Low-rank orthogonalization for large-scale matrix optimization with applications to foundation model training

Chuan He[*] Zhanwang Deng[†] Zhaosong Lu[‡]

September 16, 2025

Abstract

Neural network (NN) training is inherently a large-scale matrix optimization problem, yet the matrix structure of NN parameters has long been overlooked. Recently, the optimizer Muon [28], which explicitly exploits this structure, has gained significant attention for its strong performance in foundation model training. A key component contributing to Muon's success is matrix orthogonalization. In this paper, we propose *low-rank orthogonalization*, which explicitly leverages the low-rank nature of gradients during NN training. Building on this, we propose low-rank matrix-signed gradient descent and a low-rank variant of Muon. Our numerical experiments demonstrate the superior performance of low-rank orthogonalization, with the low-rank Muon achieving promising results in GPT-2 and LLaMA pretraining—surpassing the performance of the carefully tuned vanilla Muon. Theoretically, we establish the iteration complexity of the low-rank matrix-signed gradient descent for finding an approximate stationary solution, as well as that of low-rank Muon for finding an approximate stochastic stationary solution under heavy-tailed noise.

Keywords: Orthogonalization, Muon, foundation model training, iteration complexity, heavy-tailed noise

Mathematics Subject Classification: 49M37, 90C30, 90C90

1 Introduction

Training neural networks (NNs) [32], particularly recent foundation models [8], has consistently posed challenging large-scale optimization problems. Over the past decade, NN training has been dominated by vector-variate optimization methods—including SGD [9], AdaGrad [18], RMSprop [25], Adadelta [55], Adam [29], and AdamW [37]. Nonetheless, these methods disregard the inherent matrix structure of NN parameters—such as those in multi-layer perceptrons [45], convolutional layers [33], and the query, key, and value projections in attention mechanisms [52].

Recently, a shift has taken place: optimization methods that exploit matrix structure are receiving increasing attention and have begun to demonstrate strong performance in foundation model training [28, 35, 41]. These methods focus on solving the matrix optimization problem:

$$\min_{X\in\mathbb{R}^{m\times n}} f(X). \tag{1}$$

[*]Department of Mathematics, Linköping University, Sweden (email: chuan.he@liu.se). The work of Chuan He was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

[†]Academy for Advanced Interdisciplinary Studies, Peking University, Beijing, People's Republic of China (email: dzw_opt2022@stu.pku.edu.cn).

[‡]Department of Industrial and Systems Engineering, University of Minnesota, USA (email: zhaosong@umn.edu).

In particular, Shampoo, developed in [5, 21], applies left and right preconditioning matrices as follows:

$$X^{k+1} = X^k - \eta_k (L^k)^{-1/4} G^k (R^k)^{-1/4}, \tag{2}$$

where $\eta_k > 0$ is the step size, $G^k \in \mathbb{R}^{m\times n}$ denotes the (stochastic) gradient of f at X^k, and $L^k \in \mathbb{R}^{m\times m}$ and $R^k \in \mathbb{R}^{n\times n}$ are the left and right preconditioning matrices, respectively. Shampoo updates the left and right preconditioners $\{L^k\}$ and $\{R^k\}$ using the second-order statistics of the accumulated gradients, similarly to AdaGrad, and has demonstrated comparable performance to popular vector-variate optimizers such as Adam and AdamW on foundation model training. Following Shampoo, other matrix-variate optimizers that use two-sided preconditioners, such as CASPR [19] and SOAP [53], were also developed. A preconditioned Riemannian gradient descent method was developed in [7] for low-rank matrix recovery, which adopts only the diagonal part of the Shampoo preconditioners. Moreover, one-sided preconditioned variants of Shampoo were developed in [4, 54].

In addition to Shampoo and its variants, another matrix-variate optimizer, Muon [28], has attracted significant attention for outperforming standard optimizers such as Adam and AdamW in foundation model training [3, 35]. At each iteration, Muon performs the update:

$$M^k = (1-\theta_{k-1})M^{k-1} + \theta_{k-1} G(X^k;\xi^k), \quad X^{k+1} = X^k - \eta_k \mathrm{msgn}(M^k), \tag{3}$$

where $G(\cdot;\xi)$ is the stochastic gradient of $f(\cdot)$, and $\mathrm{msgn}(M^k) = U^k(V^k)^T$ denotes the matrix sign of M^k, with U^k and V^k being the left and right singular vectors of M^k, respectively. The matrix sign computation is often referred to as matrix orthogonalization, because calculating $\mathrm{msgn}(M)$ is equivalent to finding the (semi-)orthogonal matrix closest to M with respect to the Frobenius norm (see, e.g., [6, Proposition 4]). Muon's empirical success has sparked significant research interest, including efforts to understand its relationship with other algorithms, establish its convergence guarantees, and propose new variants (see, e.g., [4, 14, 15, 20, 30, 31, 34, 36, 38, 41, 43, 46, 47, 48]). A popular interpretation of Muon is from the perspective of a linear minimization oracle with respect to the spectral norm (e.g., see [6, 14, 20, 30, 31, 43]). That is, the matrix sign computation in (3) can be recast as:

$$-\mathrm{msgn}(M^k) = \operatorname*{arg\,min}_{\|\Delta\|\le 1}\{\langle M^k, \Delta\rangle\},$$

where $\|\cdot\|$ denotes spectral norm. Based on this interpretation, algorithmic designs leveraging general matrix-induced norms $\|\cdot\|_{p\to q}$ have been discussed in [6, 14, 20, 43]. In addition, Muon is also connected to earlier algorithms and can be viewed as a special case of Shampoo, despite not explicitly using preconditioners. As discussed in [28], by taking $L^k = G^k(G^k)^T$ and $R^k = (G^k)^T G^k$ in (2), the Shampoo updates reduce to the matrix-signed update: $X^{k+1} = X^k - \eta_k \mathrm{msgn}(G^k)$. Furthermore, convergence guarantees for Muon have been extensively studied (e.g., see [4, 15, 30, 34, 43, 46, 47, 48]), and numerous new variants—such as SWAN [38], Scion [41], Gluon [43], PolarGrad [31], Dion [1, 2], and AdaMuon [49]—have been proposed.

Beyond applying orthogonalization, a key innovation of Muon is the use of a more GPU-friendly method—specifically, Newton-Schulz iterations (typically with five steps)—to perform inexact orthogonalization, which makes Muon well-suited for modern foundation model training. In fact, several earlier methods, including spectral gradient descent [10, 11, 12] and orthogonalized gradient descent [51], have applied orthogonalization using SVD to matrix optimization problems. However, since SVD is not computationally efficient in GPU environments, these methods fail to scale to foundation model training.

Inspired by Muon and its orthogonalization subroutine, we aim at developing a faster, lightweight orthogonalization method to be implemented in Muon and its variants. Specifically, our design leverages the widely observed phenomenon that the gradient matrices of NN parameters are often low-rank (see, e.g., [23, 26, 39, 57]). To exploit this low-rank property, we propose performing low-rank orthogonalization by incorporating well-known matrix approximation techniques for low-rank matrices [16, 22]. Our approach first constructs a low-rank projection of the gradient matrix using QR decomposition on a sketched matrix, and then performs orthogonalization on the projected matrix by leveraging its structure. Our proposed low-rank orthogonalization offers two main advantages over common orthogonalization:

- **Computational efficiency:** Orthogonalization can be seen as computing the polar factor of a given full matrix. In contrast, our low-rank orthogonalization first computes the Q factor of a smaller sketched matrix, followed by the polar factor of a projected matrix constructed using the Q factor. Since both the QR decomposition and polar decomposition are performed on much smaller matrices, our low-rank approach enjoys substantial computational savings for large-scale problems.

