MARS-M: WHEN VARIANCE REDUCTION MEETS MATRICES

Anonymous authors
Paper under double-blind review

ABSTRACT

Matrix-based preconditioned optimizers, such as Muon, have recently been shown to be more efficient than scalar-based optimizers for training large-scale neural networks, including large language models (LLMs). Recent benchmark studies of LLM pretraining optimizers have demonstrated that variance-reduction techniques such as MARS can substantially speed up training compared with standard optimizers that do not employ variance reduction. In this paper, we introduce MARS-M, a new optimizer that integrates MARS-style variance reduction with Muon. Under standard regularity conditions, we prove that MARS-M converges to a first-order stationary point at a rate of $\widetilde{\mathcal{O}}(T^{-1/3})$, improving upon the $\widetilde{\mathcal{O}}(T^{-1/4})$ rate attained by Muon. Empirical results on language modeling and computer vision tasks demonstrate that MARS-M consistently yields lower losses and improved performance across various downstream benchmarks.

1 INTRODUCTION

The development of preconditioned gradient methods, such as AdamW (Loshchilov & Hutter, 2019), AdaFactor (Shazeer & Stern, 2018), and Lion (Chen et al., 2023), has played an important role in the advancement of large-scale deep learning. Many prominent large language models (LLMs), including ChatGPT (OpenAI, 2023), LLaMA-3 (Dubey et al., 2024), and DeepSeek-R1 (Guo et al., 2025), are trained with adaptive gradient methods such as Adam (Kingma & Ba, 2015) and AdamW. Recently, matrix-based preconditioned optimization methods, such as Shampoo (Gupta et al., 2018), SOAP (Vyas et al., 2024), and Muon (Jordan et al., 2024; Liu et al., 2025), have been introduced to accelerate the training of large models such as Kimi K2 (Team et al., 2025) and GLM-4.5 (Zeng et al., 2025). Unlike vector-based methods, matrix-based approaches operate directly on parameter matrices without flattening them, thereby preserving their inherent 2D structure and the underlying optimization geometry.

On the other hand, stochastic optimization methods are often hindered by high variance in stochastic gradients during training, which can slow convergence and degrade stability. To address this issue, numerous variance-reduction techniques have been proposed, including SAG (Roux et al., 2012), SVRG (Johnson & Zhang, 2013), SARAH (Nguyen et al., 2017a;b), SPIDER (Fang et al., 2018), SNVRG (Zhou et al., 2020), and STORM (Cutkosky & Orabona, 2019), to name a few. However, these methods have seen limited success in training large-scale deep neural networks, largely due to incompatibilities with modern deep-learning practice and neural network architectures (Defazio & Bottou, 2019). To make variance reduction work for training large-scale deep neural networks and LLMs, MARS (Yuan et al., 2025) was recently proposed. It introduces a scaling parameter into the STORM optimizer (Cutkosky & Orabona, 2019), effectively resolving the incompatibility between variance-reduction techniques and preconditioned optimization methods, which have demonstrated superior performance in both language modeling and computer vision tasks. A similar scaling idea has also been proposed for SVRG in Yin et al. (2024).

Recent benchmarks of optimizers for LLM pretraining (Wen et al., 2025; Semenov et al., 2025) have empirically shown that matrix-based optimizers (e.g., Shampoo, SOAP, and Muon) outperform scalar-based optimizers (e.g., AdamW and Lion), and that variance-reduction approaches (e.g., MARS and NadamW (Dozat, 2016)) can yield additional, discernible speedups. Then a natural question arises:

Can we achieve the best of both worlds by combining matrix-based optimizers with variance reduction such as MARS?

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

It is worth noting that the original MARS work (Yuan et al., 2025) proposed MARS-Shampoo, which attempts to combine a matrix-based optimizer with variance reduction. However, MARS-Shampoo performs worse than vector-based variants such as MARS-AdamW, leaving open the question of whether variance reduction can be effectively integrated with matrix-based optimizers. In this paper, we give an affirmative answer by introducing MARS-M, a matrix-based MARS optimizer that integrates variance reduction with the Moonlight version (Liu et al., 2025) of Muon. Additionally, we propose an approximate version of MARS-M as a practical variant, which also serves as a bridge between variance-reduction techniques and traditional preconditioned optimizers.

In summary, our contributions are highlighted as follows:

- We propose MARS-M, an instantiation of MARS tailored to the Moonlight optimizer. In addition, we develop an approximate version of MARS-M to further accelerate training. We show the connections and differences between MARS-M and MARS-Shampoo. We also show that the approximate version of MARS-M can be seen as a variant of Moonlight with adjusted momentum parameters.
- We provide a convergence analysis of MARS-M and show that it attains a convergence rate of $\widetilde{\mathcal{O}}(T^{-1/3})$, which improves upon the previously established $\mathcal{O}(T^{-1/4})$ rate achieved by Muon (Li & Hong, 2025; Shen et al., 2025; Pethick et al., 2025).
- Empirically, we evaluate the performance of MARS-M as well as the Muon (Moonlight) optimizer for training GPT-2 (Radford et al., 2019) series models on OpenWebText and FineWeb-Edu 100B datasets. The experimental results show that MARS-M delivers consistent improvements in training and validation losses. Moreover, on downstream tasks, MARS-M achieves better performance on benchmarks such as Hellaswag (Zellers et al., 2019) and SciQ (Welbl et al., 2017) than baseline optimizers. Additionally, MARS-M achieves higher test accuracy than AdamW and Muon on computer vision tasks.

Notation We use bold capital letters $\mathbf{X}, \mathbf{Y}, \cdots$ to denote matrices. For a matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$, we use $\|\mathbf{W}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n \mathbf{W}_{ij}^2}$ to denote its Frobenius norm. In this paper, we use $\mathbf{X}_t \in \mathbb{R}^{m\times n}$ to represent the parameters of the language model at training step t, where the training data for each step is a sequence of independent random variables $\xi_1, \ldots, \xi_T \in \mathbf{\Xi}$. For a differentiable objective function f, we assume that the expected value of $f(\mathbf{X}, \xi_t)$ given $\mathbf{X}$ is $F(\mathbf{X})$ for all $\mathbf{X}$ and t. In this paper, we may explicitly denote the input data, since variance-reduction algorithms may utilize both the training data from the current step (ξ_t) and the previous step (ξ_{t-1}) to compute different gradients for the same parameter. MARS-M is directly inspired by the success of matrix-based optimizers Muon and Moonlight.

2 Related Work

Preconditioned gradient methods Preconditioned gradient methods, inspired by Newton's method, are a powerful class of optimization algorithms widely used in training deep neural networks. To reduce the computational and memory costs of computing and storing the full Hessian matrix, many approximate preconditioners have been proposed, including diagonal approximations (Duchi et al., 2011; Loshchilov & Hutter, 2019; Liu et al., 2024; Yao et al., 2021) and sketched approximations (Erdogdu & Montanari, 2015; Gonen & Shalev-Shwartz, 2015). Despite these advances, only a few matrix-based optimization methods have been practical at large scale. A notable breakthrough was the Shampoo optimizer (Gupta et al., 2018), which demonstrated the practical feasibility of incorporating second-order information in large-scale training. To reduce the number of hyperparameters and computational overhead of Shampoo, Vyas et al. (2024) proposed SOAP, which implements AdamW in the eigenbasis of Shampoo's preconditioners, significantly improving training efficiency in terms of both iteration count and wall-clock time. Recently, Muon (Jordan et al., 2024) was proposed by leveraging a Newton–Schulz approximation of the singular value decomposition (SVD) of the gradient momentum, while Moonlight (Liu et al., 2025) extends Muon by introducing weight decay and coupling step sizes across components optimized by AdamW and Muon, thereby improving hyperparameter search efficiency. Building on this line of research, PolarGrad (Lau et al., 2025) introduces a preconditioning framework based on the polar decomposition of the gradient matrix, generalizing the principles of Muon. Scion (Pethick et al., 2025) further unifies Muon and related methods under a linear minimization oracle (LMO) framework, and Gluon (Riabinin et al., 2025)

extends both Muon and Scion by incorporating momentum-based generalizations. Today, matrix-based optimizers are becoming increasingly popular for training industrial large language models, including Kimi K2 (Team et al., 2025) and GLM-4.5 (Zeng et al., 2025).

To understand the convergence of Muon, Li & Hong (2025), Shen et al. (2025), and Pethick et al. (2025) showed that Muon has a convergence rate of $\mathcal{O}(T^{-1/4})$. An et al. (2025) and Shen et al. (2025) also analyzed the convergence of Muon without momentum. Kovalev (2025) investigated the convergence rate of Muon under additional assumptions such as spectral norm-based Lipschitz smoothness and second-order smoothness, while Li & Hong (2025), Sfyraki & Wang (2025), and Sato et al. (2025) analyzed the convergence rate under large batch sizes. Our analysis of MARS-M pushes the theoretical frontier of Muon-based optimization algorithms.

Variance Reduction Methods. The earliest efforts to accelerate stochastic gradient descent (SGD) through variance reduction include SAG (Roux et al., 2012) and SDCA (Shalev-Shwartz & Zhang, 2013). These were soon followed by simpler yet equally effective algorithms such as SVRG (Johnson & Zhang, 2013) and SAGA (Defazio et al., 2014), which achieved the same improved convergence guarantees. Building on this line of work, SARAH (Nguyen et al., 2017a) introduced a biased recursive gradient estimator that reduces memory requirements while retaining optimal complexity bounds for convex optimization. Additionally, variance reduction has also been studied in conjunction with preconditioning in the convex setting (Frangella et al., 2024; Derezinski, 2023). In the non-convex regime, algorithms like SVRG (Allen-Zhu & Yuan, 2016; Reddi et al., 2016) and SARAH (Nguyen et al., 2017b) paved the way for methods such as SPIDER (Fang et al., 2018), which integrates normalized gradient descent (Nesterov, 2013; Hazan et al., 2015) with stochastic path-integral differential estimator, and SNVRG (Zhou et al., 2020), which leverages multiple reference points to enhance variance reduction. SpiderBoost (Wang et al., 2019) further improved SPIDER by permitting large constant step sizes without compromising near-optimal oracle complexity. Subsequently, STORM (Cutkosky & Orabona, 2019) streamlined SPIDER and SNVRG via stochastic recursive momentum, and was later extended into a parameter-free variant, STORM+ (Levy et al., 2021). Recent work has also explored combining variance reduction with adaptive gradient methods. For instance, Adam$^+$ (Liu et al., 2020) reduces variance in Adam by estimating gradients only at extrapolated points, while SuperAdam (Huang et al., 2021) and VRAdam (Li, 2024) integrate variance reduction into AdamW to accelerate convergence. AdaSPIDER (Kavis et al., 2022) extends SPIDER with adaptive step sizes. However, these variance-reduced adaptive optimization algorithms have been evaluated only on relatively simple computer vision and natural language modeling benchmarks with modest model sizes. To our knowledge, MARS (Yuan et al., 2025) is the first method to achieve stellar performance on large language models. This work pointed out that previous algorithms introduced excessive gradient correction and it incorporates scaled gradient correction into adaptive gradient methods, achieving better performance in language modeling and computer vision tasks. It is worth noting that MARS attempted to incorporate matrix-level optimizers into its framework as the MARS-Shampoo instantiation but failed. And our MARS-M is built upon MARS, utilizing matrix-based optimizer Muon/Moonlight.

3 Preliminaries

In this section, we present the problem setting and relevant preliminaries, including a review of the Muon and MARS optimizers.

In this paper, we consider minimizing an objective function $F(\cdot) : \mathbb{R}^{m\times n} \to \mathbb{R}$ as follows:

$$\min_{\mathbf{X}} F(\mathbf{X}) = \mathbb{E}_{\boldsymbol{\xi}\sim\mathcal{D}}[f(\mathbf{X}, \boldsymbol{\xi})], \tag{3.1}$$

where $f(\mathbf{X}, \boldsymbol{\xi})$ is a possibly nonconvex loss function, $\mathbf{X} \in \mathbb{R}^{m\times n}$ is a matrix-variate optimization variable, and $\boldsymbol{\xi}$ is a random vector (e.g., a training data point) drawn from an unknown data distribution $\mathcal{D}$. We assume access to a first-order oracle, which returns an unbiased estimator of the gradient $\mathbb{E}[\nabla f(\mathbf{X}, \boldsymbol{\xi})] = \nabla F(\mathbf{X})$. Throughout the paper, without loss of generality, we assume $m \geq n$.

3.1 Muon

Muon (Jordan et al., 2024) was proposed to exploit the 2D geometric information of model parameter matrices during optimization. Given the momentum $\beta \in (0, 1)$ and learning rate $\eta_t > 0$, the update

rule for Muon is as follows:

$$\mathbf{M}_t = \beta \mathbf{M}_{t-1} + \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t), \tag{3.2}$$

$$\mathbf{O}_t = \text{NewtonSchulz}\left(\mathbf{M}_t\right), \tag{3.3}$$

$$\mathbf{X}_{t+1} = \mathbf{X}_t - \eta_t \mathbf{O}_t. \tag{3.4}$$

Here Newton–Schulz iteration (Bernstein & Newhouse, 2024) is used to approximate $\mathbf{U}_t\mathbf{V}_t^\top$, where $\mathbf{U}_t\mathbf{\Sigma}_t\mathbf{V}_t^\top = \mathbf{M}_t$ is the singular value decomposition (SVD) of the momentum matrix. In practice, NewtonSchulz iteration is usually applied to $\widetilde{\mathbf{M}}_t = \beta\mathbf{M}_t + \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t)$ rather than $\mathbf{M}_t$[1].

