathbf{x}^t + \frac{\beta_t}{c_t}\mathbf{v}^t, \tag{112}$$

we have by convexity that

$$\begin{aligned}
f(\mathbf{y}^t) - f(\mathbf{x}^\star) &= \left(1 - \frac{\beta_t}{c_t}\right)\big(f(\mathbf{y}^t) - f(\mathbf{x}^\star)\big) + \frac{\beta_t}{c_t}\big(f(\mathbf{y}^t) - f(\mathbf{x}^\star)\big) \\
&\leq \left(1 - \frac{\beta_t}{c_t}\right)\left[f(\mathbf{x}^t) - f(\mathbf{x}^\star) - \left\langle \nabla f(\mathbf{y}^t) \middle| \mathbf{x}^t - \mathbf{y}^t \right\rangle\right] + \frac{\beta_t}{c_t}\left\langle \nabla f(\mathbf{y}^t) \middle| \mathbf{y}^t - \mathbf{x}^\star \right\rangle \\
&= \left(1 - \frac{\beta_t}{c_t}\right)\big(f(\mathbf{x}^t) - f(\mathbf{x}^\star)\big) + \left\langle \nabla f(\mathbf{y}^t) \middle| \frac{\beta_t}{c_t}(\mathbf{y}^t - \mathbf{x}^\star) - \left(1 - \frac{\beta_t}{c_t}\right)(\mathbf{x}^t - \mathbf{y}^t) \right\rangle \\
&= \left(1 - \frac{\beta_t}{c_t}\right)\big(f(\mathbf{x}^t) - f(\mathbf{x}^\star)\big) + \frac{\beta_t}{c_t}\left\langle \nabla f(\mathbf{y}^t) \middle| \mathbf{v}^t - \mathbf{x}^\star \right\rangle.
\end{aligned} \tag{113}$$

Multiplying (113) by A_{t+1} and then using (105) and (111), we obtain

$$A_{t+1}\big(f(\mathbf{x}^{t+1}) - f(\mathbf{x}^\star)\big) \leq A_t\big(f(\mathbf{x}^t) - f(\mathbf{x}^\star)\big) + \eta_t\langle \nabla f(\mathbf{y}^t), \mathbf{v}^t - \mathbf{x}^\star\rangle - \frac{A_{t+1}}{2L}\|\nabla f(\mathbf{y}^t)\|^2. \tag{114}$$

On the other hand, by (103) we have

$$\frac{1}{2}\|\mathbf{v}^{t+1} - \mathbf{x}^\star\|^2 = \frac{1}{2}\|\mathbf{v}^t - \mathbf{x}^\star\|^2 - \eta_t\left\langle \nabla f(\mathbf{y}^t) \middle| \mathbf{v}^t - \mathbf{x}^\star \right\rangle + \frac{\eta_t^2}{2}\|\nabla f(\mathbf{y}^t)\|^2. \tag{115}$$

Adding (114) and (115) yields

$$\Phi_{t+1} \leq \Phi_t - \frac{1}{2}\left(\frac{A_{t+1}}{L} - \eta_t^2\right)\|\nabla f(\mathbf{y}^t)\|^2. \tag{116}$$

It remains to check that

$$\frac{A_{t+1}}{L} \geq \eta_t^2. \tag{117}$$

Since $A_{t+1} = a_{t+1}/L$ and $\eta_t = q_t/L$, this is equivalent to $a_{t+1} \geq q_t^2$, which apparently holds by our choice of a_{t+1} in (106). Combining with (116), we show that

$$\Phi_{t+1} \leq \Phi_t. \tag{118}$$

Consequently, $A_t\big(f(\mathbf{x}^t) - f(\mathbf{x}^\star)\big) \leq \Phi_t \leq \Phi_0$, and therefore

$$f(\mathbf{x}^t) - f(\mathbf{x}^\star) \leq \frac{\Phi_0}{A_t} = \frac{L\Phi_0}{a_t}. \tag{119}$$

Next, we lower bound a_t. For $t \geq 1$, by (106) we have

$$a_t = \frac{c_{t-1}q_{t-1}^2}{c_{t-1} - \gamma c_t}. \tag{120}$$

Recalling $s_{t-1} = 1 + \frac{t-1}{4}$, a direct calculation can verify

$$c_{t-1} \geq s_{t-1}, \qquad q_{t-1} \geq (1-\gamma)s_{t-1}, \qquad c_{t-1} - \gamma c_t = (1-\gamma)s_{t-1}. \tag{121}$$