- **Noise robustness:** In the presence of noise, singular vectors associated with small singular values often vary significantly, leading to instability in orthogonalization via Newton–Schulz iterations. By contrast, our low-rank orthogonalization method clips these singular vectors to eliminate unstable estimates of singular vectors associated with small singular values, thereby stabilizing the orthogonalization process and yielding a robust estimate of the matrix sign.

These advantages will be illustrated in detail in Sections 3.1 and 4. Based on low-rank orthogonalization, we develop low-rank matrix-signed gradient descent and a low-rank variant of Muon. We also establish their complexity guarantees under mild assumptions.

Our main contributions are highlighted below.

- We propose low-rank orthogonalization to be incorporated into matrix-variate optimization algorithms such as Muon. It can be efficiently executed on GPUs and serves as a lightweight substitute for existing orthogonalization methods such as Newton-Schulz iterations.

- Under mild assumptions, we establish the iteration complexity of low-rank matrix-signed gradient descent and a low-rank variant of Muon, respectively. To the best of our knowledge, our complexity analysis of low-rank Muon provides the first result for Muon-type algorithms—including vanilla Muon—under heavy-tailed noise.

The remainder of this paper is organized as follows. In Section 2, we introduce the notation and assumptions used throughout the paper. In Section 3, we propose low-rank orthogonalization and, based on it, develop low-rank matrix-sign gradient descent and low-rank Muon. In Section 4, we present numerical results. Finally, we provide the proofs of the main results and concluding remarks in Sections 5 and 6, respectively.

2 Notation and assumptions

Throughout this paper, we use $\mathbb{R}^{m\times n}$ to denote the Euclidean space of $m \times n$ real matrices, and $\mathbb{Z}_+$ to denote the set of all nonnegative integers. We use $\|\cdot\|$ to denote the Euclidean norm of a vector or the spectral norm of a matrix; $\|\cdot\|_*$ and $\|\cdot\|_F$ to denote the nuclear norm and the Frobenius norm of a matrix, respectively; and $\langle\cdot,\cdot\rangle$ to denote the trace inner product for matrices. For any $M \in \mathbb{R}^{m\times n}$, we use $\mathrm{rank}(M)$ to denote its rank, and $[M]_k$ to denote its best rank-k approximation with respect to $\|\cdot\|_F$. We define the matrix sign of any nonzero matrix $M \in \mathbb{R}^{m\times n}$ as $\mathrm{msgn}(M) = UV^T$, where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ are column-orthogonal matrices obtained from the reduced SVD of M. We let $\varrho := \min\{m, n\}$. In addition, we use $\widetilde{\mathcal{O}}(\cdot)$ to denote $\mathcal{O}(\cdot)$ with logarithmic factors omitted.

We now make the following assumption throughout this paper.

Assumption 1. (a) *There exists a finite f_{low} such that $f(X) \geq f_{\text{low}}$ for all $X \in \mathbb{R}^{m\times n}$.*

(b) *There exists an $L_* > 0$ such that $\|\nabla f(X) - \nabla f(Y)\|_* \leq L_*\|X - Y\|$ for all $X, Y \in \mathbb{R}^{m\times n}$.*

Assumption 1(a) is standard. Assumption 1(b) is natural in the analysis of Muon-type algorithms and has been used previously (e.g., see [15, 43, 48]). It follows from Assumption 1(b) that

$$f(Y) \leq f(X) + \langle \nabla f(X), Y - X\rangle + \frac{L_*}{2}\|Y - X\|^2 \qquad \forall X, Y \in \mathbb{R}^{m\times n}. \tag{4}$$

We next provide a definition for approximate stationary points of problem (1).

Definition 1. For any $\epsilon \in (0, 1)$, we say that $X \in \mathbb{R}^{m\times n}$ is an ϵ-nuclear norm stationary point (NSP) of problem (1) if it satisfies $\|\nabla f(X)\|_* \leq \epsilon$, and that it is an ϵ-stochastic nuclear norm stationary point (SNSP) of problem (1) if it satisfies $\mathbb{E}[\|\nabla f(X)\|_*] \leq \epsilon$.

3 Matrix optimization with low-rank orthogonalization

In this section, we propose algorithms with low-rank orthogonalization for solving (1). In particular, we first propose low-rank orthogonalization in Section 3.1, which serves as a subroutine in matrix-variate optimization algorithms. Then, we propose low-rank matrix-signed gradient descent methods in Section 3.2, and a low-rank variant of Muon in Section 3.3.

3.1 Low-rank orthogonalization

Orthogonalization has attracted increasing attention in recent optimizer designs, as it has shown strong empirical promise in foundation model training (e.g., see [6, 28, 31, 51]). In this subsection, we develop a low-rank orthogonalization method, leveraging low-rank matrix approximation techniques, that serves as a lightweight substitute for the orthogonalization subroutine used in matrix-variate optimizers.

Specifically, our low-rank orthogonalization method, presented in Algorithm 1, is based on Gaussian sketching [22]. This method first draws a Gaussian random matrix $G \in \mathbb{R}^{n\times r}$ with $r \ll \varrho$, and performs a QR decomposition on MG to obtain a column-orthogonal Q factor $Q \in \mathbb{R}^{m\times r}$. Then, it computes $\operatorname{msgn}(Q^TM) \in \mathbb{R}^{r\times n}$ and returns $M_O = Q\operatorname{msgn}(Q^TM)$ as a low-rank approximation for $\operatorname{msgn}(M)$. As will be shown in Theorem 1, M_O represents the matrix sign of QQ^TM, which is a low-rank approximation of M. Its proof is deferred to Section 5.1.

Algorithm 1 A low-rank orthogonalization method

```
**Input:** matrix $M \in \mathbb{R}^{m\times n}$, rank trial $r \in \mathbb{Z}_+ \cap [1, \varrho]$.
**Output:** approximate matrix sign $M_O \in \mathbb{R}^{m\times n}$.
Draw a Gaussian random matrix $G \in \mathbb{R}^{n\times r}$.
Perform a QR decomposition on $MG$ to obtain a column-orthogonal Q factor $Q \in \mathbb{R}^{m\times r}$.
Return $M_O = Q\operatorname{msgn}(Q^TM)$. (On GPUs, $\operatorname{msgn}(Q^TM)$ is recommended to be estimated via Newton-Schulz iterations.)
```

Theorem 1. *Consider Algorithm 1 with inputs $M \in \mathbb{R}^{m\times n}$ and $r \in \mathbb{Z}_+ \cap [1, \varrho]$, where $\varrho := \min\{m, n\}$. Let $Q \in \mathbb{R}^{m\times r}$ be generated by Algorithm 1. Then, for any r_* satisfying $2 \leq r_* \leq r - 2$, it holds that*

$$\mathbb{E}[\|(I - QQ^T)M\|_F] \leq \Big(1 + \frac{r_*}{r - r_* - 1}\Big)^{1/2}\|M - [M]_{r_*}\|_F. \tag{5}$$

Moreover, we have

$$\mathrm{msgn}(QQ^TM) = Q\mathrm{msgn}(Q^TM). \tag{6}$$

Remark 1. The relation (5) is adapted from [22, Theorem 10.5], where additional guarantees—such as those involving different matrix norms and those providing high-probability bounds—can also be found. In addition, low-rank matrix approximation based on column selection (e.g., see [16, 17]) can also be used to develop a low-rank orthogonalization method. However, since its approximation guarantee is more complicated than that of the Gaussian sketching-based approach, we defer the column-selection-based low-rank orthogonalization method to Algorithm 5 in Appendix A.