It is worth noting that Muon is designed primarily for optimizing matrix-like parameters, while AdamW is often applied to vector-like parameters, including embeddings, the language model head, and RMSNorm. However, empirical evidence (Yuan et al., 2025; Semenov et al., 2025) has shown that the original version of Muon does not perform as well in practice. Liu et al. (2025) attributed the inferior performance to mismatched update magnitudes across the two types of parameters (i.e., matrix-like and vector-like parameters). In particular, the RMS norm of AdamW updates typically ranges from approximately 0.2 to 0.4 during LLM training. Based on this observation, they proposed a variant of Muon (referred to as "Moonlight" in this paper):

$$\mathbf{U}_t = \beta \mathbf{U}_{t-1} + \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t), \tag{3.5}$$

$$\mathbf{M}_t = \beta \mathbf{U}_t + \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t), \tag{3.6}$$

$$\mathbf{O}_t = \text{NewtonSchulz}\left(\mathbf{M}_t\right), \tag{3.7}$$

$$\mathbf{X}_{t+1} = \mathbf{X}_t - \eta_t(0.2 \cdot \mathbf{O}_t \cdot \sqrt{\max(m, n)} + \lambda \mathbf{X}_t), \tag{3.8}$$

where $\mathbf{X}_t \in \mathbb{R}^{m\times n}$ is the matrix-like parameter to be optimized, and $\lambda > 0$ is the decoupled weight decay parameter. Such a corrected version has demonstrated strong performance across several benchmarks (Semenov et al., 2025; Wen et al., 2025) and successfully applied in training industrial large language models (Team et al., 2025; Zeng et al., 2025).

3.2 MARS

To accelerate the convergence of SGD, a lot of variance reduction techniques have been proposed trying to reduce the variance in update and achieve stabler and faster training.

A typical form of variance reduction is exemplified by STORM (Cutkosky & Orabona, 2019), which introduces a gradient-correction term in the momentum update rule:

$$\mathbf{m}_t = \beta \mathbf{m}_{t-1} + (1-\beta)\Big[\nabla f(\mathbf{x}_t, \boldsymbol{\xi}_t) + \underbrace{\frac{\beta}{1-\beta}\big(\nabla f(\mathbf{x}_t, \boldsymbol{\xi}_t) - \nabla f(\mathbf{x}_{t-1}, \boldsymbol{\xi}_t)\big)}_{\text{gradient correction}}\Big].$$

where $\beta > 0$ is the momentum parameter. Here the noise brought by the randomness of data can be canceled out by the stochastic gradient difference term $\frac{\beta}{1-\beta}(\nabla f(\mathbf{x}_t, \boldsymbol{\xi}_t) - \nabla f(\mathbf{x}_{t-1}, \boldsymbol{\xi}_t))$. Based on this term, the estimation of the true gradient for the parameter in the last step $\nabla F(\mathbf{x}_{t-1})$ can be transferred to the true gradient for the parameter in the current step $\nabla F(\mathbf{x}_t)$.

Based on STORM, MARS (Yuan et al., 2025) (see Algorithm 1) introduces a scaling parameter for the gradient-correction term, yielding the corrected gradient:

$$\boldsymbol{c}_t = \nabla f(\mathbf{x}_t, \boldsymbol{\xi}_t) + \underbrace{\gamma_t \frac{\beta}{1-\beta}\big(\nabla f(\mathbf{x}_t, \boldsymbol{\xi}_t) - \nabla f(\mathbf{x}_{t-1}, \boldsymbol{\xi}_t)\big)}_{\text{scaled gradient correction}}, \tag{3.9}$$

where $\gamma_t > 0$ is the scaling coefficient. Moreover, it also applies gradient clipping to the corrected gradient to improve training stability:

$$\widetilde{\mathbf{c}}_t = \text{Clip}(\mathbf{c}_t, 1) = \begin{cases} \frac{\mathbf{c}_t}{\|\mathbf{c}_t\|_2} & \text{if } \|\mathbf{c}_t\|_2 > 1, \\ \mathbf{c}_t & \text{otherwise.} \end{cases} \tag{3.10}$$

[1] In Jordan et al. (2024)'s newest implementation, both $\nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t)$ in $\widetilde{\mathbf{M}_t}$ and in equation 3.2 are substituted with $(1-\beta)\nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t)$. However, in this paper, we just focus on the original algorithm.

Algorithm 1 MARS

1: **input:** $\mathbf{x}_0, \beta, \{\gamma_t\}, \{\eta_t\}$
2: Set $\mathbf{m}_0 \leftarrow \mathbf{0}$ and $\mathbf{x}_1 \leftarrow \mathbf{x}_0$
3: **for** $t = 1,$ **to** T **do**
4: Sample $\boldsymbol{\xi}_t$ and let $\mathbf{c}_t = \nabla f(\mathbf{x}_t, \boldsymbol{\xi}_t) + \gamma_t \frac{\beta}{1-\beta}\big(\nabla f(\mathbf{x}_t, \boldsymbol{\xi}_t) - \nabla f(\mathbf{x}_{t-1}, \boldsymbol{\xi}_t)\big)$
5: if $\|\mathbf{c}_t\|_2 > 1$, then $\widetilde{\mathbf{c}}_t = \frac{\mathbf{c}_t}{\|\mathbf{c}_t\|_2}$ else $\widetilde{\mathbf{c}}_t = \mathbf{c}_t$
6: $\mathbf{m}_t = \beta \mathbf{m}_{t-1} + (1-\beta)\widetilde{\mathbf{c}}_t$
7: $\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} \big\{\eta_t \langle \mathbf{m}_t, \mathbf{x}\rangle + \frac{1}{2}\|\mathbf{x} - \mathbf{x}_t\|^2_{\mathbf{H}_t}\big\}$
8: **end for**

For completeness, the MARS algorithm is summarized in Algorithm 1, where line 7 is the general formula for adaptive gradient methods such as AdamW (Loshchilov & Hutter, 2019), Lion (Chen et al., 2023) and Shampoo (Gupta et al., 2018), with an approximated Hessian matrix $\mathbf{H}_t$. As will be seen later, our work can be viewed as a tailored version of MARS for Muon/Moonlight.

4 METHOD

In this section, we propose the MARS-M optimizer and establish its convergence guarantees.

4.1 MARS-M

We present MARS-M, which is an instantiation of MARS by substituting the update rule equation 3.4 with equation 3.8, and the resulted algorithm is shown in Algorithm 2. MARS-M is an instantiation of MARS tailored to the Moonlight optimizer. It is worth noting that MARS-M is related to MARS-Shampoo in (Yuan et al., 2025), which is another instantiation of MARS based on a simplified version of Shampoo. Specifically, the update rule in MARS-Shampoo is instantiated as:

$$\begin{aligned} \mathbf{U}_t, \mathbf{\Sigma}_t, \mathbf{V}_t &= \mathrm{SVD}(\mathbf{M}_t), \\ \mathbf{X}_{t+1} &= \mathbf{X}_t - \eta_t \mathbf{U}_t \mathbf{V}_t^\top. \end{aligned} \tag{4.1}$$

In practice, the SVD computation can be substituted by approximate numerical methods such as Newton iteration (Lakić, 1998; Higham, 2008; Anil et al., 2020; Bernstein & Newhouse, 2024) and Newton-Schulz iteration (Schulz, 1933; Higham, 2008). The main differences between MARS-M and MARS-Shampoo are that MARS-M scales $\mathbf{O}_t \approx \mathbf{U}_t \mathbf{V}_t^\top$ by $0.2 \cdot \sqrt{\max(m,n)}$ and incorporates weight decay $\lambda \mathbf{X}_t$, both of which are essential for achieving superior performance according to Liu et al. (2025). Moreover, we notice that MARS' style of clipping in equation 3.10 is redundant in MARS-M in practice, then we remove this operation in our algorithm.

Algorithm 2 MARS-M

1: **input:** $\mathbf{X}_0 \in \mathbb{R}^{m\times n}, \lambda, \beta, \{\gamma_t\}, \{\eta_t\}$
2: Set $\mathbf{M}_0 \leftarrow \mathbf{0}$ and $\mathbf{X}_1 \leftarrow \mathbf{X}_0$
3: **for** $t = 1,$ **to** T **do**
4: sample $\boldsymbol{\xi}_t$ and let $\mathbf{C}_t = \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t) + \gamma_t(\frac{\beta}{1-\beta})\big(\nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t) - \nabla f(\mathbf{X}_{t-1}, \boldsymbol{\xi}_t)\big)$
5: $\mathbf{M}_t = \beta \mathbf{M}_{t-1} + (1-\beta)\mathbf{C}_t$
6: $\mathbf{O}_t = \mathrm{NewtonSchulz}(\mathbf{M}_t)$
7: $\mathbf{X}_{t+1} = \mathbf{X}_t - \eta_t(0.2 \cdot \mathbf{O}_t \cdot \sqrt{\max(m,n)} + \lambda \mathbf{X}_t)$
8: **end for**

4.2 APPROXIMATED MARS-M

Since computing the stochastic gradients twice is significantly more expensive, in practice, an approximate version of MARS is usually utilized where $\nabla f(\mathbf{X}_{t-1}, \boldsymbol{\xi}_t)$ is replaced by $\nabla f(\mathbf{X}_{t-1}, \boldsymbol{\xi}_{t-1})$. And lines 4 to 5 in Algorithm 2 can be approximated by:

$$\mathbf{C}_t = \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t) + \gamma_t \left(\frac{\beta}{1-\beta}\right)\big(\nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t) - \nabla f(\mathbf{X}_{t-1}, \boldsymbol{\xi}_{t-1})\big), \tag{4.2}$$

$$\mathbf{M}_t = \beta \mathbf{M}_{t-1} + (1-\beta)\mathbf{C}_t. \tag{4.3}$$

With some calculation, it can be shown that the approximate version of MARS-M is equivalent to the following Muon/Moonlight-style update rule:

$$\mathbf{U}_t = \beta \mathbf{U}_{t-1} + \frac{(1-\gamma_t)(1-\beta)}{\beta} \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t), \tag{4.4}$$

$$\mathbf{M}_t = \beta \mathbf{U}_t + \gamma_t \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t), \tag{4.5}$$

$$\mathbf{O}_t = \text{NewtonSchulz}\left(\mathbf{M}_t\right),$$

$$\mathbf{X}_{t+1} = \mathbf{X}_t - \eta_t (0.2 \cdot \mathbf{O}_t \cdot \sqrt{\max(m,n)} + \lambda \mathbf{X}_t).$$

By comparing equation 4.4 and equation 4.5 with equation 3.5 and equation 3.6, we can see that approximate MARS-M can be viewed as a variant of Moonlight with adjusted momentum parameters $(1-\gamma_t)(1-\beta)/\beta$ and γ_t (in contrast to the coefficients equal to 1 in Moonlight).

From another perspective, we note that equation 3.5 and equation 3.6 in the original Moonlight algorithm can be equivalently expressed as:

$$\mathbf{C}_t = \frac{1}{1-\beta} \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t) + 1 \cdot \left(\frac{\beta}{1-\beta}\right) \left(\nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t) - \nabla f(\mathbf{X}_{t-1}, \boldsymbol{\xi}_{t-1})\right), \tag{4.6}$$

$$\mathbf{M}_t = \beta \mathbf{M}_{t-1} + (1-\beta)\mathbf{C}_t. \tag{4.7}$$

Comparing equation 4.2 with equation 4.6, it can be seen that the approximate version of MARS-M corresponds to Moonlight with the stochastic gradient $\nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t)$ being scaled down by a factor of $(1-\beta)$ and setting $\gamma_t = 1$. The experiments in Section 5 empirically demonstrate that this difference in momentum parameters leads to a noticeable impact on the loss.

4.3 Convergence Analysis

To analyze the convergence of MARS-M, similar to Yuan et al. (2025), we first make the following assumptions:

Assumption 4.1 (Bounded Variance)**.** We assume that the variance of gradient estimator is bounded by σ^2. i.e., for any noise $\boldsymbol{\xi}$, parameter $\mathbf{X}$, and $\nabla F(\mathbf{X}) = \mathbb{E}[\nabla f(\mathbf{X}, \boldsymbol{\xi})]$, there exists a positive σ such that:

$$\mathbb{E}\left[\|\nabla f(\mathbf{X}, \boldsymbol{\xi}) - \nabla F(\mathbf{X})\|_F^2\right] \leq \sigma^2. \tag{4.8}$$

Assumption 4.2 (L-Smoothness)**.** We assume that for arbitrary $\boldsymbol{\xi}$, $f(\mathbf{X}, \boldsymbol{\xi})$ is L-smooth:

$$\|\nabla f(\mathbf{X}, \boldsymbol{\xi}) - \nabla f(\mathbf{Y}, \boldsymbol{\xi})\|_F \leq L \cdot \|\mathbf{X} - \mathbf{Y}\|_F, \ \forall \mathbf{X}, \mathbf{Y}. \tag{4.9}$$

Both assumptions are standard in the literature (Cutkosky & Orabona, 2019; Yuan et al., 2025). We have the following theorem, which guarantees the convergence of MARS-M in Algorithm 2.

Theorem 4.3. In Algorithm 2, under Assumptions 4.1 and 4.2, when choosing $\lambda = 0$, $\eta_t = (s+t)^{-2/3}$, $s \geq 2$, suppose $\beta_{t+1} = 1 - 2\eta_t$, then $\forall T \geq s$, it holds that

$$\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\|\nabla F(\mathbf{X}_t)\|_F \leq \frac{2\sqrt{2LG}}{T^{1/3}} + \frac{2\sqrt{2LB\log(s+T)}}{T^{1/3}} + \frac{2G}{T^{1/3}} + \frac{2B}{T^{1/3}}\log(s+T) - \frac{\sqrt{2}}{4LT^{1/3}} \sum_{t=1}^{T} \frac{\widetilde{M}_{t+1}}{\sqrt{\eta_t}}.$$

where $G = F(\mathbf{X}_1) - \min_{\mathbf{X}} F(\mathbf{X}) + \frac{\sqrt{2}s^{1/3}\sigma^2}{4L} + \frac{3Ln}{2s^{1/3}}$, $B = \left(\frac{2\sqrt{2}\sigma^2}{L} + \frac{3\sqrt{2}Ln}{2}\right)$ and $\widetilde{M}_{t+1}$ is a non-negative value defined in equation C.1.