Thus

$$a_t \geq \frac{s_{t-1}\big((1-\gamma)s_{t-1}\big)^2}{(1-\gamma)s_{t-1}} = (1-\gamma)s_{t-1}^2 \geq \frac{(1-\gamma)t^2}{16}. \tag{122}$$

Therefore by (119) and (122), we obtain

$$f(\mathbf{x}^t) - f(\mathbf{x}^\star) \leq \frac{16L\Phi_0}{(1-\gamma)t^2}. \tag{123}$$

(123) provides the $O(1/t^2)$ rate of the algorithm. We further upper bound Φ_0 to provide the γ-dependency. Note that $\mathbf{m}^0 = 0$. Since f is convex and L-smooth and $\mathbf{x}^\star$ is a minimizer, we have

$$f(\mathbf{x}^0) - f(\mathbf{x}^\star) \leq \frac{L}{2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2. \tag{124}$$

Combining (94) and (124) yields

$$\Phi_0 = \frac{a_0}{L}\big(f(\mathbf{x}^0) - f(\mathbf{x}^\star)\big) + \frac{1}{2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 \leq \frac{a_0+1}{2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2. \tag{125}$$

Now we calculate a_0. By the definition of c_t, a_t and q_t (recall (106) and (107)), we have

$$c_0 = 1 + \frac{\gamma}{4(1-\gamma)}, \qquad c_1 = \frac{5}{4} + \frac{\gamma}{4(1-\gamma)}, \qquad q_0 = 1 + (1-\gamma)c_1, \qquad a_1 = \frac{c_0 q_0^2}{c_0 - \gamma c_1}. \tag{126}$$

We need $a_0 \geq a_1 - q_0$. By simply choosing $a_0 = a_1$, a direct calculation yields

$$a_0 = a_1 = \frac{(4-3\gamma)(9-4\gamma)^2}{64(1-\gamma)^2} \leq \frac{81}{16(1-\gamma)^2}. \tag{127}$$

Therefore, by combining (123), (125) and (127), we get

$$\begin{aligned} f(\mathbf{x}^t) - f(\mathbf{x}^\star) \leq & \frac{16L}{(1-\gamma)t^2} \cdot \frac{a_0+1}{2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2 \\ \leq & \frac{16L}{(1-\gamma)t^2} \cdot \frac{\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2}{2} \cdot \frac{97}{16(1-\gamma)^2} \\ = & \frac{97L}{2(1-\gamma)^3 t^2}\|\mathbf{x}^0 - \mathbf{x}^\star\|_2^2. \end{aligned} \tag{128}$$

This completes the proof. □

C Additional Experiments Details

C.1 Validation Loss at Lookahead Positions

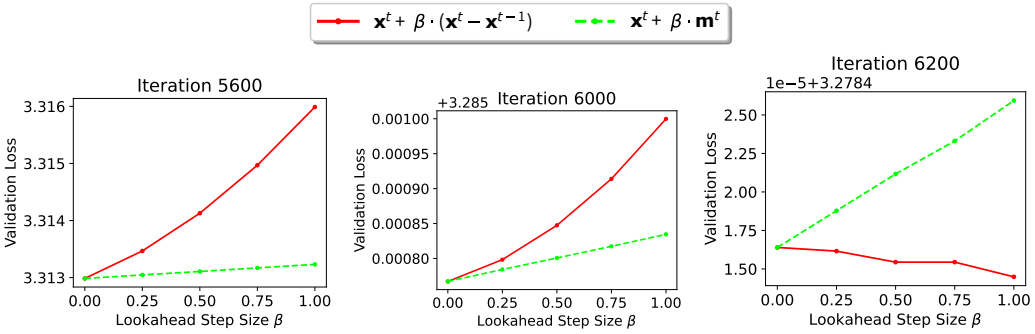


Figure 8: Under the NanoGPT setup, we measure the validation loss in lookahead positions $\mathbf{y}^t = \mathbf{x}^t + \beta \mathbf{d}^t$ at iterations $t = 5600, 6000, 6200$ using (Solid Red) $\mathbf{y}^t = \mathbf{x}^t + \beta \cdot (\mathbf{x}^t - \mathbf{x}^{t-1})$ and (Dashed Green) $\mathbf{y}^t = \mathbf{x}^t + \beta \cdot \mathbf{m}^t$ where $\mathbf{m}^t = \gamma \mathbf{m}^{t-1} + (1-\gamma)(\mathbf{x}^t - \mathbf{x}^{t-1})$ with $\gamma = 0.995$.