Next, we illustrate two major advantages of our low-rank orthogonalization method, namely, *computational efficiency* and *noise robustness*, through synthetic experiments on randomly generated matrices.

Computational efficiency: We compare the computation time on GPUs for calculating inexact matrix sign of high-dimensional matrices using Newton-Schulz iterations, low-rank orthogonalization based on Gaussian sketching (Algorithm 1) and column selection (Algorithm 5), and truncated SVD.

For each $n \in \{1000, 2000, 5000, 10000\}$, we generate 50 random matrices $M \in \mathbb{R}^{n\times n}$, with each entry following the standard Gaussian distribution. We then apply all competing orthogonalization methods to estimate $\mathrm{msgn}(M)$. We implement Newton-Schulz iterations as provided in Muon [28] with 5 iterations. For our low-rank orthogonalization methods, we set $r = 0.1n$ as the input rank parameter, we use command `torch.linalg.qr` to compute the Q factor, and apply 5 iterations of Newton-Schulz scheme following Muon [28] to compute the matrix sign of Q^TM. In addition, we use command `torch.pca_lowrank` for efficiently performing truncated SVD, setting the rank to $0.1n$ for the truncation.

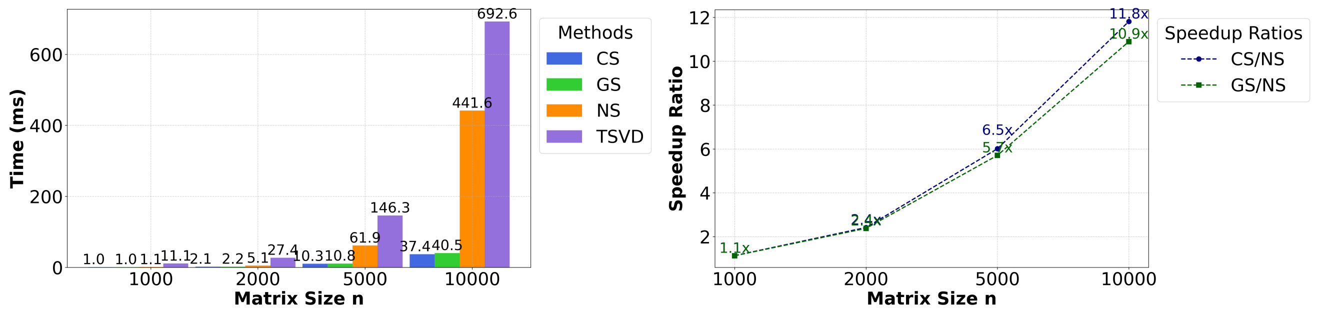

Figure 1: Left: Comparison of GPU computation time across Newton-Schulz iterations (NS), our low-rank orthogonalization with Gaussian sketching (GS) and column selection (CS), and truncated SVD (TSVD). Right: Speedup ratios of our low-rank orthogonalization methods compared to Newton-Schulz iterations.

We present a comparison of the computation time on GPUs for all competing methods in Figure 1, and the time distribution of our low-rank orthogonalization method with Gaussian sketching in Figure 2. From Figure 1, we observe that our low-rank orthogonalization method significantly reduces computation time compared to Newton-Schulz iterations and truncated SVD. Although both truncated SVD and our low-rank orthogonalization exploit low-rank structure for computation, truncated SVD is not well-suited to GPU environments and is therefore slower than the Newton-Schulz iterations. From Figure 2, we observe that in our low-rank orthogonalization method with Gaussian sketching, the QR decomposition accounts for approximately half of the total time, while the Newton-Schulz iterations and other operations (such as generating Gaussian random matrices and performing matrix multiplications) each make up roughly half of the remaining time.

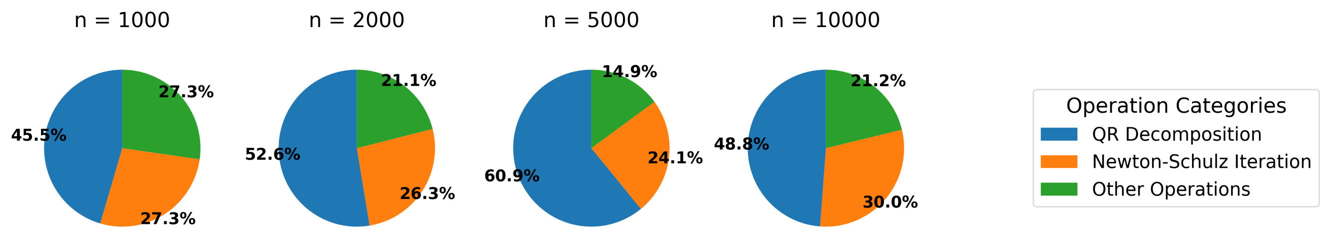

Figure 2: Time distribution for low-rank orthogonalization with Gaussian sketching, including the QR decomposition, the Newton-Schulz iterations for computing the matrix sign, and other computations.

Noise robustness: In addition to reducing computation time, our low-rank orthogonalization methods also produce more robust estimates of the matrix sign for low-rank matrices in the presence of noise. This robustness is particularly important in foundation model training, where gradients often exhibit low-rank structure. We now compare the performance of Newton–Schulz iterations and our low-rank methods in estimating the matrix sign of noisy, low-rank matrices.

For each $n \in \{1000, 2000, 5000, 10000\}$, we first randomly generate 10 nearly low-rank matrices $M \in \mathbb{R}^{n\times n}$ following strategy: the top $0.1n$ singular values are set to 1, the remaining singular values are set to 10^{-4}, and the singular vectors are randomly generated orthogonal vectors. For each M, we generate 50 noise matrices $N \in \mathbb{R}^{n\times n}$, with each entry drawn from a Gaussian distribution with mean zero and variance $\sigma^2 \in \{0.1, 1, 10\}$, and construct noisy matrices $M_N = M + N$. We next apply Newton-Schulz iterations, and our low-rank orthogonalization method based on Gaussian sketching (Algorithm 1), using an input rank parameter $0.1n$, to estimate $\mathrm{msgn}(M_N)$.[1] For each M, we compute the trace of the empirical covariance matrix of the estimates of $\mathrm{msgn}(M_N)$ produced by both Newton-Schulz iterations and Algorithm 1. Both methods are implemented in the same way as in the above synthetic experiments to illustrate the *computational efficiency*.