We defer the proof of the above theorem to Appendix C. It is worth noting that time-varying momentum parameter β_t is required for theoretical analysis, while it can be chosen as a constant in practice.

Theorem 4.3 suggests that MARS-M can achieve a non-asymptotic convergence rate of $\mathcal{O}(T^{-1/3}\log(T))$, the same rate as general MARS as proved in Yuan et al. (2025). As a comparison, Li & Hong (2025), Shen et al. (2025) and Pethick et al. (2025) proved that Muon can achieve only $\mathcal{O}(T^{-1/4})$ convergence rate.

To sum up, MARS-M constitutes a new addition to the MARS family, broadening the framework to encompass more matrix-based optimizers with strong theoretical guarantees.

Table 1: The evaluation results of medium models pre-trained using the FineWeb-Edu 100B dataset (2-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
AdamW	33.53	66.46	45.02	**25.32**	**35.40**	69.10	86.70	**55.80**	52.17
Muon (Moonlight)	33.45	66.33	45.48	24.73	**35.40**	69.21	89.00	54.06	52.21
MARS-M ($\gamma = 0.01$)	34.73	66.12	**46.23**	24.56	34.00	**70.40**	88.70	53.83	52.32
MARS-M ($\gamma = 0.025$)	**35.24**	**67.30**	45.82	24.98	33.40	68.28	**89.10**	54.93	**52.38**

Table 2: The evaluation results of large models pre-trained using the FineWeb-Edu 100B dataset (2-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
AdamW	38.05	70.29	50.30	**26.87**	38.20	70.46	92.10	55.80	55.26
Muon (Moonlight)	37.03	70.08	51.57	25.32	38.20	72.03	91.00	56.43	55.21
MARS-M ($\gamma = 0.01$)	**39.08**	70.16	52.07	24.90	37.60	**72.25**	**93.30**	**56.91**	**55.78**
MARS-M ($\gamma = 0.025$)	37.03	**72.01**	**52.22**	25.10	**39.40**	71.22	90.90	55.64	55.41

Table 3: The evaluation results of XL models pre-trained using the FineWeb-Edu 100B dataset (2-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
Muon (Moonlight)	40.87	71.21	55.80	24.89	**42.40**	74.05	91.90	57.54	57.33
MARS-M ($\gamma = 0.01$)	41.64	**73.57**	57.01	24.75	39.60	73.99	**92.30**	**58.96**	**57.73**
MARS-M ($\gamma = 0.025$)	**42.66**	72.10	**57.09**	**25.57**	39.60	**74.21**	**92.30**	58.09	57.70

5 EXPERIMENTS

5.1 LLM EXPERIMENTS

We evaluate the performances of our algorithm with baseline algorithms[2], including Moonlight and AdamW, on language modeling tasks based on the nanoGPT (Karpathy, 2022) architecture and GPT-2 (Radford et al., 2019) model. We conduct experiments on OpenWebText (Gokaslan et al., 2019) and FineWeb-Edu 100B (Lozhkov et al., 2024) datasets. For OpenWebText, the training and validation datasets contain approximately 9 billion and 4.4 million tokens, respectively; while the training and validation sets of FineWeb-Edu 100B are with 100 billion and 0.1 billion tokens. We train for 100,000 steps with 2,000 warm-up steps, using a context length of 1024 and a total batch size of 480.

We run experiments at four scales: small (125M parameters), medium (355M parameters), large (770M parameters) as well as XL (1.5B parameters). Additionally, we disable biases and set the dropout rate (Srivastava et al., 2014) to 0.0. And we also use Cosine learning rate scheduler and set the gradient clipping threshold to 1.0. For training parameters, for experiments with either OpenWebText or FineWeb-Edu 100B datasets, we perform a grid search over learning rates between $\{5e-4, 1e-3, 3e-3, 5e-3, 6e-3, 1e-2\}$ for Moonlight optimizer (the detailed learning rates are listed in Table 5 in Appendix A). We just apply the same learning rate for MARS-M optimizer. And we use $\beta = 0.95$ for both Moonlight and MARS-M optimizers. Since Moonlight and MARS-M optimizers are designed only for matrix parameters, we optimize the vector-like parameters and embeddings with AdamW with the same learning rate.

For experiments with small models, we use 16 NVIDIA H800 GPUs; and for large models, we implement with 32 NVIDIA H800. Other training hyper-parameters are listed in Appendix A.

5.2 EXPERIMENT RESULTS

We show the zoomed-in curves of training and validation losses for different sizes of models on OpenWebText and FineWeb-Edu 100B datasets in Figures 1-2 and Figures 3-4, respectively. And the entire curves can be found in Appendix B. We also plot the curves for the experiments trained with AdamW in Yuan et al. (2025), and the corresponding learning rates are listed in Table 5, which are also attained from grid search as described in Yuan et al. (2025).

[2]For training efficiency, we use approximated MARS-M for the LLM experiments, since the performances of exact version and approximate version performs similarly as discussed in Yuan et al. (2025). And we compare their differences in computer vision experiments.

It can be observed that the experiments trained with MARS-M display steady improvement on both training and validation losses over Moonlight. Additionally, although the loss for AdamW is lower in the middle of training due to a smaller maximum learning rate, model trained with MARS-M achieves lower losses than that of AdamW in the final phase. Since the learning rates for all the experiments trained with baseline optimizers results from grid search, it can be concluded that MARS-M can indeed improve the performance of large language model training.

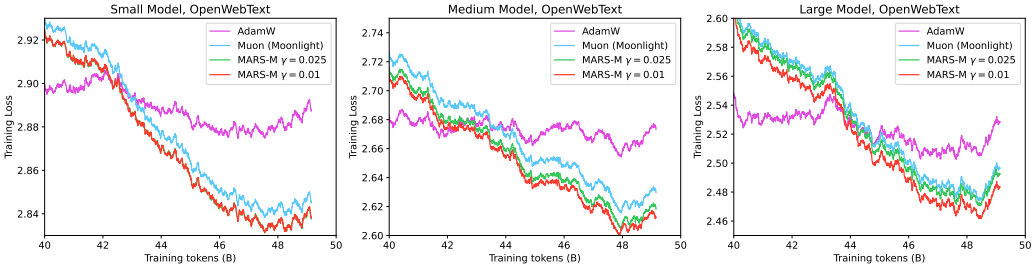

Figure 1: The zoomed-in training loss of small-size (125M), medium-size (355M) and large-size (770M) models trained with different optimizers on the OpenWebText dataset.

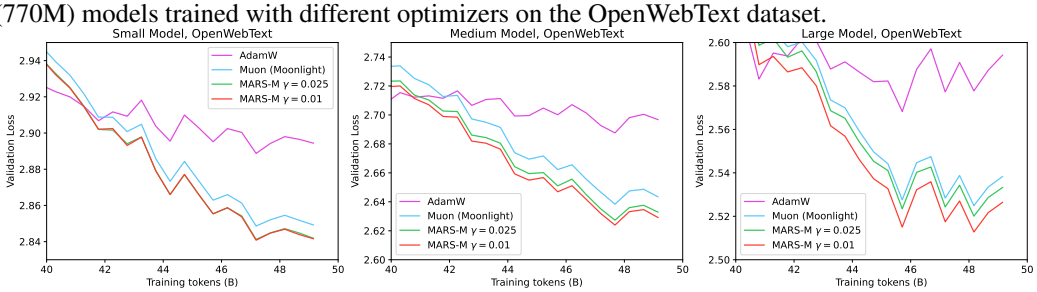

Figure 2: The zoomed-in training and validation loss of small-size (125M), medium-size (355M) and large-size models (770M) trained with different optimizers on the OpenWebText dataset.

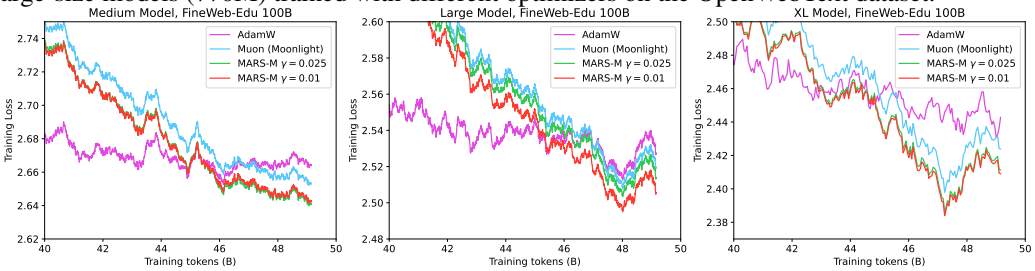

Figure 3: The zoomed-in training loss of medium-size (355M), large-size (770M) and XL-size (1.5B) models trained with different optimizers on the FineWeb-Edu 100B dataset.

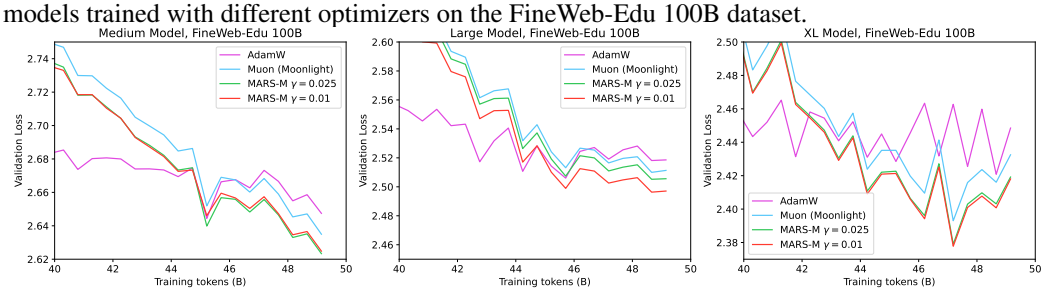

Figure 4: The zoomed-in validation loss of medium-size (355M), large-size (770M) and XL-size (1.5B) models trained with different optimizers on the FineWeb-Edu 100B dataset.

Furthermore, We also evaluate 0-shot and 2-shot performances of these optimizers on benchmarks including ARC (Yadav et al., 2019), HellaSwag (Zellers et al., 2019), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017), WinoGrande (Sakaguchi et al., 2020) and MMLU (Hendrycks et al., 2021), based on `lm-evaluation-harness` codebase (Gao et al., 2024). We only display the 2-shot performances for medium, large and XL models trained on FineWeb-Edu 100B dataset in Tables 1, 2 and 3, respectively. And other results are postponed to Appendix B due to page limitation. It can be observed that MARS-M outperforms Moonlight on

most of the benchmarks, showing that our algorithm can enhance the performance of pre-trained large language models on a wide range of downstream tasks.

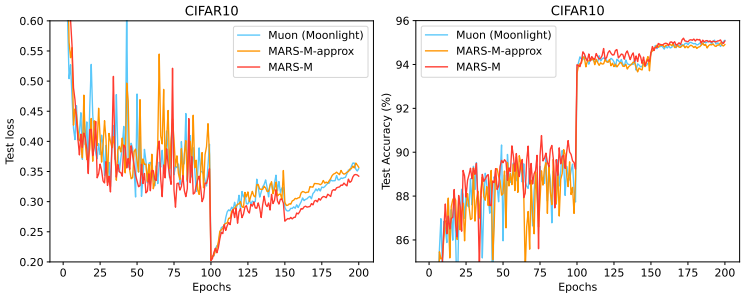

Figure 5: The test loss and test accuracy for different optimizers on CIFAR-10 dataset.

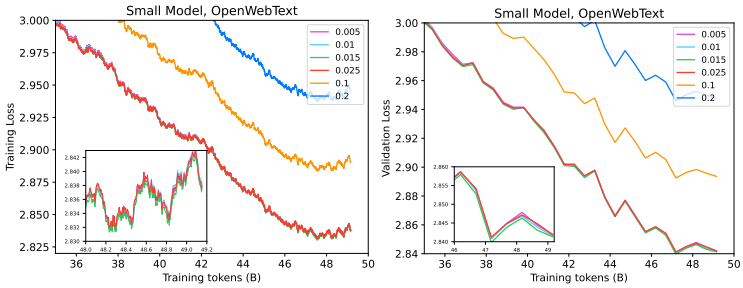

Figure 6: The zoomed-in training and validation loss of small-size models (125M) trained with MARS-M with different γs on the OpenWebText 100B dataset.

5.3 Computer Vision Experiments

Besides language model task, we also conduct experiments on computer vision tasks with these optimizers on CIFAR-10 dataset (Krizhevsky et al., 2009) and ResNet-18 model (He et al., 2016). We set $\gamma = 0.025$ for MARS-M and do grid search over $\{10^{-4}, \cdots, 10^{-2}\}$ for the learning rate. We set $\beta = 0.95$ for all the optimizers. All the experiments are done with a training batch size of 128 on 1 NVIDIA A6000 GPU, with a total of 200 training epochs. Additionally, we use MultiStepLR scheduler so that the learning rate will decrease to 10% at 100th epoch and 1% at 150th epoch.

The test loss and accuracy for MARS-M and MARS-M approximate, as well as Moonlight optimizer, are shown in Figure 5. It can be seen that MARS-M achieves better test accuracy and lower test loss over Moonlight and MARS-M-approx, validating the effect of variance reduction.