In Figure 8, we measure the validation loss at different lookahead positions during the learning rate decay phase when the learning rate is very small. Since the loss landscape changes rapidly near the local minima, our EMA lookahead direction $\mathbf{m}^t$ with large γ retained the lookahead momentum from past iterations which is not applicable to the sharp local minima. This motivates the decaying schedule of β_t and the early rest stage described in Section 3.3.

C.2 Learning Rate Scaling with Batch Size

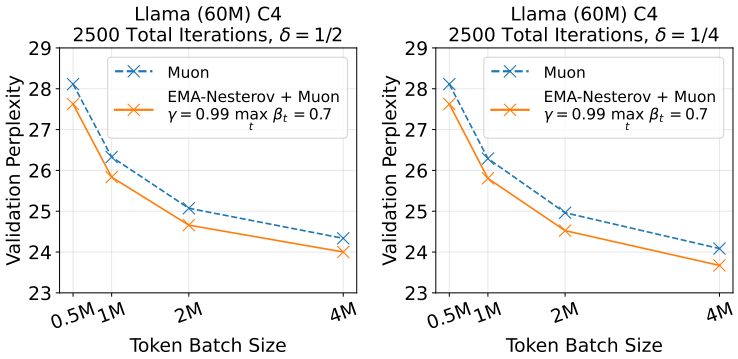


Figure 9: Data scaling to total training budgets of $\geq \{20\times, 40\times, 80\times, 160\times\}$ tokens-per-parameter with increased token batch size $\{0.5\text{M}, 1\text{M}, 2\text{M}, 4\text{M}\}$ respectively and fixed 2500 total iterations. The learning rates are adapted under the power scaling rule: for tuned learning rate α on batch size B, the adapted learning rate α' for larger batch size $B' > B$ is $\alpha' = \alpha \cdot (B'/B)^\delta$ with the learning rate scaling exponent (Left) $\delta = 1/2$ and (Right) $\delta = 1/4$.

In Figure 9, we compare the two learning rate scaling rules under different scaling exponents. Compared to the usual scaling exponent $\delta = 1/2$ used in SGD [Hoffer et al., 2017], we tried alternative scaling exponent $\delta = 1/4$ and observed a better performance for Muon especially on larger batch sizes.

C.3 Hyperparameter Sensitivity on Model Scaling

In Figure 10 we provide additional results on the hyperparameter sweep for `EMA-Nesterov` when training the Llama model for 5000 iterations. Compared to the 2500 iterations setup in Figure 6 where the optimal EMA rate $\gamma = 0.99$, the optimal EMA rate in Figure 10 shifts to $\gamma = 0.995$. This

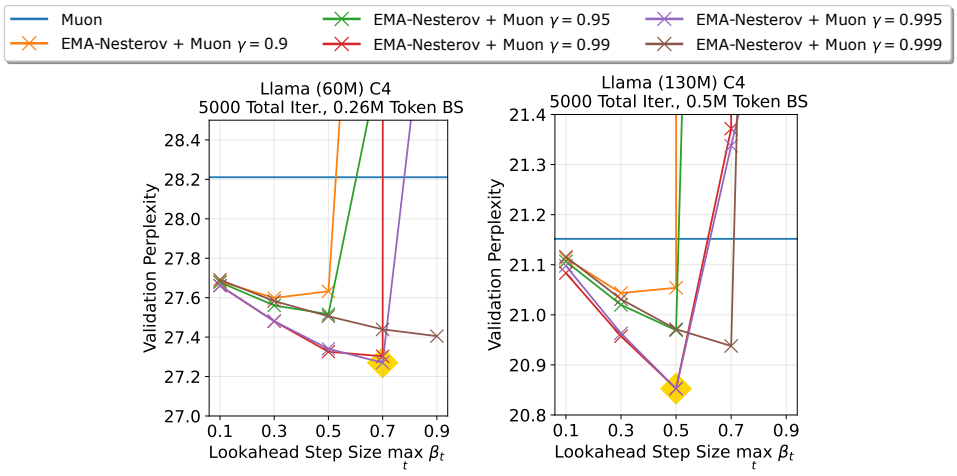


Figure 10: Sensitivity of hyperparameters for `EMA-Nesterov`-Muon across different model sizes on the Llama setup. The value of $\max_t \beta_t$ is searched in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ across the horizontal axis and the value of γ is searched in $\{0.9, 0.95, 0.99, 0.995, 0.999\}$ across different colors. The learning rates are adapted under the power scaling rule: for tuned learning rate α on batch size B, the adapted learning rate α' for larger batch size $B' > B$ is $\alpha' = \alpha \cdot (B'/B)^{\delta}$ with $\delta = 1/2$.

indicates that the choice of γ is determined by the temporal behavior of the training trajectory, such that the different training trajectories due to different total iterations and learning rate schedule lead to different optimal choice of γ.