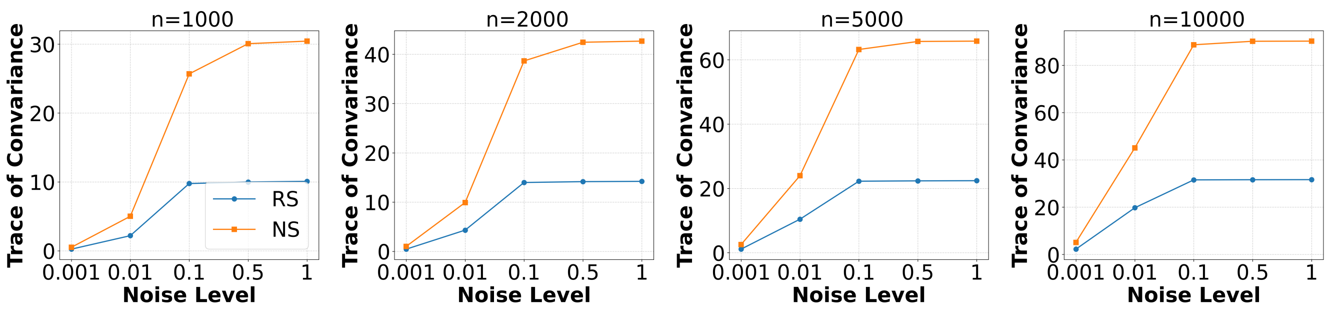

Figure 3: Comparison of variance levels in matrix sign estimation across Newton-Schulz iterations (NS), and our low-rank orthogonalization methods using Gaussian sketching (GS), applied to nearly low-rank matrices in the presence of noise.

Figure 3 presents a comparison of the average trace of the covariance matrices across all competing methods for matrix sign estimation. From this figure, we observe that compared to applying the Newton-Schulz iterations to the full matrix, our low-rank orthogonalization method yields matrix sign estimates that are significantly less sensitive to noise. This is because singular vectors associated with small singular values are more sensitive to noise than those corresponding to large singular values. As a result, our

[1]We omit low-rank orthogonalization with column selection (Algorithm 5) because its performance is similar to that of the version with Gaussian sketching.

low-rank method provides a more stable and robust estimate of the matrix sign function, particularly for nearly low-rank matrices.

3.2 Low-rank matrix-signed gradient descent

In this subsection, we propose low-rank matrix-signed gradient descent methods, including a fixed-rank variant and a safeguarded variant with adaptive ranks. Both variants use low-rank orthogonalization as a subroutine.

Algorithm 2 Low-rank matrix-signed gradient descent

```
Input: starting point $X^0 \in \mathbb{R}^{m\times n}$, rank parameter $r \in \mathbb{Z}_+ \cap [1, \varrho]$, step sizes $\{\eta_k\} \subset (0, \infty)$.
for $k = 0, 1, 2, \ldots$ do
    Call Algorithm 1 (see Section 3.1) with $(M, r) = (\nabla f(X^k), r)$ to obtain an approximate matrix sign
    $M_O^k$, such that $M_O^k = \mathrm{msgn}(M_Q^k)$, where $M_Q^k$ is a low-rank approximation for $\nabla f(X^k)$.
    Update the next iterate:
        $X^{k+1} = X^k - \eta_k M_O^k.$
end for
```

At each iteration $k \geq 0$, the fixed-rank variant invokes Algorithm 1 to compute $M_O^k = \mathrm{msgn}(M_Q^k)$, where M_Q^k is a low-rank approximation of $\nabla f(X^k)$. Then, the next iterate X^{k+1} is obtained by performing a line search update from X^k along the matrix-signed direction $-M_O^k$ with a suitable step size. Details of this method are presented in Algorithm 2.

The following theorem provides a convergence guarantee for Algorithm 2, whose proof is deferred to Section 5.2.

Theorem 2. *Suppose that Assumption 1 holds. Let $\{(X^k, M_O^k)\}$ be the sequence generated by Algorithm 2 with $M_O^k = \mathrm{msgn}(M_Q^k)$ for all $k \geq 0$ and step sizes $\{\eta_k\}$ given by*

$$\eta_k = \frac{1}{(k+1)^{1/2}} \qquad \forall k \geq 0. \tag{7}$$

Then, it holds that for all $K \geq 3$,

$$\min_{0\leq k\leq K-1}\{\|\nabla f(X^k)\|\} \leq \frac{f(X^0) - f_{\mathrm{low}} + L_* \ln K}{K^{1/2}} + \frac{2}{K^{1/2}} \sum_{k=0}^{K-1} \frac{\|\nabla f(X^k) - M_Q^k\|_*}{(k+1)^{1/2}}, \tag{8}$$

where L_* are defined in Assumption 1.

Remark 2. From Theorem 2, we observe that when $\{(X^k, M_Q^k)\}$ satisfies $\sum_{k=0}^{K-1} \|\nabla f(X^k) - M_Q^k\|_*(k+1)^{-1/2} = \widetilde{\mathcal{O}}(1)$, it holds that $\min_{0\leq k\leq K-1}\{\|\nabla f(X^k)\|_*\} = \widetilde{\mathcal{O}}(K^{-1/2})$, which matches, up to a logarithmic factor, the well-established optimal convergence rate for nonconvex optimization (e.g., see [13]). Moreover, if the gradients $\{\nabla f(X^k)\}$ have low rank when k sufficiently large (as is often the case during deep neural network training [57]), we can tune the rank parameter r such that it is close to the effective rank of the gradients, making the sequence $\{\|\nabla f(X^k) - M_Q^k\|_*\}$ to remain close to zero.

We next describe a safeguarded low-rank matrix-signed gradient descent method with adaptively updated ranks. At each iteration $k \geq 0$, this method invokes Algorithm 1 with $(M, r) = (\nabla f(X^k), r_k)$ to obtain $M_O^k = \mathrm{msgn}(M_Q^k)$, where M_Q^k is a low-rank approximation of $\nabla f(X^k)$ such that the approximation error satisfies (9). Then, the next iterate X^{k+1} is obtained by performing a line search update from X^k along the matrix-signed direction $-M_O^k$, with a suitable step size. Details of this method are provided in Algorithm 3.

It is noteworthy that the approximation error in (9) holds for some $r_k \le \varrho$ because when $r_k = \varrho$, the error $\|M_Q^k - \nabla f(X^k)\|$ is zero. When the matrix is low-rank or has a small effective rank, r_k can be chosen to be much smaller than ϱ. In practice, one can gradually increase the trial ranks to find r_k such that (9) holds.

Algorithm 3 Safeguarded low-rank matrix-signed gradient descent

```
**Input:** starting point $X^0 \in \mathbb{R}^{m\times n}$, initial rank trial $r_0 \in \mathbb{Z}_+ \cap [1, \varrho]$, step sizes $\{\eta_k\} \subset (0,\infty)$, control errors $\{\delta_k\} \subset (0,\infty)$
**for** $k = 0, 1, 2, \dots$ **do**
    Call Algorithm 1 (see Section 3.1) with $(M, r) = (\nabla f(X^k), r_k)$ to obtain an approximate matrix sign $M_O^k$ such that $M_O^k = \mathrm{msgn}(M_Q^k)$ and
        $\|\nabla f(X^k) - M_Q^k\|_* \le \delta_k.$    (9)
    Update the next iterate:
        $X^{k+1} = X^k - \eta_k M_O^k.$
**end for**
```

The following theorem establishes an iteration complexity of Algorithm 3, whose proof is deferred to Section 5.2.