5.4 Ablation Study

We also implement an ablation study on γ in MARS-M to investigate how the scale of gradient-correction term affects performance. The experiments are performed on small models trained on OpenWebText datasets, and the results are shown in Figure 6. It can be observed that the scale on gradient correction should be much smaller than 1, as usually taken in most of the variance reduction approaches (Cutkosky & Orabona, 2019; Nguyen et al., 2017a; Fang et al., 2018). However, the training and validation losses for experiments with $\gamma \in [0.005, 0.025]$ are very close, suggesting that a small enough γ is relatively robust to variation.

6 Conclusion

In this work, we have introduced MARS-M, a new algorithm that integrates MARS-based variance reduction with Muon/Moonlight to accelerate large language model training. Our algorithm integrates the benefits of variance reduction and matrix-based optimization. Through empirical experiments on language model and computer vision tasks, we demonstrate that models trained with MARS-M gain steady edges on loss and evaluation on downstream tasks over Moonlight baseline. In summary, our algorithm demonstrates an effective pathway for improving state-of-the-art matrix-based optimizers using variance reduction techniques.

AI USE STATEMENT

In this work, we have not used generative AI tools for Generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Zeyuan Allen-Zhu and Yang Yuan. Improved svrg for non-strongly-convex or sum-of-non-convex objectives. In *International conference on machine learning*, pp. 1080–1089. PMLR, 2016.

Kang An, Yuxing Liu, Rui Pan, Yi Ren, Shiqian Ma, Donald Goldfarb, and Tong Zhang. Asgo: Adaptive structured gradient optimization. *arXiv preprint arXiv:2503.20762*, 2025.

Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. In *OPT 2024: Optimization for Machine Learning*, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36, 2023.

Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32, 2019.

Aaron Defazio and Léon Bottou. On the ineffectiveness of variance reduced optimization for deep learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

Michal Derezinski. Stochastic variance-reduced newton: Accelerating finite-sum minimization with large batches. In *OPT 2023: Optimization for Machine Learning*, 2023.

Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

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

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Murat A Erdogdu and Andrea Montanari. Convergence rates of sub-sampled newton methods. *Advances in Neural Information Processing Systems*, 28, 2015.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in neural information processing systems*, 31, 2018.

Zachary Frangella, Pratik Rathore, Shipu Zhao, and Madeleine Udell. Promise: Preconditioned stochastic optimization methods by incorporating scalable curvature estimates. *Journal of Machine Learning Research*, 25(346):1–57, 2024.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Alon Gonen and Shai Shalev-Shwartz. Faster sgd using sketched conditioning. *arXiv preprint arXiv:1506.02649*, 2015.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nat.*, 645(8081):633–638, 2025. doi: 10.1038/S41586-025-09422-Z.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *Advances in neural information processing systems*, 28, 2015.

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

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Nicholas J. Higham. *Functions of Matrices*. Society for Industrial and Applied Mathematics, 2008.

Feihu Huang, Junyi Li, and Heng Huang. Super-adam: faster and universal framework of adaptive gradients. *Advances in Neural Information Processing Systems*, 34:9074–9085, 2021.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

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

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

Ali Kavis, Stratis Skoulakis, Kimon Antonakopoulos, Leello Tadesse Dadi, and Volkan Cevher. Adaptive stochastic variance reduction for non-convex finite-sum minimization. *Advances in Neural Information Processing Systems*, 35:23524–23538, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Dmitry Kovalev. Understanding gradient orthogonalization for deep learning via non-euclidean trust-region optimization. *arXiv preprint arXiv:2503.12645*, 2025.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Slobodan Lakić. On the computation of the matrix k-th root. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics*, 78(3):167–172, 1998.

Tim Tsz-Kit Lau, Qi Long, and Weijie Su. Polargrad: A class of matrix-gradient optimizers from a unifying preconditioning perspective. *arXiv preprint arXiv:2505.21799*, 2025.

Kfir Levy, Ali Kavis, and Volkan Cevher. Storm+: Fully adaptive sgd with recursive momentum for nonconvex optimization. *Advances in Neural Information Processing Systems*, 34:20571–20582, 2021.

Haochuan Li. *Smoothness and Adaptivity in Nonlinear Optimization for Machine Learning Applications*. PhD thesis, Massachusetts Institute of Technology, 2024.

Jiaxiang Li and Mingyi Hong. A note on the convergence of muon. *arXiv preprint arXiv:2502.02900*, 2025.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations*, 2024.

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

Mingrui Liu, Wei Zhang, Francesco Orabona, and Tianbao Yang. Adam $^+$: A stochastic method with adaptive variance reduction. *arXiv preprint arXiv:2011.11985*, 2020.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pp. 2381–2391. Association for Computational Linguistics, 2018. doi: 10.18653/V1/D18-1260.

Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International conference on machine learning*, pp. 2613–2621. PMLR, 2017a.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Stochastic recursive gradient algorithm for nonconvex optimization. *arXiv preprint arXiv:1705.07261*, 2017b.

OpenAI. Chatgpt. https://chat.openai.com/, 2023.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained lmos. In *Forty-second International Conference on Machine Learning*, 2025.

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

Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pp. 314–323. PMLR, 2016.

Artem Riabinin, Egor Shulgin, Kaja Gruntkowska, and Peter Richtárik. Gluon: Making muon & scion great again!(bridging theory and practice of lmo-based optimizers for llms). In *High-dimensional Learning Dynamics 2025*, 2025.

Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. *Advances in neural information processing systems*, 25, 2012.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 8732–8740. AAAI Press, 2020.

Naoki Sato, Hiroki Naganuma, and Hideaki Iiduka. Convergence bound and critical batch size of muon optimizer. 2025. URL https://api.semanticscholar.org/CorpusID:280140878.

Günther Schulz. Iterative berechnung der reziproken matrix. *Z. Angew. Math. Mech.*, 13:57–59, 1933.

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

Maria-Eleni Sfyraki and Jun-Kun Wang. Lions and muons: Optimization via stochastic frank-wolfe. *arXiv preprint arXiv:2506.04192*, 2025.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(1), 2013.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Wei Shen, Ruichuan Huang, Minhui Huang, Cong Shen, and Jiawei Zhang. On the convergence analysis of muon. *arXiv preprint arXiv:2505.23737*, 2025.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M Kakade. Soap: Improving and stabilizing shampoo using adam for language modeling. In *The Thirteenth International Conference on Learning Representations*, 2024.

Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. Spiderboost and momentum: Faster variance reduction algorithms. *Advances in Neural Information Processing Systems*, 32, 2019.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025.

Vikas Yadav, Steven Bethard, and Mihai Surdeanu. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China*, pp. 2578–2589. Association for Computational Linguistics, 2019. doi: 10.18653/V1/D19-1260.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021.

Yida Yin, Zhiqiu Xu, Zhiyuan Li, Trevor Darrell, and Zhuang Liu. A coefficient makes svrg effective. In *The Thirteenth International Conference on Learning Representations*, 2024.

Huizhuo Yuan, Yifeng Liu, Shuang Wu, Quanquan Gu, et al. Mars: Unleashing the power of variance reduction for training large models. In *Forty-second International Conference on Machine Learning*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 4791–4800. Association for Computational Linguistics, 2019. doi: 10.18653/V1/P19-1472.

Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, et al. Glm-4.5: Agentic, reasoning, and coding (arc) foundation models. *arXiv preprint arXiv:2508.06471*, 2025.

Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduction for nonconvex optimization. *Journal of machine learning research*, 21(103):1–63, 2020.

A Hyper-parameter Settings

We list the architectural hyperparameters for GPT-2 models with 125M (small), 355M (medium), 770M (large) and 1.5B (XL) parameters in Table 4. Moreover, Table 5 display the training learning rates for models with different sizes.

Table 4: Architecture hyperparameters for GPT-2 series models (Radford et al., 2019).

Model	**#Param**	**#Layer**	n_{head}	d_{emb}
GPT-2 small	125M	12	12	768
GPT-2 medium	355M	24	16	1024
GPT-2 large	770M	36	20	1280
GPT-2 XL	1.5B	48	25	1600

Table 5: Learning rates for GPT-2 experiments for different datasets.

Hyper-parameter	**GPT-2 Size**	OpenWebText	FineWebEdu 100B
Max learning rate	small (125M)	6e-3	1e-2
	medium (355M)	5e-3	5e-3
	large (770M)	5e-3	5e-3
	XL (1.5B)	-	3e-3
Min learning rate	small (125M)	3e-5	3e-5
	medium (355M)	6e-5	6e-5
	large (770M)	1e-5	1e-5
	XL (1.5B)	-	1e-5

B Additional Experiments

We just display the curves for training and validation losses for small models on the FineWeb-Edu 100B in Figure 7 with different optimizers including AdamW, original Muon Jordan et al. (2024), MARS-AdamW Yuan et al. (2025), SOAP Vyas et al. (2024), Moonlight Liu et al. (2025) and MARS-M. And we also display the curves for training and validation losses for the entire training process on the OpenWebText and FineWeb-Edu 100B datasets in Figures 8-9 and Figures 10-11, respectively. And the 0-shot and 2-shot evaluation results for these models on two datasets are shown in Tables 6-10 and Tables 11-16, respectively. The results of AdamW are taken from Yuan et al. (2025). According to these figures and tables, it can be observed that MARS-M can achieve better performances than AdamW and Moonlight optimizers in most of these cases.

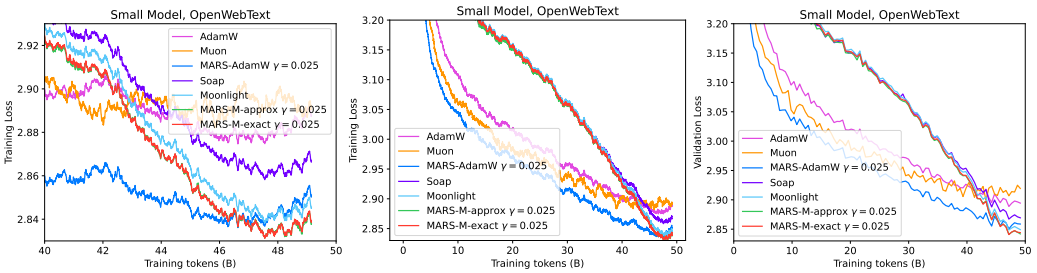

Figure 7: The training and validation loss curves of small-size (125M) models with different optimizers on the OpenWebText dataset. Here Muon is the original version proposed by (Keller Jordan, 2024) and Moonlight is the improved version with factor $0.2\sqrt{\max(m,n)}$ proposed by (Liu et al., 2025). The baseline curves of AdamW, Muon and MARS-AdamW are taken from (Yuan et al., 2025). **Left:** zoomed-in training loss curves. **Middle:** training loss curves. **Right:** Validation loss curves.

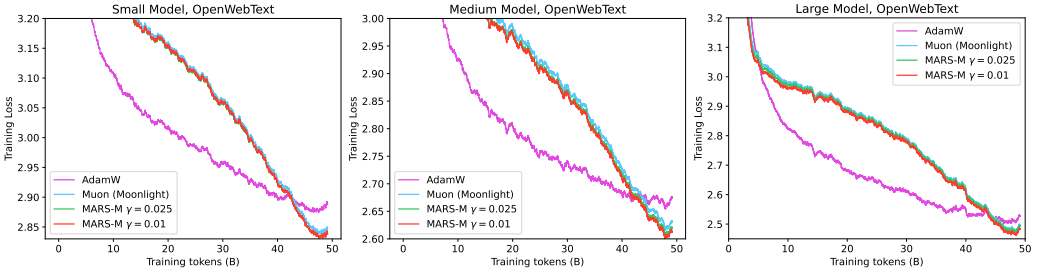

Figure 8: The training loss curves of small-size (125M), medium-size (355M) and large-size (770M) models trained with different optimizers on the OpenWebText dataset.

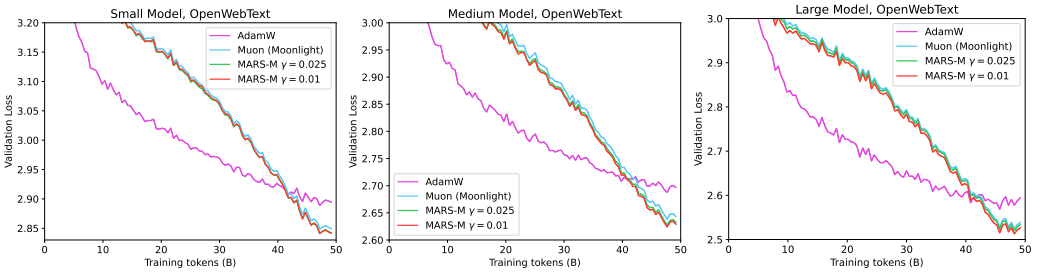

Figure 9: The validation loss curves of small-size (125M), medium-size (355M) and large-size models (770M) trained with different optimizers on the OpenWebText dataset.

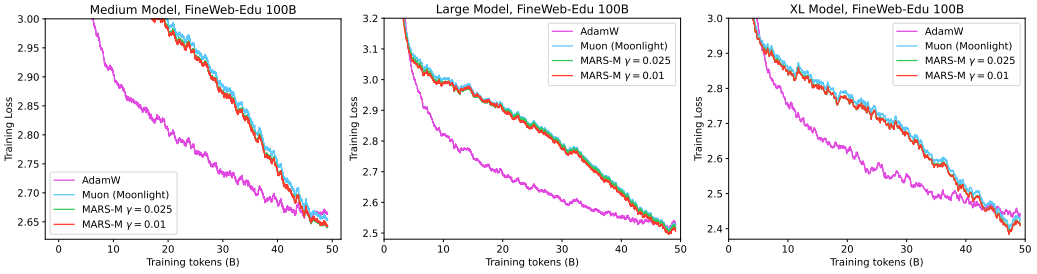

Figure 10: The training loss curves of medium-size (355M), large-size (770M) and XL-size (1.5B) models trained with different optimizers on the FineWeb-Edu 100B dataset.