C.4 Experiment Setting

In experiment (I) training NanoGPT (124M) model on FineWeb dataset, our setting strictly follows an early version (*track_1_short/2024-10-10_Muon*) of the NanoGPT Track 1 benchmark, including the implementation of the Muon algorithm. A total of 6200 iterations is trained with batch size 512 on sequence length of 1024 tokens, approximately 0.5M token batch size. The learning rate is kept constant from the beginning and a linear decay to 0 for the last 1800 iterations is adopted. In Table 3, we summarize the hyperparameters of the base optimizers used in this setting. For `EMA-Nesterov`, we adopted the same β_t scheduling of $T_w = 1800$ and $T_r = 6200 - 600 = 5600$ for all NanoGPT (124M) experiments. Since no learning rate warm-up is applied, we expect the training takes longer to enter the stable stage and therefore the slightly larger T_w. Meanwhile, as the learning rate decay phase is long, a late T_r is adopted to accelerate the mid-to-late stage training. The time for each experiment run varies for different base optimizers, ranging from 24 minutes for Muon and 37 minutes for SOAP in Figure 4.

We also conducted experiments using the NanoGPT Track 3 setup[6], which differs from Track 1 by increasing the number of trainable parameters to 162M through detaching the parameters of input and output layer. For the base optimizers in Track 3 of Figure 4, we adopted Muon from leaderboard entry #12 and Aurora from pull request entry #300. For the β_t scheduling in `EMA-Nesterov`, we used $T_w = 500$, $T_r = 3150 - 750 = 2400$, $\max_t \beta_t = 0.6$ for Muon and $T_w = 300$, $T_r = 2900 - 950 = 1950$, $\max_t \beta_t = 0.3$ for Aurora. We used $\gamma = 0.99$ for `EMA-Nesterov` in both algorithms.

	Muon LR	Adam LR	Muon Momentum	Adam 1st Momentum	Adam 2nd Momentum
Muon	3.6×10^{-4}	3.6×10^{-3}	0.95	0.9	0.95
NorMuon	3.6×10^{-4}	3.6×10^{-3}	0.95	0.9	0.95
	LR	1st Momentum	2nd Momentum	Precondition Frequency	-
SOAP	10^{-3}	0.9	0.95	10	-
Adam	10^{-3}	0.9	0.95	-	-

Table 3: Hyperparameter values of the base optimizers used in NanoGPT Track 1 (Figure 4 and 5).

In experiment (II) training Llama models on C4 dataset, we used a sequence length of 256 tokens. Total training iterations and batch sizes vary between different experiment runs, as described in the title of Figure 7, 6, 9 and 10. The learning rate follows a warmup-stable-decay scheduling [Hu et al., 2024], with the first 10% of iterations performing linear warm-up, followed with staying constant in the middle 80% of iterations and ends with an exponential decay to a minimum of $0.1\times$ the maximum learning rate for the last 10% of iterations. For the scheduling of β_t in `EMA-Nesterov`, when the total iteration is 2500, we used $T_w = 750$ and $T_r = 2500 - 500 = 2000$. When the total iteration is 5000, it is relatively scaled up to $T_w = 1500$ and $T_r = 5000 - 1000 = 4000$. This β_t schedule roughly follows the learning rate schedule with an additional safe margin, i.e., β_t used a 30% of early iterations to warm-up (i.e., $\beta_t = 0$ for $t \leq 0.3 \cdot T$) and uses a 20% of late iterations to rest (i.e., $\beta_t = 0$ for $t > 0.8 \cdot T$). We only used Muon[7] as the base optimizer in this setting and it used a momentum of 0.95 for intermediate layers, a 1st and 2nd order momentum of 0.9, 0.95 for first/last layers, see Algorithm 2 for a detailed description. We summarize the remaining hyperpameters for each figure in Table 4. The time for experiment varies across model sizes, ranging from 40 mintues for 60M model in Figure 6 to 24 hours for 1B model in Table 2.

[6]NanoGPT Track 3 benchmark from `https://github.com/KellerJordan/modded-nanogpt/tree/master/records/track_3_optimization`.