Theorem 3. *Suppose that Assumption 1 holds. Let f_{low} and L_* be given in Assumption 1, and define*

$$U_{\mathrm{gd}} := f(X^0) - f_{\mathrm{low}} + L_* + 4. \tag{10}$$

Let $\{X^k\}$ be generated by Algorithm 3 with input parameters $\{(\eta_k, \delta_k)\}$ given by

$$\eta_k = \delta_k = \frac{1}{(k+1)^{1/2}} \qquad \forall k \ge 0. \tag{11}$$

Then, for any $\epsilon \in (0,1)$, it holds that $\min_{0\le k\le K-1}\{\|\nabla f(X^k)\|_*\} \le \epsilon$ for all K satisfying

$$K \ge \max\Big\{\Big(\frac{4U_{\mathrm{gd}}}{\epsilon}\ln\Big(\frac{4U_{\mathrm{gd}}}{\epsilon}\Big)\Big)^2, 3\Big\}.$$

Remark 3. From Theorem 3, we observe that Algorithm 3 achieves an iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-2})$ for finding an ϵ-NSP of (1). This complexity bound matches, up to a polylogarithmic factor, the lower complexity bound as established in [13].

3.3 Low-rank Muon

In this subsection, we propose a low-rank variant of Muon [28], and analyze its iteration complexity under heavy-tailed noise.

This method follows a framework similar to Muon [28], but instead of directly computing the matrix sign of the momentum update, it computes only its low-rank approximation. Specifically, this method generates three sequences $\{X^k\}$, $\{M^k\}$, and $\{M_O^k\}$. At each iteration $k \ge 0$, it first performs a momentum update to generate M^k by aggregating stochastic gradients of f evaluated at $X^0, \ldots, X^k$. Then, Algorithm 1 is invoked to obtain $M_O^k = \mathrm{msgn}(M_Q^k)$, where M_Q^k is a low-rank approximation of $\nabla f(X^k)$ such that the approximation error satisfies (13). The next iterate X^{k+1} is obtained by performing a line search update from X^k along the matrix-signed direction $-M_O^k$ with a suitable step size. Details of this method are given in Algorithm 4.

Before analyzing the complexity of Algorithm 4 for computing an approximate solution to (1), we make the following heavy-tailed noise assumption regarding the stochastic gradient $G(\cdot;\xi)$.

Assumption 2. *The stochastic gradient estimator $G : \mathbb{R}^{m\times n} \times \Xi \to \mathbb{R}^{m\times n}$ satisfies*

$$\mathbb{E}[G(X;\xi)] = \nabla f(X), \quad \mathbb{E}[\|G(X;\xi) - \nabla f(X)\|_F^\alpha] \le \sigma^\alpha$$

for some $\sigma > 0$ and $\alpha \in (1, 2]$.

Algorithm 4 Low-rank Muon

```
**Input:** starting point $X^0 \in \mathbb{R}^{m\times n}$, initial rank trial $r_0 \in \mathbb{Z}_+ \cap [1, \varrho]$, step sizes $\{\eta_k\} \subset (0,\infty)$, weighting parameters $\{\theta_k\} \subset (0,1]$, control errors $\{\delta_k\} \subset (0,\infty)$.
**Initialize:** $M^{-1} = \mathbf{0}_{m\times n}$ and $\theta_{-1} = 1$.
**for** $k = 0, 1, 2, \ldots$ **do**
    Compute the full-rank search direction:
        $M^k = (1-\theta_{k-1})M^{k-1} + \theta_{k-1}G(X^k;\xi^k).$    (12)
    Call Algorithm 1 (see Section 3.1) with $(M, r) = (M^k, r_k)$ to obtain an approximate matrix sign $M_O^k$ such that $M_O^k = \mathrm{msgn}(M_Q^k)$ and
        $\|M^k - M_Q^k\|_* \le \delta_k.$    (13)
    Update the next iterate:
        $X^{k+1} = X^k - \eta_k M_O^k.$    (14)
**end for**
```

The following theorem establishes the iteration complexity of Algorithm 4, whose proof is deferred to Section 5.3.

Theorem 4. *Suppose that Assumptions 1 and 2 hold. Let f_{low} and L_* be given in Assumption 1, and σ and α be given in Assumption 2, and define*

$$U_{\mathrm{mn}} := f(X^0) - f_{\mathrm{low}} + \sigma^\alpha + 2L_* + 4 + 2(\alpha-1)(2\varrho^{1/2}/\alpha)^{\alpha/(\alpha-1)} + 6L_*^\alpha + 4\sigma^\alpha. \tag{15}$$

Let $\{X^k\}$ be generated by Algorithm 4 with input parameters $\{(\eta_k, \theta_k, \delta_k)\}$ given by

$$\eta_k = \frac{1}{(k+1)^{(2\alpha-1)/(3\alpha-2)}}, \quad \theta_k = \frac{1}{(k+1)^{\alpha/(3\alpha-2)}}, \quad \delta_k = \frac{1}{(k+1)^{(\alpha-1)/(3\alpha-2)}} \qquad \forall k \ge 0. \tag{16}$$

Then, for any $\epsilon \in (0,1)$, it holds that $\mathbb{E}[\|\nabla f(X^{\iota_K})\|_*] \le \epsilon$ for all K satisfying

$$K \ge \max\Big\{\Big(\frac{2(3\alpha-2)U_{\mathrm{mn}}}{(\alpha-1)\epsilon}\ln\Big(\frac{2(3\alpha-2)U_{\mathrm{mn}}}{(\alpha-1)\epsilon}\Big)\Big)^{(3\alpha-2)/(\alpha-1)}, 3\Big\},$$

where ι_K is uniformly drawn from $\{0, \ldots, K-1\}$.

Remark 4. From Theorem 4, one can observe that Algorithm 4 achieves an iteration complexity of $\widetilde{\mathcal{O}}(\epsilon^{-(3\alpha-2)/(\alpha-1)})$ for finding an ϵ-SNSP of problem (1), which matches the optimal dependence on ϵ in the complexity results for vector-variate stochastic first-order methods under heavy-tailed noise (see, e.g., [24, 56]). To the best of our knowledge, our result is the first to show that the Muon-type algorithm can achieve an iteration complexity with optimal dependence on ϵ when applied to matrix-variate optimization under heavy-tailed noise.

4 Numerical experiments

In this section, we present numerical experiments to evaluate the performance of low-rank Muon (Algorithm 4) and compare our method with vanilla Muon, AdamW, and SGD. The experiments are conducted on a synthetic matrix regression problem (Section 4.1) and foundation model pretraining with GPT-2 (Section 4.2) and LLaMA (Section 4.3). All experiments are executed on a server with two NVIDIA A100 GPUs (80 GB).

4.1 Matrix regression

In this subsection, we consider the matrix regression problem:

$$\min_{X\in\mathbb{R}^{n\times n}} \left\{ f(X) = \frac{1}{2}\|AXB - C\|_F^2 \right\}, \tag{17}$$

where $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{n\times p}$, and $C \in \mathbb{R}^{p\times p}$ are given data matrices. We simulate low-rank structure for the gradients $\nabla f(X)$ by choosing $n \gg p = 100$. For each $n \in \{10^3, 5\times 10^3, 10^4\}$, we set $A = U_A\Sigma_A V_A^T$ and $B = U_B\Sigma_B V_B^T$, where $\Sigma_A = \Sigma_B = \mathrm{Diag}([2,\ldots,2^p])$, and U_A, V_A, U_B, and V_B are randomly generated column-orthogonal matrices. We also generate a ground truth solution X^* with all entries independently drawn from the standard Gaussian distribution, and set $C = AXB + 10^{-3}E$, where all entries of E are independently drawn from the standard Gaussian distribution.