Table 6: The evaluation results of small models pre-trained using the FineWeb-Edu 100B dataset (0-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
AdamW	26.54	51.43	36.26	24.49	30.60	64.53	71.50	50.36	44.46
Muon (Moonlight)	26.88	52.48	37.58	23.12	**32.80**	64.85	71.20	50.67	44.95
MARS-M ($\gamma = 0.01$)	26.28	51.47	**37.77**	23.26	31.80	**65.94**	70.90	**52.01**	44.93
MARS-M ($\gamma = 0.025$)	**27.82**	**52.57**	37.74	**26.01**	32.00	65.23	**71.90**	51.14	**45.55**

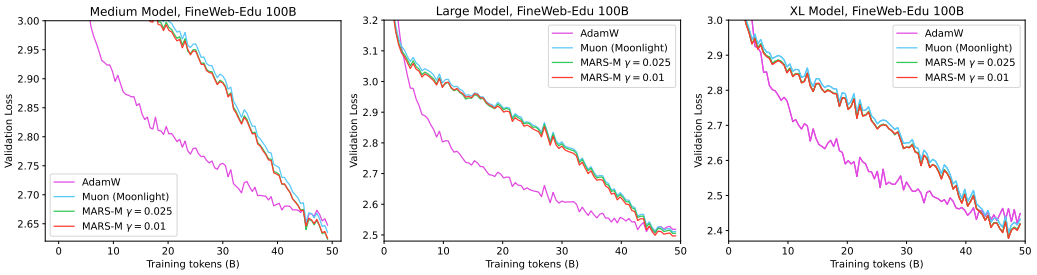

Figure 11: The validation loss curves of medium-size (355M), large-size (770M) and XL-size (1.5B) models trained with different optimizers on the FineWeb-Edu 100B dataset.

Table 7: The evaluation results of small models pre-trained using the FineWeb-Edu 100B dataset (2-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
Muon (Moonlight)	**28.41**	57.62	37.00	25.83	30.00	64.20	82.90	51.38	47.17
MARS-M ($\gamma = 0.01$)	27.82	58.16	**37.50**	25.66	**31.60**	65.02	**86.40**	**53.51**	**48.21**
MARS-M ($\gamma = 0.025$)	**28.41**	**58.42**	37.06	**26.46**	31.20	**65.34**	82.10	50.75	47.47

Table 8: The evaluation results of medium models pre-trained using the FineWeb-Edu 100B dataset (0-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
AdamW	**32.68**	59.34	45.05	24.98	**35.40**	68.88	76.10	54.85	49.66
Muon (Moonlight)	32.17	58.92	45.46	23.28	34.80	69.64	76.80	54.14	49.40
MARS-M ($\gamma = 0.01$)	30.55	59.60	**46.11**	**25.13**	31.80	**69.91**	77.50	54.06	49.33
MARS-M ($\gamma = 0.025$)	32.25	**60.65**	45.92	24.52	33.80	67.90	**78.30**	**56.20**	**49.94**

Table 9: The evaluation results of large models pre-trained using the FineWeb-Edu 100B dataset (0-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
AdamW	34.64	63.22	50.10	25.45	38.60	70.89	80.60	57.54	52.63
Muon (Moonlight)	34.30	63.34	51.93	24.18	**38.80**	**72.31**	81.50	**57.77**	53.07
MARS-M ($\gamma = 0.01$)	**36.18**	**64.52**	**52.29**	**25.89**	37.60	71.65	**82.30**	56.59	**53.45**
MARS-M ($\gamma = 0.025$)	31.74	62.84	51.91	24.35	35.80	70.35	79.70	54.85	51.45

Table 10: The evaluation results of XL models pre-trained using the FineWeb-Edu 100B dataset (0-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
AdamW	38.40	68.22	53.93	25.47	39.00	72.69	**85.30**	54.78	54.72
Muon (Moonlight)	**39.08**	67.30	56.37	**25.32**	**43.20**	**73.78**	83.50	**58.41**	**55.87**
MARS-M ($\gamma = 0.01$)	38.74	68.14	56.83	24.68	39.80	73.23	84.70	57.22	55.42
MARS-M ($\gamma = 0.025$)	38.31	**68.56**	**57.12**	23.57	41.40	73.72	**85.30**	56.75	55.59

Table 11: The evaluation results of small models pre-trained using the OpenWebText dataset (0-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
AdamW	22.27	41.37	31.73	**22.97**	27.80	63.00	**67.50**	**52.01**	41.08
Muon (Moonlight)	**23.81**	**41.67**	33.19	22.83	28.60	63.06	66.50	51.38	41.38
MARS-M ($\gamma = 0.01$)	23.72	40.66	**33.36**	**22.97**	27.80	**74.80**	65.10	51.38	**42.47**
MARS-M ($\gamma = 0.025$)	23.04	40.49	33.44	22.93	**30.20**	63.22	66.40	50.83	41.32

Table 12: The evaluation results of small models pre-trained using the OpenWebText dataset (2-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
Muon (Moonlight)	24.23	43.18	33.20	25.00	26.80	**62.79**	75.20	**53.91**	**43.04**
MARS-M ($\gamma = 0.01$)	**24.49**	**43.52**	**33.47**	**25.35**	26.20	62.02	73.50	51.46	42.50
MARS-M ($\gamma = 0.025$)	23.38	43.43	33.39	23.99	**27.40**	62.40	**77.40**	49.41	42.60

C Proof of Theorem 4.3

For terms for variance reduction, we have the following lemma from Yuan et al. (2025):

Table 13: The evaluation results of medium models pre-trained using the OpenWebText dataset (0-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
AdamW	23.98	43.43	37.76	22.80	27.20	65.56	67.60	52.49	42.60
Muon (Moonlight)	**25.60**	44.99	40.00	23.20	31.20	65.94	**72.00**	52.25	44.40
MARS-M ($\gamma = 0.01$)	24.49	44.49	**40.22**	**23.34**	**33.40**	**67.08**	68.01	**54.46**	**44.44**
MARS-M ($\gamma = 0.025$)	24.57	**46.42**	40.21	23.28	32.00	66.27	70.00	52.64	44.42

Table 14: The evaluation results of medium models pre-trained using the OpenWebText dataset (2-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
Muon (Moonlight)	**27.13**	48.82	39.77	24.93	31.40	67.03	81.30	52.80	46.65
MARS-M ($\gamma = 0.01$)	26.79	**50.17**	**40.87**	**25.42**	28.80	**67.14**	81.60	52.64	46.68
MARS-M ($\gamma = 0.025$)	26.37	48.82	40.53	24.68	**31.60**	65.78	**82.40**	**53.67**	**46.73**

Table 15: The evaluation results of large models pre-trained using the OpenWebText dataset (0-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
AdamW	26.19	46.30	41.70	23.10	31.40	68.12	72.80	51.46	45.13
Muon (Moonlight)	**27.30**	47.85	45.56	23.85	30.80	68.93	72.50	55.64	46.55
MARS-M ($\gamma = 0.01$)	26.88	**49.37**	**45.74**	**24.41**	32.00	**69.37**	**74.30**	53.43	46.94
MARS-M ($\gamma = 0.025$)	27.05	48.48	45.19	23.59	**33.20**	68.82	72.50	**57.30**	**47.02**

Table 16: The evaluation results of large models pre-trained using the OpenWebText dataset (2-shot with lm-evaluation-harness). The best scores in each column are bolded. Abbreviations: WG = WinoGrande.

Method	ARC-C	ARC-E	Hellaswag	MMLU	OpenBookQA	PIQA	SciQ	WG	Avg.
Muon (Moonlight)	27.90	53.45	45.27	**25.77**	**32.80**	**69.53**	84.20	56.12	49.38
MARS-M ($\gamma = 0.01$)	**27.99**	**53.96**	**45.94**	24.76	31.40	68.82	**86.40**	**56.27**	**49.44**
MARS-M ($\gamma = 0.025$)	27.47	53.07	45.33	25.10	31.80	69.21	85.50	55.49	49.12

Lemma C.1 (Yuan et al. (2025))**.** In Algorithm 2. Under Assumption 4.1 and 4.2, if $1 \geq \beta_{t+1} \geq 0$, $\forall t$, denote $\boldsymbol{\Delta}_t := \nabla f(\mathbf{X}_{t+1}, \boldsymbol{\xi}_{t+1}) - \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_{t+1})$, under approximate choice of γ_{t+1}:

$$\gamma_{t+1} = 1 - \frac{\Big(G_{t+1} + \beta_{t+1}\big(\mathbb{E}\|\boldsymbol{\Delta}_t\|_F^2 - \|\mathbb{E}\boldsymbol{\Delta}_t\|_F^2\big)\Big)}{\beta_{t+1}\mathbb{E}\|\boldsymbol{\Delta}_t\|_F^2} = \frac{\beta_{t+1}\|\mathbb{E}\boldsymbol{\Delta}_t\|_F^2 - G_{t+1}}{\beta_{t+1}\mathbb{E}\|\boldsymbol{\Delta}_t\|_F^2},$$

we have

$$\mathbb{E}\|\nabla F(\mathbf{X}_{t+1}) - \mathbf{M}_{t+1}\|_F^2 \leq \beta_{t+1}^2\mathbb{E}\|\nabla F(\mathbf{X}_t) - \mathbf{M}_t\|_F^2 + 2\beta_{t+1}^2L^2\mathbb{E}\|\mathbf{X}_{t+1} - \mathbf{X}_t\|_F^2 + 2(1-\beta_{t+1})^2\sigma^2 - \widetilde{M}_{t+1},$$

where

$$\widetilde{M}_{t+1} := \mathbb{E}\|\boldsymbol{\Delta}_t\|_F^2\left(A_{t+1}^2 - \Big(\beta_{t+1}(1-\gamma_{t+1}) - A_{t+1}\Big)^2\right), \tag{C.1}$$

$$A_{t+1} := \frac{G_{t+1} + \beta_{t+1}\mathrm{tr}\big(\mathrm{Var}\big(\boldsymbol{\Delta}_t\big)\big)}{\mathbb{E}\|\boldsymbol{\Delta}_t\|_F^2},$$

, and

$$G_{t+1} := (1-\beta_{t+1})\mathbb{E}\Big\langle \boldsymbol{\Delta}_t, \nabla f(\mathbf{X}_{t+1}, \boldsymbol{\xi}_{t+1}) - \nabla F(\mathbf{X}_{t+1})\Big\rangle + \beta_{t+1}\mathbb{E}\Big\langle \boldsymbol{\Delta}_t, \mathbf{M}_t - \nabla F(\mathbf{X}_t)\Big\rangle.$$

We also need the following lemmas, which can be seen as an extension of Lemma C.3 in Yuan et al. (2025).

Lemma C.2. For Algorithm 2, under Assumption 4.2, we have the following inequality hold:

$$F(\mathbf{X}_{t+1}) \leq F(\mathbf{X}_t) - \eta_t\|\mathbf{M}_t\|_F + \frac{\eta_t}{\rho}\|\nabla F(\mathbf{X}_t) - \mathbf{M}_t\|_F^2 + \frac{\eta_t\rho}{4}\|\mathbf{O}_t\|_F^2 + \frac{L\eta_t^2}{2}\|\mathbf{O}_t\|_F^2,$$

where $\rho > 0$ is an arbitrarily chosen parameter.

Proof. According to Assumption 4.2, we have the upper bound for the function value:

$$\begin{aligned} F(\mathbf{X}_{t+1}) &\le F(\mathbf{X}_t) + \langle \nabla F(\mathbf{X}_t), \mathbf{X}_{t+1} - \mathbf{X}_t \rangle + \frac{L}{2}\|\mathbf{X}_{t+1} - \mathbf{X}_t\|_F^2 \\ &= F(\mathbf{X}_t) + \langle \nabla F(\mathbf{X}_t), \mathbf{X}_{t+1} - \mathbf{X}_t \rangle + \frac{L}{2}\|\mathbf{X}_{t+1} - \mathbf{X}_t\|_F^2 \\ &= F(\mathbf{X}_t) - \eta_t \langle \mathbf{M}_t, \mathbf{O}_t \rangle - \eta_t \langle \nabla F(\mathbf{X}_t) - \mathbf{M}_t, \mathbf{O}_t \rangle + \frac{L\eta_t^2}{2}\|\mathbf{O}_t\|_F^2 \end{aligned}$$

Here $\mathbf{O}_t = \mathbf{U}_t \mathbf{V}_t^\top$ is defined in equation 4.1. [3]According to the definition of $\mathbf{O}_t$, we have:

$$-\eta_t \langle \mathbf{M}_t, \mathbf{O}_t \rangle = -\eta_t \langle \mathbf{M}_t, \mathbf{U}_r \mathbf{V}_r^\top \rangle = -\eta_t \|\mathbf{M}_t\|_* \le -\eta_t \|\mathbf{M}_t\|_F,$$

where the last inequality follows from $\|\mathbf{M}_t\|_* \ge \|\mathbf{M}_t\|_F$. Moreover, by AM-GM inequality, it holds that

$$-\eta_t \langle \nabla F(\mathbf{X}_t) - \mathbf{M}_t, \mathbf{O}_t \rangle \le \eta_t \left(\frac{1}{\rho}\|\nabla F(\mathbf{X}_t) - \mathbf{M}_t\|_F^2 + \frac{\rho}{4}\|\mathbf{O}_t\|_F^2 \right)$$

for some constant $\rho > 0$. Putting all pieces together, we can obtain

$$F(\mathbf{X}_{t+1}) \le F(\mathbf{X}_t) - \eta_t \|\mathbf{M}_t\|_F + \frac{\eta_t}{\rho}\|\nabla F(\mathbf{X}_t) - \mathbf{M}_t\|_F^2 + \frac{\eta_t \rho}{4}\|\mathbf{O}_t\|_F^2 + \frac{L\eta_t^2}{2}\|\mathbf{O}_t\|_F^2$$

This completes the proof. □

Lemma C.3. Let $\eta_t = (s+t)^{-2/3}, s \ge 1, \forall t \ge 1$. Then $\eta_t^{-1} - \eta_{t-1}^{-1} \le \eta_t^{1/2}, \forall t \ge 1$.