[7]For all Llama experiments, we adopted the open-sourced implementation of Muon in `https://github.com/KellerJordan/Muon`.

	Model Size	Total Iterations	Token Batch Size	Muon LR	Adam LR
Figure 6	60M	2500	0.5M	3×10^{-2}	3×10^{-2}
Figure 6	130M	2500	1M	4.24×10^{-2}	4.24×10^{-2}
Figure 6	350M	2500	3M	7.35×10^{-2}	7.35×10^{-2}
Figure 9 ($\delta = 1/2$)	60M	2500	0.5M	3×10^{-2}	3×10^{-2}
Figure 9 ($\delta = 1/2$)			1M	4.24×10^{-2}	4.24×10^{-2}
Figure 9 ($\delta = 1/2$)			2M	6×10^{-2}	6×10^{-2}
Figure 9 ($\delta = 1/2$)			4M	8.49×10^{-2}	8.49×10^{-2}
Figure 7, 9 ($\delta = 1/4$)	60M	2500	0.5M	3×10^{-2}	3×10^{-2}
Figure 7, 9 ($\delta = 1/4$)			1M	3.56×10^{-2}	3.56×10^{-2}
Figure 7, 9 ($\delta = 1/4$)			2M	4.24×10^{-2}	4.24×10^{-2}
Figure 7, 9 ($\delta = 1/4$)			4M	5.05×10^{-2}	5.05×10^{-2}
Figure 7 ($\delta = 1/4$)	130M	2500	1M	4.24×10^{-2}	4.24×10^{-2}
Figure 7 ($\delta = 1/4$)			2M	5.05×10^{-2}	5.05×10^{-2}
Figure 7 ($\delta = 1/4$)			4M	6×10^{-2}	6×10^{-2}
Figure 7 ($\delta = 1/4$)			8M	7.14×10^{-2}	7.14×10^{-2}
Figure 7 ($\delta = 1/4$)	350M	2500	3M	3×10^{-2}	3×10^{-2}
Figure 7 ($\delta = 1/4$)			6M	3.57×10^{-2}	3.57×10^{-2}
Figure 7 ($\delta = 1/4$)			12M	4.24×10^{-2}	4.24×10^{-2}
Figure 7 ($\delta = 1/4$)			25M	5.05×10^{-2}	5.05×10^{-2}
Figure 10	60M	5000	0.26M	2.12×10^{-2}	2.12×10^{-2}
Figure 10	130M		0.5M	3×10^{-2}	3×10^{-2}
Table 2	1B	5000	4M	5×10^{-2}	5×10^{-2}

Table 4: Hyperparameter values of the base optimizers used in Figure 6, 7, 9, 10 and Table 2. The learning rates are adapted under the power scaling rule: for tuned learning rate α on batch size B, the adapted learning rate α' for larger batch size $B' > B$ is $\alpha' = \alpha \cdot (B'/B)^{\delta}$ with a default choice of $\delta = 1/2$.

C.5 Muon Optimizer

For completeness, we provide the exact open-sourced implementation of Muon[8] in Algorithm 2. Unless otherwise specified, we do not apply weight decay in the base optimizer, i.e., we use $\lambda = 0$ as default. We denote $\odot$ as the element-wise product operator.