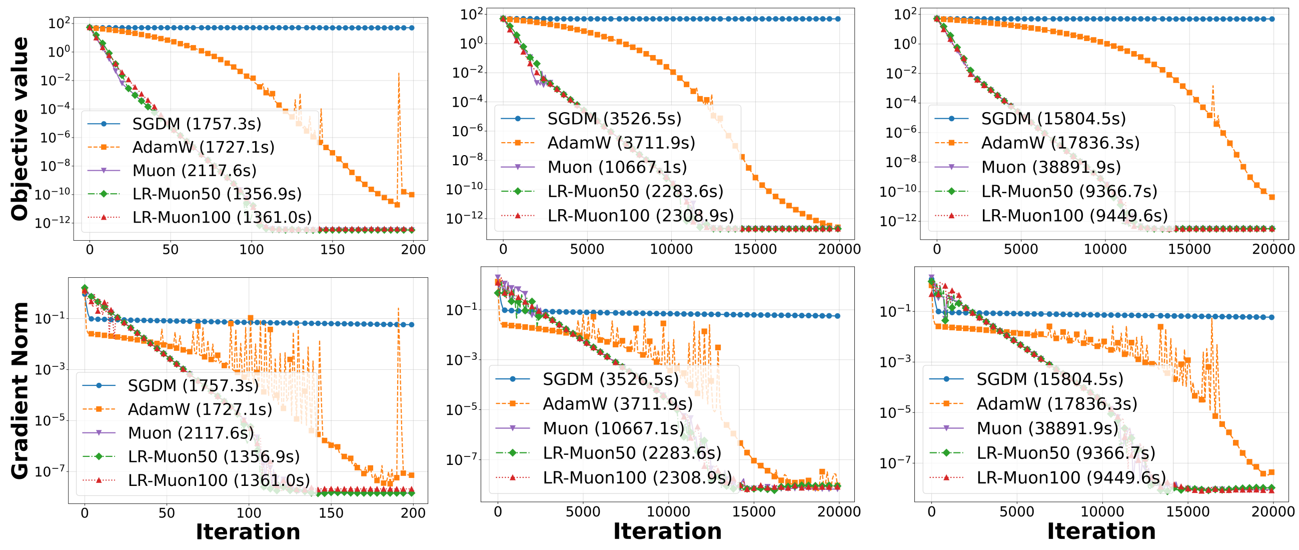

Figure 4: Objective values (top row) and gradient Frobenius norms (bottom row) for all methods during the first 2×10^4 iterations. Results for $n = 10^3$, 5×10^3, and 10^4 are shown in the left, middle, and right columns, respectively. The computation times on GPUs are displayed in the legend for each plot.

We apply low-rank Muon, Muon, AdamW, and SGD to solve (17). We implement two low-rank Muon variants with rank parameters 50 and 100, abbreviated as LR-Muon50 and LR-Muon100, respectively. All methods are implemented with full-batch (deterministic) gradients and initialized from the zero matrix. We compare their performance based on the objective values and the Frobenius norm of the gradient for problem (17), evaluated over the first 2×10^4 iterations. The algorithmic parameters are selected to suit each method well in terms of computational performance.

For each n, we plot the objective values and the gradient Frobenius norms in Figure 4 to illustrate the convergence behavior of all methods. Figure 4 demonstrates that our low-rank Muon achieves performance comparable to vanilla Muon in iteration count, while significantly outperforming both SGD and AdamW. In addition, SGD fails to converge effectively for problem (17). Furthermore, our low-rank Muon demonstrates significantly faster computation times on GPUs compared to other variants, highlighting the advantages of our low-rank orthogonalization procedures for computational efficiency.

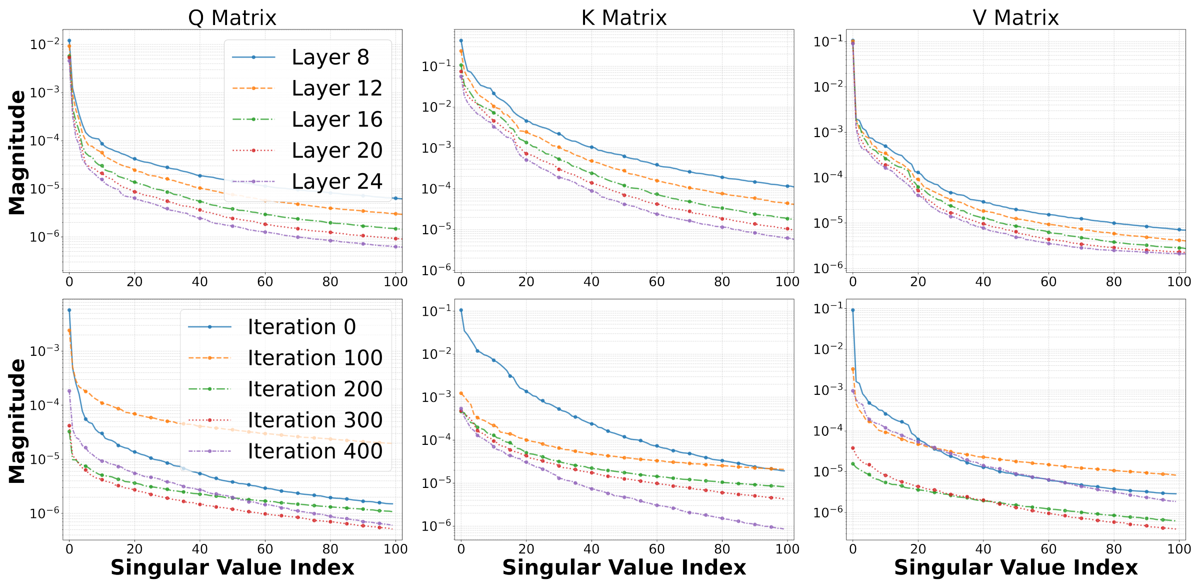

Figure 5: Singular value distributions of the Q, K, and V matrices: across layers at iteration 300 (top row) and over training iterations at layer 16 (bottom row) during GPT-2 pretraining.

4.2 GPT-2 pretraining

In this subsection, we consider pretraining GPT-2 [42], a transformer-based language model. We experiment with GPT-2 models of sizes 60M, 135M, 350M, and 1B on the same three datasets tested in the Muon GitHub repository [28]: FineWeb10B, FineWeb100B, and FineWebEdu10B.

We apply low-rank Muon, Muon, AdamW, and SGD with momentum (SGDM) for GPT-2 pretraining. We implement two low-rank Muon variants with rank parameters 100 and 200, abbreviated as LR-Muon100 and LR-Muon200, respectively. Following the experiments in [27, 31], we use AdamW to train the embedding and head layers for Muon and low-rank Muon. We initialize all methods with the same weights from pretrained GPT-2 models, and terminate all methods after one training epoch, consisting of 5000 iterations. We compare all methods using validation perplexity, a standard metric in foundation model training (e.g., [42, 50, 52]), which is defined as the exponential of the validation loss and serves to amplify performance differences. The algorithmic parameters are carefully tuned to suit each method well in terms of computational performance. More details about the experimental setups including algorithm and GPT-2 model parameters are deferred to Appendix B.