Proof. By the definition of η_t, it holds that

$$\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} = (s+t)^{2/3} - (s+t-1)^{2/3} \le \frac{2}{3(s+t-1)^{1/3}} \le \eta_t^{1/2},$$

where the first inequality follows by the concavity of $h(\mathbf{x}) = x^{2/3}$. This finishes the proof. □

Now we're ready to prove Theorem 4.3.

Proof of Theorem 4.3. [4] First, we define the Lyapunov function as

$$\Phi_t = \mathbb{E}\Big[F(\mathbf{X}_t) + \frac{\rho_t}{16L^2\eta_{t-1}} \cdot \|\nabla F(\mathbf{X}_t) - \mathbf{M}_t\|_F^2 \Big], \quad \forall t \ge 1.$$

where $\rho_t = 4\sqrt{2}L\sqrt{\eta_t}$. Then we calculate the difference between two consecutive Lyapunov functions as:

$$\begin{aligned} \Phi_{t+1} - \Phi_t = &\underbrace{\mathbb{E}[F(\mathbf{X}_{t+1}) - F(\mathbf{X}_t)]}_{I_1} \\ &+ \underbrace{\mathbb{E}\left[\frac{\rho_{t+1}}{16L^2\eta_t} \cdot \|\nabla F(\mathbf{X}_{t+1}) - \mathbf{M}_{t+1}\|_F^2 - \frac{\rho_t}{16L^2\eta_{t-1}} \cdot \|\nabla F(\mathbf{X}_t) - \mathbf{M}_t\|_F^2 \right]}_{I_2}. \end{aligned} \tag{C.2}$$

For I_1, we use Lemma C.2 to obtain

$$I_1 \le \mathbb{E}\Big[-\eta_t \|\mathbf{M}_t\|_F + \frac{\eta_t}{\rho_t} \cdot \|\nabla F(\mathbf{X}_t) - \mathbf{M}_t\|_F^2 + \frac{\eta_t \rho_t}{4}\|\mathbf{O}_t\|_F^2 + \frac{L\eta_t^2}{2}\|\mathbf{O}_t\|_F^2 \Big]. \tag{C.3}$$

[3]In Algorithm 2, $\mathbf{O}_t$ is computed via a (truncated) Newton–Schulz iteration; for the purpose of this analysis, we treat $\mathbf{O}_t$ as the exact polar factor $\mathbf{U}_t \mathbf{V}_t^\top$ and ignore the approximation error.

[4]Here we just ignore the factor $0.2 \cdot \sqrt{\max(m,n)}$ since such a factor can be integrated into η_t.

For I_2, we use Lemma C.1 to obtain

$$
\begin{aligned}
I_2 &= \mathbb{E}\Big[\frac{\rho_{t+1}}{16L^2\eta_t}\cdot\|\nabla F(\mathbf{X}_{t+1})-\mathbf{M}_{t+1}\|_F^2-\frac{\rho_t}{16L^2\eta_{t-1}}\cdot\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F^2\Big]\\
&\le \frac{\rho_t}{16L^2}\cdot\Big(\frac{\beta_{t+1}^2}{\eta_t}-\frac{1}{\eta_{t-1}}\Big)\mathbb{E}\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F^2+\frac{\rho_t\beta_{t+1}^2}{8\eta_t}\cdot\mathbb{E}\|\mathbf{X}_{t+1}-\mathbf{X}_t\|_F^2+\frac{\rho_t(1-\beta_{t+1})^2\sigma^2}{8L^2\eta_t}\\
&\quad-\frac{\rho_t}{16L^2\eta_t}\widetilde{M}_{t+1}\\
&\le \frac{\rho_t}{16L^2}\cdot\Big(\frac{\beta_{t+1}^2}{\eta_t}-\frac{1}{\eta_{t-1}}\Big)\mathbb{E}\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F^2+\frac{\rho_t}{8\eta_t}\cdot\mathbb{E}\|\mathbf{X}_{t+1}-\mathbf{X}_t\|_F^2+\frac{\rho_t\eta_t\sigma^2}{2L^2}-\frac{\rho_t}{16L^2\eta_t}\widetilde{M}_{t+1},
\end{aligned}
\tag{C.4}
$$

where the first inequality is due to $\rho_{t+1}\le\rho_t$, and the last inequality follows from the definition that $\beta_{t+1}=1-2\eta_t$. Further, for the first term on the right hand side, we have

$$
\begin{aligned}
\frac{\rho_t}{16L^2}\cdot\Big(\frac{\beta_{t+1}^2}{\eta_t}-\frac{1}{\eta_{t-1}}\Big)&\le\frac{\rho_t}{16L^2}\cdot\Big(\frac{\beta_{t+1}}{\eta_t}-\frac{1}{\eta_{t-1}}\Big)=\frac{\rho_t}{16L^2}\cdot\Big(\frac{1-2\eta_t}{\eta_t}-\frac{1}{\eta_{t-1}}\Big)\\
&=\frac{\rho_t}{16L^2}\cdot\Big(\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}-2\Big).
\end{aligned}
$$

From Lemma C.3, we know that $\frac{1}{\eta_t}-\frac{1}{\eta_{t-1}}\le\eta_t^{1/2}$. Since $\rho_t^2=32L^2\eta_t$, we obtain[5]

$$
\frac{\rho_t}{16L^2}\cdot\Big(\frac{\beta_{t+1}^2}{\eta_t}-\frac{1}{\eta_{t-1}}\Big)\le\frac{\rho_t}{16L^2}\cdot\Big(\eta_t^{1/2}-2\Big)\le\frac{\rho_t}{16L^2}\cdot\Big(1-2\Big)=-2\eta_t\rho_t^{-1}. \tag{C.5}
$$

Bringing equation C.5 into equation C.4, we arrive at the upper bound for I_2:

$$
I_2\le-\frac{2\eta_t}{\rho_t}\mathbb{E}\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F^2+\frac{\rho_t}{8\eta_t}\cdot\mathbb{E}\|\mathbf{X}_{t+1}-\mathbf{X}_t\|_F^2+\frac{\rho_t\eta_t\sigma^2}{2L^2}-\frac{\rho_t}{16L^2\eta_t}\widetilde{M}_{t+1}. \tag{C.6}
$$

Now combining equation C.2, equation C.3 and equation C.6, we derive

$$
\begin{aligned}
\Phi_{t+1}-\Phi_t\le&-\frac{\eta_t}{\rho_t}\mathbb{E}\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F^2-\eta_t\mathbb{E}\|\mathbf{M}_t\|_F+\frac{\rho_t}{8\eta_t}\cdot\mathbb{E}\|\mathbf{X}_{t+1}-\mathbf{X}_t\|_F^2+\frac{\rho_t\eta_t\sigma^2}{2L^2}\\
&-\frac{\rho_t}{16L^2\eta_t}\widetilde{M}_{t+1}+\frac{\eta_t\rho_t}{4}\|\mathbf{O}_t\|_F^2+\frac{L\eta_t^2}{2}\|\mathbf{O}_t\|_F^2.
\end{aligned}
\tag{C.7}
$$

Moreover, according to the definition of $\mathbf{O}_t$, we have

$$
\|\mathbf{O}_t\|_F^2\le\sum_{i=1}^{n}1=n.
$$

Therefore,

$$
\begin{aligned}
\frac{\rho_t}{8\eta_t}\cdot\mathbb{E}\|\mathbf{X}_{t+1}-\mathbf{X}_t\|_F^2=\frac{\rho_t}{8\eta_t}\cdot\mathbb{E}\|\eta_t\mathbf{O}_t\|_F^2=\frac{\rho_t\eta_t}{8}\cdot\mathbb{E}\|\mathbf{O}_t\|_F^2\le\frac{\rho_t\eta_t n}{8},\\
\frac{\eta_t\rho_t}{4}\|\mathbf{O}_t\|_F^2\le\frac{\rho_t\eta_t n}{4},\\
\frac{L\eta_t^2}{2}\|\mathbf{O}_t\|_F^2\le\frac{L\eta_t^2 n}{2}.
\end{aligned}
$$

[5]As long as $\eta_t\le1/2$, $1\ge\beta_{t+1}=1-2\eta_t\ge0$ is satisfied.

Taking a telescoping sum for $t = 1, \cdots, T$ in equation C.7, and applying $\rho_t = 4\sqrt{2}L\sqrt{\eta_t}$ give

$$\begin{aligned}
&\sum_{t=1}^{T}\Big(\frac{\sqrt{\eta_t}}{4\sqrt{2}L}\mathbb{E}\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F^2+\eta_t\mathbb{E}\|\mathbf{M}_t\|_F\Big)\\
&\le \Phi_1-\Phi_{T+1}+\frac{\sigma^2}{2L^2}\sum_{t=1}^{T}\rho_t\eta_t+\sum_{t=1}^{T}\big(\frac{3\rho_t\eta_t n}{8}+\frac{L\eta_t^2 n}{2}\big)-\sum_{t=1}^{T}\frac{\sqrt{2}}{4L\sqrt{\eta_t}}\widetilde{M}_{t+1}\\
&\le \Phi_1-\Phi_{T+1}+\Big(\frac{2\sqrt{2}\sigma^2}{L}+\frac{3\sqrt{2}Ln}{2}\Big)\sum_{t=1}^{T}\frac{1}{s+t}+\frac{Ln}{2}\sum_{t=1}^{T}\frac{1}{(s+t)^{4/3}}-\sum_{t=1}^{T}\frac{\sqrt{2}}{4L\sqrt{\eta_t}}\widetilde{M}_{t+1}\\
&\le \Phi_1-\Phi_{T+1}+B\cdot\log(s+T)+\frac{3Ln}{2s^{1/3}}-\sum_{t=1}^{T}\frac{\sqrt{2}}{4L\sqrt{\eta_t}}\widetilde{M}_{t+1},
\end{aligned}$$

where $B=\Big(\frac{2\sqrt{2}\sigma^2}{L}+\frac{3\sqrt{2}Ln}{2}\Big)$. By the definition of Φ_t, we have $\Phi_{T+1}\ge F(\mathbf{X}_{T+1})\ge \min_{\mathbf{X}}F(\mathbf{X})$. And for Φ_1,

$$\begin{aligned}
\Phi_1&=\mathbb{E}\Big[F(\mathbf{X}_1)+\frac{\rho_1 s^{2/3}}{16L^2}\cdot\|\nabla F(\mathbf{X}_1)-\mathbf{M}_1\|_F^2\Big]\\
&\le F(\mathbf{X}_1)+\frac{\sqrt{2}s^{1/3}}{4L}\cdot\mathbb{E}[\|\nabla F(\mathbf{X}_1)-\nabla f(\mathbf{X}_1,\boldsymbol{\xi}_1)\|_F^2]\\
&\le F(\mathbf{X}_1)+\frac{\sqrt{2}s^{1/3}\sigma^2}{4L}.
\end{aligned}$$

Consequently, defining $G=F(\mathbf{X}_1)-\min_{\mathbf{X}}F(\mathbf{X})+\frac{\sqrt{2}s^{1/3}\sigma^2}{4L}+\frac{3Ln}{2s^{1/3}}$, the following inequality holds:

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F^2&\le\frac{4\sqrt{2}LG}{T\sqrt{\eta_T}}+\frac{4\sqrt{2}LB}{T\sqrt{\eta_T}}\log(s+T)-\frac{2}{T\sqrt{\eta_T}}\sum_{t=1}^{T}\frac{\widetilde{M}_{t+1}}{\sqrt{\eta_t}}\\
&\le\frac{8LG}{T^{2/3}}+\frac{8LB}{T^{2/3}}\log(s+T)-\frac{2}{T^{2/3}}\sum_{t=1}^{T}\frac{\widetilde{M}_{t+1}}{\sqrt{\eta_t}},
\end{aligned}$$

where the last inequality holds when $T\ge s$. And it also holds that

$$\frac{1}{T}\sum_{t=1}^{T}\eta_t\mathbb{E}\|\mathbf{M}_t\|_F\le\frac{G}{T}+\frac{B}{T}\log(s+T)-\frac{\sqrt{2}}{4LT}\sum_{t=1}^{T}\frac{\widetilde{M}_{t+1}}{\sqrt{\eta_t}}.$$

Therefore,

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathbf{M}_t\|_F&\le\frac{G}{T\eta_T}+\frac{B}{T\eta_T}\log(s+T)-\frac{\sqrt{2}}{4LT\eta_T}\sum_{t=1}^{T}\frac{\widetilde{M}_{t+1}}{\sqrt{\eta_t}}\\
&\le\frac{G(s+T)^{2/3}}{T}+\frac{B(s+T)^{2/3}}{T}\log(s+T)-\frac{\sqrt{2}(s+T)^{2/3}}{4LT}\sum_{t=1}^{T}\frac{\widetilde{M}_{t+1}}{\sqrt{\eta_t}}\\
&\le\frac{2G}{T^{1/3}}+\frac{2B}{T^{1/3}}\log(s+T)-\frac{\sqrt{2}}{4LT^{1/3}}\sum_{t=1}^{T}\frac{\widetilde{M}_{t+1}}{\sqrt{\eta_t}}.
\end{aligned}$$

Finally, by triangle inequality, we have

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\mathbf{X}_t)\|_F &\le \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F+\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathbf{M}_t\|_F\\ &\le \frac{1}{T}\sum_{t=1}^{T}\sqrt{\mathbb{E}\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F^2}+\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathbf{M}_t\|_F\\ &\le \sqrt{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\mathbf{X}_t)-\mathbf{M}_t\|_F^2}+\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\mathbf{M}_t\|_F\\ &\le \frac{2\sqrt{2LG}}{T^{1/3}}+\frac{2\sqrt{2LB\log(s+T)}}{T^{1/3}}+\frac{2G}{T^{1/3}}\\ &\quad+\frac{2B}{T^{1/3}}\log(s+T)-\frac{\sqrt{2}}{4LT^{1/3}}\sum_{t=1}^{T}\frac{\widetilde{M}_{t+1}}{\sqrt{\eta_t}}.\end{aligned}$$

This completes the proof. □

D Convergence Analysis of Approximated MARS-M

In this section, we provide the proof of the $\widetilde{O}(T^{-1/4})$ convergence rate of approximated MARS-M (Algorithm 3). And in the analysis below, we assume $\lambda = 0$ and $0.2\sqrt{\max(m,n)}$ is absorbed in η_t.