Algorithm 2 Muon

1: **Input:** Muon learning rates $\alpha_t > 0$, Adam learning rates $\widehat{\alpha}_t > 0$, weight decay $\lambda \geq 0$, Muon momentum $\phi \in [0, 1)$, Adam momentum $\phi_1, \phi_2 \in [0, 1)$.
2: **Initialize:** Model parameters $\mathbf{x}^0 = [\mathbf{x}_1^0, ..., \mathbf{x}_L^0]$ for L layers s.t. $\mathbf{x}_\ell^0 \in \mathbb{R}^{M_\ell \times N_\ell}$ with input dimension M_ℓ and output dimension N_ℓ. Muon gradient momentum $\mathbf{v}^0 = [\mathbf{v}_2^0, ..., \mathbf{v}_{L-1}^0] = \mathbf{0}$. Adam 1st and 2nd order gradient momentum $\mathbf{w}^0 = [\mathbf{w}_1^0, ..., \mathbf{w}_L^0] = \mathbf{0}$, $\widehat{\mathbf{w}}^0 = [\widehat{\mathbf{w}}_1^0, ..., \widehat{\mathbf{w}}_L^0] = \mathbf{0}$.
3: **for** $t = 0, ..., T-1$ **do**
4: **for** $\ell = 1, ..., L$ **do**
5: **if** $\ell \in [2, L-1]$ and $M_\ell > 1$ and $N_\ell > 1$ **then**
6: $\alpha_{t,\ell} = \alpha_t \cdot \sqrt{\max(1, N_\ell / M_\ell)}$
7: $\mathbf{x}_\ell^{t+1} = (1 - \alpha_t \lambda)\mathbf{x}_\ell^t - \alpha_{t,\ell} \cdot \text{Newton-Schulz5}(\phi^2 \mathbf{v}_\ell^t + (1-\phi^2)\nabla f(\mathbf{x}^t; \xi^t)_\ell)$
8: $\mathbf{v}_\ell^{t+1} = \phi \mathbf{v}_\ell^t + (1-\phi)\nabla f(\mathbf{x}^t; \xi^t)_\ell$
9: **else**
10: $\mathbf{w}_\ell^{t+1} = \phi_1 \mathbf{w}_\ell^t + (1-\phi_1)\nabla f(\mathbf{x}^t; \xi^t)_\ell$
11: $\widehat{\mathbf{w}}_\ell^{t+1} = \phi_2 \widehat{\mathbf{w}}_\ell^t + (1-\phi_2)\nabla f(\mathbf{x}^t; \xi^t)_\ell \odot f(\mathbf{x}^t; \xi^t)_\ell$
12: $\mathbf{x}_{\ell,ij}^{t+1} = (1 - \widehat{\alpha}_t \lambda)\mathbf{x}_{\ell,ij}^t - \widehat{\alpha}_t \cdot \frac{\mathbf{w}_{\ell,ij}^{t+1}/(1-\phi_1^t)}{\sqrt{\widehat{\mathbf{w}}_{\ell,ij}^{t+1}/(1-\phi_2^t)} + 10^{-10}}$ for $i = 1, ..., M_\ell, j = 1, ..., N_\ell$.
13: **end if**
14: **end for**
15: **end for**
16: **Output:** last iterate solution $\mathbf{x}^T$.

Notice that Algorithm 2 relies on the following 5-steps Newton-Schulz iteration algorithm to approximately normalize the spectral values of the update direction.

Algorithm 3 Newton-Schulz5

Input: $\mathbf{X} \in \mathbb{R}^{M \times N}$.
if $N > M$ **then**
 $\widehat{\mathbf{X}} = \mathbf{X}^\top$
else
 $\widehat{\mathbf{X}} = \mathbf{X}$
end if
$\mathbf{X}^0 = \widehat{\mathbf{X}} / (\|\widehat{\mathbf{X}}\|_F + 10^{-7})$
for $t = 0$ **to** 4 **do**
 $\mathbf{A}^t = \mathbf{X}^t (\mathbf{X}^t)^\top$
 $\mathbf{B}^t = (2.0315 \cdot \mathbf{A}^t - 4.7750 \cdot \mathbf{I})\mathbf{A}^t$
 $\mathbf{X}^{t+1} = (3.4445 \cdot \mathbf{I} + \mathbf{B}^t)\mathbf{X}^t$
end for
if $N > M$ **then**
 $\widetilde{\mathbf{X}} = (\mathbf{X}^5)^\top$
else
 $\widetilde{\mathbf{X}} = \mathbf{X}^5$
end if
Output: $\widetilde{\mathbf{X}}$.

We remark that in the source code of Muon, the convex combination between $\mathbf{v}_\ell^t$ and $\nabla f(\mathbf{x}^t; \xi^t)_\ell$ in line 7 of Algorithm 2 is referred to as using "Nesterov". In contrary to the description of its source code, we demonstrate explicitly that line 7 of Algorithm 2 is still based on EMA of stochastic

[8] See the open-sourced implementation here `https://github.com/KellerJordan/Muon`.

gradient with different EMA rate. In particular, $\mathbf{v}_\ell^t$ is an EMA of rate ϕ for $\{\nabla f(\mathbf{x}^r;\xi^r)\}_{r=0}^{t-1}$ which is combined on-the-fly with the current stochastic gradient $\nabla f(\mathbf{x}^t;\xi^t)_\ell$ of a larger weighting $1-\phi^2$ because $1-\phi^2 > 1-\phi$ for $\phi \in (0,1)$. The larger weighting $1-\phi^2$ controls a bias-variance tradeoff as the current stochastic gradient is unbiased to the current solution $\mathbf{x}^t$.