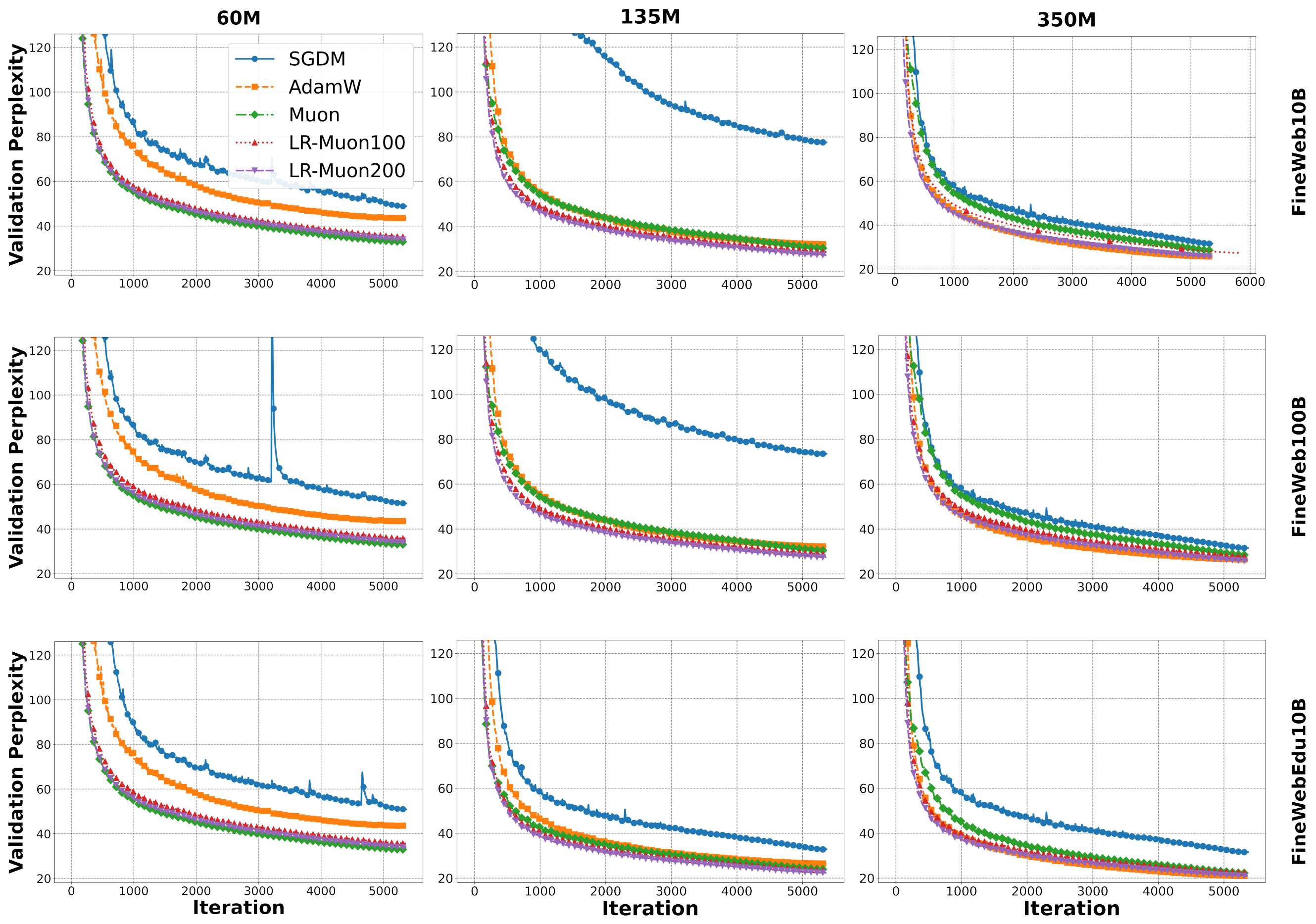

Figure 6: Comparison of validation perplexity versus computational time for all competing methods in training GPT-2 models.

Table 1: Comparison of validation perplexity and computational time for all competing methods on GPT-2 pretraining.

Dataset	Method	Validation Perplexity				Computational Time			
		60M	135M	350M	1B	60M	135M	350M	1B
FineWeb10B	SGDM	48.85	48.85	31.54	27.86	2.51e3	7.08e3	1.20e4	2.70e5
	AdamW	43.56	43.56	25.72	21.58	2.52e3	7.08e3	1.21e4	2.69e5
	Muon	**32.89**	32.99	28.48	20.99	2.76e3	7.57e3	1.32e4	3.04e5
	LR-Muon100	35.21	28.83	27.32	21.70	2.69e3	7.49e3	1.25e4	2.90e5
	LR-Muon200	33.98	**27.34**	**25.64**	**20.69**	2.79e3	7.54e3	1.26e4	2.91e5
FineWeb100B	SGDM	51.48	51.48	77.50	29.75	2.69e3	7.25e3	1.04e4	2.52e5
	AdamW	43.52	43.51	27.47	23.53	2.70e3	7.21e3	1.05e4	2.56e5
	Muon	34.04	33.02	32.76	20.75	2.79e3	7.70e3	1.19e4	3.09e5
	LR-Muon100	35.78	30.19	27.59	21.76	2.77e3	7.62e3	1.09e4	2.87e5
	LR-Muon200	**34.02**	**28.31**	**25.86**	20.45	2.80e3	7.69e3	1.10e4	2.91e5
FineWebEdu10B	SGDM	50.92	77.50	31.54	25.49	2.56e3	7.09e3	1.20e4	2.53e5
	AdamW	43.56	26.45	21.01	22.45	2.59e3	7.09e3	1.20e4	2.56e5
	Muon	**32.88**	23.89	22.23	19.54	2.80e3	7.63e3	1.33e4	3.05e5
	LR-Muon100	35.60	23.50	22.39	19.97	2.74e3	7.53e3	1.26e4	2.85e5
	LR-Muon200	33.93	**22.37**	**20.96**	**18.85**	2.83e3	7.59e3	1.29e4	2.87e5

We visualize the top 100 singular values of the momentum updates $\{M^k\}$ associated with the Q, K, and V matrices for a 350M GPT-2 model trained by Muon on the FineWeb100B dataset in Figure 5. The first and second rows of the figure display the top 100 singular values across different training iterations and neural network layers, respectively. This visualization partly supports the low-rank nature of the momentum updates.

We present a comparison of the validation perplexity and computational time in Table 1. From this table, we observe that for GPT-2 with a model size of 60M, our low-rank Muon achieves validation perplexity worse than Muon but better than AdamW and SGDM. For GPT-2 models with larger sizes, our low-rank Muon achieves better validation perplexity than all other competing methods. We also observe that the computational time of our low-rank Muon improves upon that of vanilla Muon, though all Muon-type methods remain slower than SGDM and AdamW. These observations show that our low-rank orthogonalization enables obtaining more generalizable GPT-2 models when the model size is large, but may be less effective for smaller models. In addition, our low-rank orthogonalization saves computational time compared to orthogonalization by Newton-Schulz iterations, but the gain is not as significant as those discussed in Sections 3.1 and 4.1, since the forward and backward passes of neural networks account for the majority of the computational time.