Algorithm 3 MARS-M (Approximated)

1: **input:** $\mathbf{X}_0 \in \mathbb{R}^{m\times n}, \lambda, \{\beta_t\}, \{\gamma_t\}, \{\eta_t\}$
2: Set $\mathbf{X}_1 \leftarrow \mathbf{X}_0$ and $\mathbf{M}_0 \leftarrow \nabla f(\mathbf{X}_1, \boldsymbol{\xi}_1)$, assume $\xi_0 = \xi_1$
3: **for** $t = 1$, **to** T **do**
4: sample $\boldsymbol{\xi}_t$ and let $\mathbf{C}_t = \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t) + \gamma_t(\frac{\beta_t}{1-\beta_t})\big(\nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t) - \nabla f(\mathbf{X}_{t-1}, \boldsymbol{\xi}_{t-1})\big)$
5: $\mathbf{M}_t = \beta_t \mathbf{M}_{t-1} + (1-\beta_t)\mathbf{C}_t$
6: $\mathbf{O}_t = \text{NewtonSchulz}(\mathbf{M}_t)$
7: $\mathbf{X}_{t+1} = \mathbf{X}_t - \eta_t(0.2 \cdot \mathbf{O}_t \cdot \sqrt{\max(m,n)} + \lambda \mathbf{X}_t)$
8: **end for**

Lemma D.1 (Gradient Estimation Error Recursion)**.** In Algorithm 3. Under Assumption 4.1 and 4.2, for some constant $a > 0$, if $\eta_t = \eta(s+t)^{-3/4}$, $\gamma_t = \gamma(s+t)^{-1/2} \in (0,1]$, $\beta_{t+1} = 1 - a_t$, where $a_t = a(s+t)^{-1/2}$, $\forall s \ge \max(1, 1/a^2)$, $\forall t$, denote $\mathbf{E}_t = \mathbf{M}_t - \nabla F(\mathbf{X}_t)$, then

$$\mathbb{E}\|\mathbf{E}_{t+1}\|_F^2 \le (1-a_t)\mathbb{E}\|\mathbf{E}_t\|_F^2 + C'(s+t)^{-1},$$

where $C' = \frac{3L^2 n\eta^2}{a} + (3\gamma^2 + 2a^2)\sigma^2$.

Proof. Using the approximate MARS-M update rule, we can rewrite $\mathbf{M}_{t+1}$ as:

$$\mathbf{M}_{t+1} = \beta_{t+1}\mathbf{M}_t + a_t\nabla f(\mathbf{X}_{t+1}, \boldsymbol{\xi}_{t+1}) + \gamma_{t+1}\beta_{t+1}\big(\nabla f(\mathbf{X}_{t+1}, \boldsymbol{\xi}_{t+1}) - \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t)\big)$$

Subtracting $\nabla F(\mathbf{X}_{t+1})$ and grouping terms and denoting $\boldsymbol{\epsilon}_t = \nabla f(\mathbf{X}_t, \boldsymbol{\xi}_t) - \nabla F(\mathbf{X}_t)$, we get:

$$\mathbf{E}_{t+1} = \beta_{t+1}(\mathbf{E}_t - \gamma_{t+1}\boldsymbol{\epsilon}_t) + \beta_{t+1}(1-\gamma_{t+1})\big(\nabla F(\mathbf{X}_t) - \nabla F(\mathbf{X}_{t+1})\big) + (a_t + \gamma_{t+1}\beta_{t+1})\boldsymbol{\epsilon}_{t+1} \tag{D.1}$$

Taking the expected squared norm, the cross-terms with $\boldsymbol{\epsilon}_{t+1}$ vanish because $\boldsymbol{\epsilon}_{t+1}$ is orthogonal to other terms, and by applying $\mathbb{E}\|\boldsymbol{\epsilon}_{t+1}\|_F^2 \le \sigma^2$ according to Assumption 4.1:

$$\mathbb{E}\|\mathbf{E}_{t+1}\|_F^2 = \beta_{t+1}^2\mathbb{E}\|(\mathbf{E}_t - \gamma_{t+1}\boldsymbol{\epsilon}_t) + (1-\gamma_{t+1})(\nabla F(\mathbf{X}_t) - \nabla F(\mathbf{X}_{t+1}))\|_F^2 + (a_t + \gamma_{t+1}\beta_{t+1})^2\sigma^2$$

By Young's inequality $\|\mathbf{A}+\mathbf{B}\|_F^2 \le (1 + a_t/2)\|\mathbf{A}\|_F^2 + (1 + 2/a_t)\|\mathbf{B}\|_F^2$, the first term splits with $\mathbf{A} = \mathbf{E}_t - \gamma_{t+1}\boldsymbol{\epsilon}_t$ and $\mathbf{B} = (1-\gamma_{t+1})(\nabla F(\mathbf{X}_t) - \nabla F(\mathbf{X}_{t+1}))$.

For $\|\mathbf{A}\|_F^2$, according to equation D.1, since $\boldsymbol{\epsilon}_t$ is orthogonal to $\mathbf{E}_{t-1}$, $\boldsymbol{\epsilon}_{t-1}$ and $\nabla F(\mathbf{X}_{t-1}) - \nabla F(\mathbf{X}_t)$, $\mathbb{E}\langle \mathbf{E}_t, \boldsymbol{\epsilon}_t\rangle = \mathbb{E}\langle (a_{t-1} + \gamma_t\beta_t)\boldsymbol{\epsilon}_t, \boldsymbol{\epsilon}_t\rangle \ge 0$. Thus:

$$\mathbb{E}\|\mathbf{E}_t - \gamma_{t+1}\boldsymbol{\epsilon}_t\|_F^2 = \mathbb{E}\|\mathbf{E}_t\|_F^2 - 2\gamma_{t+1}\mathbb{E}\langle \mathbf{E}_t, \boldsymbol{\epsilon}_t\rangle + \gamma_{t+1}^2\mathbb{E}\|\boldsymbol{\epsilon}_t\|_F^2 \le \mathbb{E}\|\mathbf{E}_t\|_F^2 + \gamma_{t+1}^2\sigma^2.$$

Moreover, since $\beta_{t+1} = 1 - a_t$, we have $(1 + a_t/2)\beta_{t+1}^2 \leq 1 - a_t$ for $a_t \leq 1$, and $(1 + a_t/2)\beta_{t+1}^2\|\mathbf{A}\|_F^2 \leq (1 - a_t)\mathbb{E}\|\mathbf{E}_t\|_F^2 + \beta_{t+1}\gamma_{t+1}^2\sigma^2$.

For $\|\mathbf{B}\|_F^2$, we have

$$\mathbb{E}\|\nabla F(\mathbf{X}_t) - \nabla F(\mathbf{X}_{t+1})\|_F^2 \leq L^2\mathbb{E}\|\mathbf{X}_t - \mathbf{X}_{t+1}\|_F^2 = L^2\eta_t^2\mathbb{E}\|\mathbf{O}_t\|_F^2 \leq n\eta_t^2L^2,$$

where the first inequality is due to L-smoothness in Assumption 4.2, the second inequality holds by the definition of $\mathbf{O}_t$. Additionally, since $a_t, \beta_{t+1}, \gamma_{t+1} \in (0, 1]$, $\beta_{t+1}^2(1 + 2/a_t)(1 - \gamma_{t+1})^2 \leq 3/a_t$. Combining everything gives:

$$\mathbb{E}\|\mathbf{E}_{t+1}\|_F^2 \leq (1 - a_t)\mathbb{E}\|\mathbf{E}_t\|_F^2 + \frac{3L^2n\eta_t^2}{a_t} + [(a_t + \gamma_{t+1}\beta_{t+1})^2 + \gamma_{t+1}^2\beta_{t+1}]\sigma^2.$$

Moreover, since $\beta_{t+1} \in (0, 1]$, $(a_t + \gamma_{t+1}\beta_{t+1})^2 + \gamma_{t+1}^2\beta_{t+1} \leq 2a_t^2 + 3\gamma_{t+1}^2$. Applying $a_t = a(s + t)^{-1/2}$, $\gamma_t = \gamma(s + t)^{-1/2}$, and $\eta_t = \eta(s + t)^{-3/4}$, we have $\frac{\eta_t^2}{a_t} = \frac{\eta^2}{a}(s + t)^{-1}$ and $\left(3\gamma_{t+1}^2 + 2a_t^2\right) \leq (3\gamma^2 + 2a^2)(s + t)^{-1}$. Thus:

$$\mathbb{E}\|\mathbf{E}_{t+1}\|_F^2 \leq (1 - a_t)\mathbb{E}\|\mathbf{E}_t\|_F^2 + C'(s + t)^{-1}$$

where $C' = \frac{3L^2n\eta^2}{a} + (3\gamma^2 + 2a^2)\sigma^2$. □

Corollary D.2 (Inductive Bound on $\mathbb{E}\|\mathbf{E}_t\|_F^2$)**.**

$$\mathbb{E}\|\mathbf{E}_t\|_F^2 \leq E_0(s + t)^{-1/2},$$

where $E_0 = \max(\frac{2C'}{a}, \sigma^2(s + 1)^{1/2})$.

Proof. By induction. Assuming it holds for t:

$$\mathbb{E}\|\mathbf{E}_{t+1}\|_F^2 \leq (1 - a(s + t)^{-1/2})E_0(s + t)^{-1/2} + C'(s + t)^{-1}$$

To prove $\mathbb{E}\|\mathbf{E}_{t+1}\|_F^2 \leq E_0(s + t + 1)^{-1/2}$, we need:

$$E_0\left((s + t)^{-1/2} - (s + t + 1)^{-1/2}\right) \leq aE_0(s + t)^{-1} - C'(s + t)^{-1}$$

Since $(s + t)^{-1/2} - (s + t + 1)^{-1/2} \leq \frac{1}{2(s+t)^{3/2}}$, it suffices to show $\frac{E_0}{2(s+t)^{1/2}} \leq aE_0 - C'$, which is guaranteed since $s \geq 1/a^2$ and $E_0 \geq \frac{2C'}{a}$. Moreover, for $\mathbb{E}\|\mathbf{E}_1\|_F^2$, we have

$$\mathbb{E}\|\mathbf{E}_1\|_F^2 = \mathbb{E}[\|\nabla f(\mathbf{X}_1, \boldsymbol{\xi}_1) - \nabla F(\mathbf{X}_1)\|_F^2] \leq \sigma^2,$$

which completes the proof. □

Theorem D.3 (Lyapunov Analysis of Approximated MARS-M)**.** In Algorithm 3. Under Assumptions 4.1 and 4.2, for some constant $a > 0$, if $\eta_t = \eta(s+t)^{-3/4}$, $\gamma_t = \gamma(s+t)^{-1/2} \in (0, 1]$, $\beta_{t+1} = 1-a_t$, where $a_t = a(s + t)^{-1/2}$, $\forall s \geq \max(1, 1/a^2)$, $\forall t \geq s$, it holds that

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\mathbf{X}_t)\|_F \leq \frac{G}{T\eta_T} + (K + \eta\sqrt{E_0})\frac{\log(s + T)}{T\eta_T} \leq \frac{2G}{\eta T^{1/4}} + (K + \eta\sqrt{E_0})\frac{2\log(s + T)}{\eta T^{1/4}},$$

where $G = F(\mathbf{X}_1) - \min_{\mathbf{X}} F(\mathbf{X}) + \frac{\eta\sigma^2}{a\rho}$, $K = \frac{\eta\rho n}{4} + \frac{L\eta^2 n}{2} + \frac{\eta C'}{a\rho}$, $C' = \frac{3L^2n\eta^2}{a} + (3\gamma^2 + 2a^2)\sigma^2$ and $E_0 = \max\left(\frac{2C'}{a}, \sigma^2(s + 1)^{1/2}\right)$.

Proof. We define a carefully balanced Lyapunov function to exactly cancel the residual $\mathbf{E}_t$ terms:

$$\Phi_t = \mathbb{E}[F(\mathbf{X}_t) + \omega\|\mathbf{E}_t\|_F^2],$$

where $\omega = \frac{\eta}{a\rho}$ for some $\rho > 0$. According to Lemma C.2 and applying $\|\mathbf{O}_t\|_F^2 \leq n$,

$$F(\mathbf{X}_{t+1}) \leq F(\mathbf{X}_t) - \eta_t\|\mathbf{M}_t\|_F + \frac{\eta_t}{\rho_t}\|\mathbf{E}_t\|_F^2 + \frac{\eta_t\rho_t}{4}n + \frac{L\eta_t^2}{2}n$$

where we set $\rho_t = \rho(s+t)^{-1/4}$. Then we have

$$\Phi_{t+1} - \Phi_t \leq -\eta_t \mathbb{E}\|\mathbf{M}_t\|_F + \left(\frac{\eta_t}{\rho_t} - \omega a_t\right)\mathbb{E}\|\mathbf{E}_t\|_F^2 + \frac{\eta_t \rho_t}{4} n + \frac{L\eta_t^2}{2} n + \omega C'(s+t)^{-1}. \quad \text{(D.2)}$$

Here $\frac{\eta_t}{\rho_t} = \frac{\eta}{\rho}(s+t)^{-1/2}$ and $a_t = a(s+t)^{-1/2}$, Therefore, $\frac{\eta_t}{\rho_t} - \omega a_t = 0$. Moreover, let $K = \frac{\eta\rho n}{4} + \frac{L\eta^2 n}{2} + \omega C'$. Then

$$\frac{\eta_t \rho_t}{4} n + \frac{L\eta_t^2}{2} n + \omega C'(s+t)^{-1} = (\frac{\eta\rho n}{4} + \frac{L\eta^2 n}{2\sqrt{s+t}} + \omega C')(s+t)^{-1} \leq \frac{K}{s+t}.$$

Summing equation D.2 from $t=1$ to T, we have

$$\sum_{t=1}^{T} \eta_t \mathbb{E}\|\mathbf{M}_t\|_F \leq \Phi_1 - \Phi_{T+1} + K\sum_{t=1}^{T}(s+t)^{-1} \leq \Phi_1 - \Phi_{T+1} + K\log(s+T),$$

By the definition of Φ_t, we have $\Phi_{T+1} \geq F(\mathbf{X}_{T+1}) \geq \min_{\mathbf{X}} F(\mathbf{X})$. And

$$\Phi_1 = \mathbb{E}[F(\mathbf{X}_1) + \omega\|\mathbf{E}_1\|_F^2] \leq F(\mathbf{X}_1) + \omega\sigma^2 = F(\mathbf{X}_1) + \frac{\eta\sigma^2}{a\rho}.$$

Define $G = F(\mathbf{X}_1) - \min_{\mathbf{X}} F(\mathbf{X}) + \frac{\eta\sigma^2}{a\rho}$, we have $\sum_{t=1}^{T} \eta_t \mathbb{E}\|\mathbf{M}_t\|_F \leq G + K\log(s+T)$. Moreover, by triangle inequality, $\mathbb{E}\|\nabla F(\mathbf{X}_t)\|_F \leq \mathbb{E}\|\mathbf{M}_t\|_F + \mathbb{E}\|\mathbf{E}_t\|_F$. Using Corollary D.2, we sum the $\mathbf{E}_t$ errors:

$$\sum_{t=1}^{T} \eta_t \mathbb{E}\|\mathbf{E}_t\|_F \leq \sum_{t=1}^{T} \eta(s+t)^{-3/4}\sqrt{E_0}(s+t)^{-1/4} = \eta\sqrt{E_0}\sum_{t=1}^{T}(s+t)^{-1} \leq \eta\sqrt{E_0}\log(s+T)$$

Therefore,

$$\sum_{t=1}^{T} \eta_t \mathbb{E}\|\nabla F(\mathbf{X}_t)\|_F \leq G + (K + \eta\sqrt{E_0})\log(s+T).$$

Since η_t is decreasing,

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla F(\mathbf{X}_t)\|_F \leq \frac{G}{T\eta_T} + (K+\eta\sqrt{E_0})\frac{\log(s+T)}{T\eta_T} \leq \frac{2G}{\eta T^{1/4}} + (K+\eta\sqrt{E_0})\frac{2\log(s+T)}{\eta T^{1/4}}.$$

This completes the proof. □

E THE ANALYSIS OF THE DYNAMICS

To further investigate the training dynamics of our algorithms, we first plot the training gradient norms, momentum norms as well as gradient clip rate (the proportion of the parameters with gradients greater than 1) in Figure 12. It can be observed that both MARS-M-exact and MARS-M-approx have lower gradient 2-norm, momentum norms and clip rate, showing that our proposed algorithms can achieve faster convergence in terms of gradient than Moonlight optimizer. We have to acknowledge that due to cosine learning rate scheduler, the gradient norm rises in the end. However, most of the time, the much lower decaying gradient of MARS-M still verifies the theoretical convergence bound of MARS-M.

Moreover, we also plot the gradient norms in a simpler experiments to further validate the theoretical convergence rate of the proposed algorithms. Here we train a $y = \tanh(WX)$ model with random Gaussian noise with a variance of 1, where $W \in \mathbb{R}^{H\times d}, X \in \mathbb{R}^d, y \in \mathbb{R}^H$ with $H = 32, d = 64$ and a batch size of 64. The results in Figure 13 show that both MARS-M-exact and MARS-M-approx converges faster than the theoretical convergence rates, consistent with the theoretical upper bound. Moreover, the gradients of MARS-M-exact are smaller than Moonlight, showing that the variance reduction can indeed reduce the scale (and variance) of gradient. And gradients for MARS-M-approx are also smaller in the beginning, indicating that even approximated MARS-M can have a similar reduction effect.

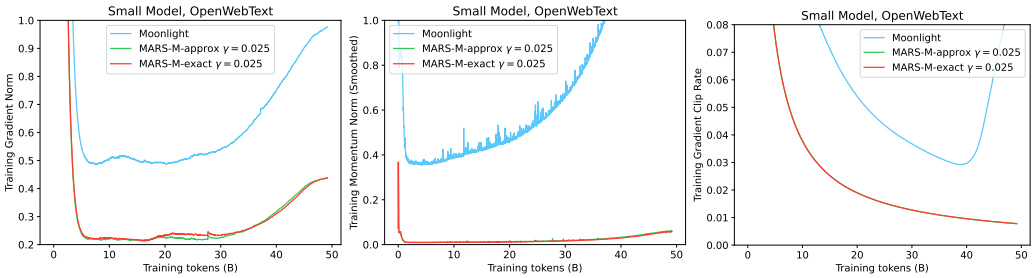

Figure 12: The dynamics during the training of small-size (125M) models with different optimizers on the OpenWebText dataset. The turning up in the final steps may be the result of decaying learning rate. **Left:** the gradient 2-norm. **Middle:** the smoothed training momentum norm. **Right:** the gradient clip rate (since we clip the original gradient to 1).

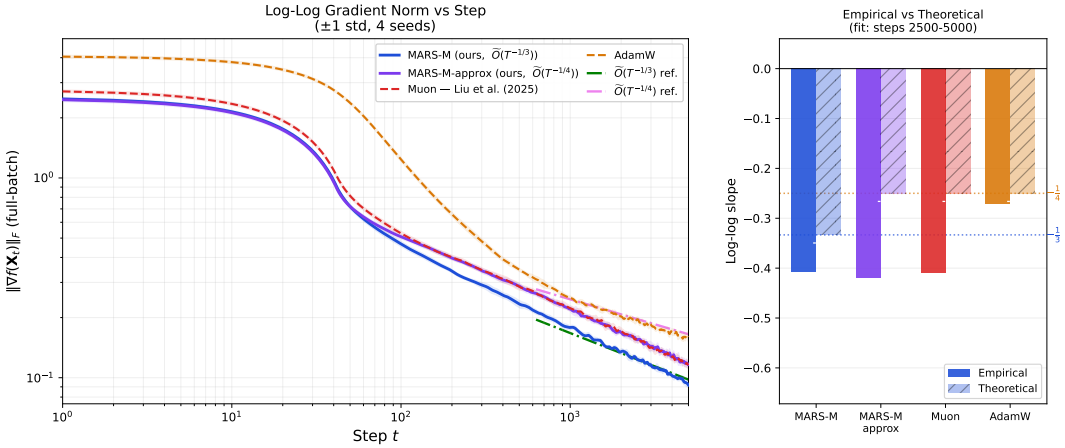

Figure 13: The dynamics of fitting $y = \tanh(WX)$ model with random Gaussian noise with a variance of 1 with different optimizers. Left: The log-log gradient norms w.r.t. step. We use 4 seeds and plot the $\mathcal{O}(T^{-1/4})$ and $\mathcal{O}(T^{-1/3})$ reference lines. Right: the slope of the curves on the log-log plot. Both the MARS-M and MARS-M-approx optimizers achieve a greater slope than the theoretical bound.

F Additional experiments on robustness of γ

To further validate the robustness of the hyperparameter γ, we implemented experiments on medium-size GPT-2 models with different γs on both OpenWebText and FineWeb-Edu 100B datasets. The results are shown in Figures 14 and 15, respectively. The figures demonstrate that the algorithm is highly robust to γ since training and validation losses remain very stable across $\gamma \in [0.005, 0.025]$. In practice, it is recommended to set $\gamma = 0.01$ for MARS-M.

G Memory usage and time efficiency analysis

It is worth noting that MARS-M incurs almost no computational overhead and almost the same memory as Muon. In detail, for compute budget, take GPT-2 small for example, the forward-backward pass requires about 4.20×10^{14} FLOPs, while for optimizer updates, AdamW, MARS-AdamW, MARS-Lion and MARS-M only requires 1.36×10^{9} FLOPs, 2.10×10^{9} FLOPS, 1.24×10^{9} FLOPs and 1.52×10^{12} FLOPs. Therefore, MARS-M only requires 0.36% theoretical compute overhead.

And the time required for training ResNet-18 on CIFAR10 with 1 A6000 GPU for 200 epochs with AdamW, MARS-AdamW, SOAP, Moonlight and MARS-M is 42min, 42min, 68min, 91min and 89min, where the ignorable 2-minute fluctration may be due to the background programs. It can be seen that MARS-M incurs almost no computational overhead compared to Moonlight. Moreover, the time for training small-size GPT-2 models on 8 H100 GPUs with these optimizers is 381min, 397min, 583min, 411min and 423min, showing that our algorithm is data-efficient and with similar

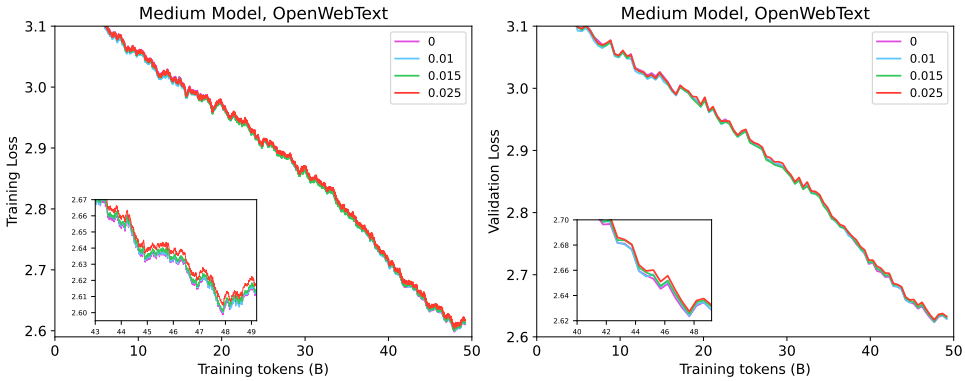

Figure 14: The zoomed-in training and validation loss of medium-size models (355M) trained with MARS-M with different γs on the OpenWebText 100B dataset.

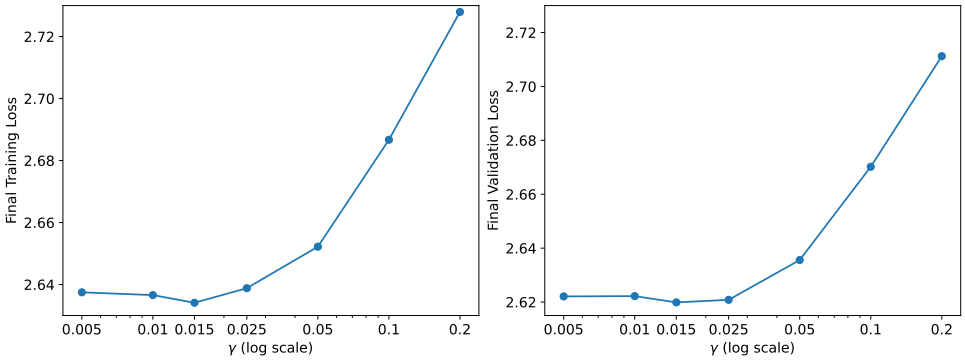

Figure 15: The final training and validation loss of medium-size models (355M) trained with MARS-M with different γs on the FineWeb-Edu 100B dataset.

time-efficiency to matrix-level optimizers. And the GPU memory costs for them are 22,633MB, 24,109MB, 25,873MB, 22,633MB and 22,829MB on a single GPU with micro batch size 15 for GPT-2 small. Considering the superior performance of MARS-M-approx in LLM tasks with almost the same time efficiency, we conclude that MARS-M does improve the performance of training deep learning models.

H Repeated-run uncertainty

Due to the limitation of computation resources, previously we only report results with a fixed seed. To make the results more comparable, we repeat GPT-2 small and medium experiments on OpenWebText with 3 more different seeds. Since different seeds would introduce a great range of span for training curves, we only report the final losses here. And the benchmarks for subsets will also be affected much due to different seeds, so we only report the averaged number of 8 different subsets.

In detail, for GPT-2 small, the final validation losses for AdamW, Moonlight and MARS-M (γ=0.01) are 2.892 ± 0.008, 2.849 ± 0.003, 2.840 ± 0.005, and the average downstream 0-shot performances are 41.05 ± 0.10, 41.42 ± 0.08, 42.43 ± 0.10. And for GPT-2 medium, the final validation losses for AdamW, Moonlight and MARS-M (γ=0.01) are 2.696 ± 0.007, 2.644 ± 0.004 and 2.631 ± 0.004, and the average downstream 0-shot performances are 42.63 ± 0.07, 44.38 ± 0.06, 44.44 ± 0.06. Moreover, it is more comparable among experiments with the same seed, and for each seed, MARS-M can outperform Moonlight. Therefore, MARS-M can indeed improve performance steadily.