We plot the convergence behavior of validation perplexity versus training iterations in Figure 6. From this figure, we observe that for each model size, the competing methods exhibit similar performance patterns across the three datasets. For the model size of 60M, we observe that all Muon-type methods consistently outperforms AdamW and SGDM, while our low-rank Muon performs slightly worse than the vanilla Muon. For the 135M model size, we observe that our low-rank Muon outperforms the vanilla Muon and other training methods. The vanilla Muon performs similarly to AdamW, and all methods show a large improvement compared to SGDM. For the model size of 350M, we see that our low-rank Muon has slightly better performance than AdamW, while outperforming Muon and SGDM by a larger margin. These observations support that our low-rank Muon improves upon Muon in GPT-2 training and is more effective for models with larger sizes.

With all the above observations and discussions, we conclude that our low-rank Muon is a viable alternative to existing training methods and is particularly more effective than other methods for training GPT-2 models with larger sizes.

4.3 LLaMA pretraining

In this subsection, we consider pretraining LLaMA [50], a transformer-based language model with a more refined architecture than GPT-2. We experiment with LLaMA models of sizes 60M, 135M, 350M, and 1B on the same three datasets tested in the Muon GitHub repository [28]: FineWeb10B, FineWeb100B, and FineWebEdu10B.

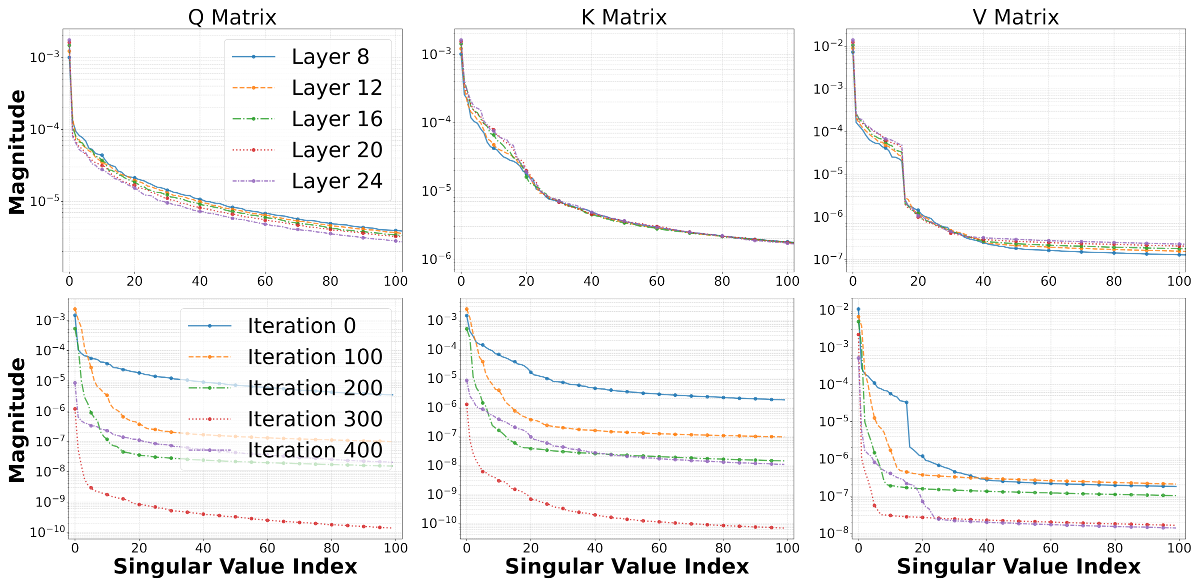

Figure 7: Singular value distributions of the Q, K, and V matrices: across layers at iteration 300 (top row) and over training iterations at layer 16 (bottom row) during LLaMA pretraining.

We apply low-rank Muon, Muon, AdamW, and SGD with momentum (SGDM) for pretraining LLaMA. Similar to Section 4.2, we implement two versions of low-rank Muon with rank parameters 100 and 200, and we use AdamW to train the embedding and head layers for Muon and low-rank Muon. We initialize all methods with the same weights from pretrained LLaMA models, and terminate all methods after one training epoch, consisting of 5000 iterations. We compare all methods using validation perplexity. The algorithmic parameters are carefully tuned to suit each method well in terms of computational performance. More details about the experimental setups including algorithm and LLaMA model parameters are deferred to Appendix B.

We visualize the top 100 singular values of the momentum updates $\{M^k\}$ associated with the Q, K, and V matrices for a 350M LLaMA model trained by Muon on the FineWeb100B dataset in Figure 7. The first and second rows of the figure display the top 100 singular values across different training iterations and neural network layers, respectively. This visualization partly supports the low-rank nature of the momentum updates.

We present a comparison of the validation perplexity and computational time in Table 2. From this table, we observe that our low-rank Muon consistently achieves better validation perplexity compared to Muon for the majority of model sizes and tested datasets, while vastly improving upon the validation

Table 2: Comparison of validation perplexity and computational time for all competing methods on LLaMA pretraining.

Dataset	Method	Validation Perplexity				Computational Time			
		60M	135M	350M	1B	60M	135M	350M	1B
FineWeb10B	SGDM	94.90	86.99	86.49	44.71	2.49e3	5.27e3	1.20e4	2.77e5
	AdamW	50.51	40.17	35.76	24.14	2.46e3	5.27e3	1.21e4	2.78e5
	Muon	36.68	37.96	36.43	20.99	2.59e3	5.69e3	1.35e4	3.18e5
	LR-Muon100	37.33	**34.08**	33.54	21.70	2.69e3	5.66e3	1.28e4	2.79e5
	LR-Muon200	**36.05**	34.37	**32.37**	**20.69**	2.70e3	5.70e3	1.30e4	2.91e5
FineWeb100B	SGDM	81.59	77.50	87.49	37.56	2.60e3	5.44e3	1.05e4	2.65e5
	AdamW	41.49	40.54	35.47	23.89	2.61e3	5.47e3	1.06e4	2.69e5
	Muon	38.49	37.66	36.66	21.76	2.58e3	5.63e3	1.07e4	3.16e5
	LR-Muon100	37.68	34.58	33.71	22.86	2.63e3	5.55e3	1.10e4	2.79e5
	LR-Muon200	**36.26**	**34.20**	**32.73**	**20.98**	2.75e3	5.69e3	1.11e4	2.88e5
FineWebEdu10B	SGDM	82.63	73.48	73.48	47.56	2.57e3	5.27e3	1.20e4	2.76e5
	AdamW	41.82	32.96	29.12	22.54	2.60e3	5.29e3	1.20e4	2.78e5
	Muon	**28.57**	30.72	29.47	22.67	2.68e3	5.84e3	1.33e4	3.25e5
	LR-Muon100	30.52	27.72	27.24	21.36	2.72e3	5.66e3	1.24e4	2.88e5
	LR-Muon200	29.29	**27.65**	**26.87**	**19.54**	2.81e3	5.70e3	1.26e4	2.98e5

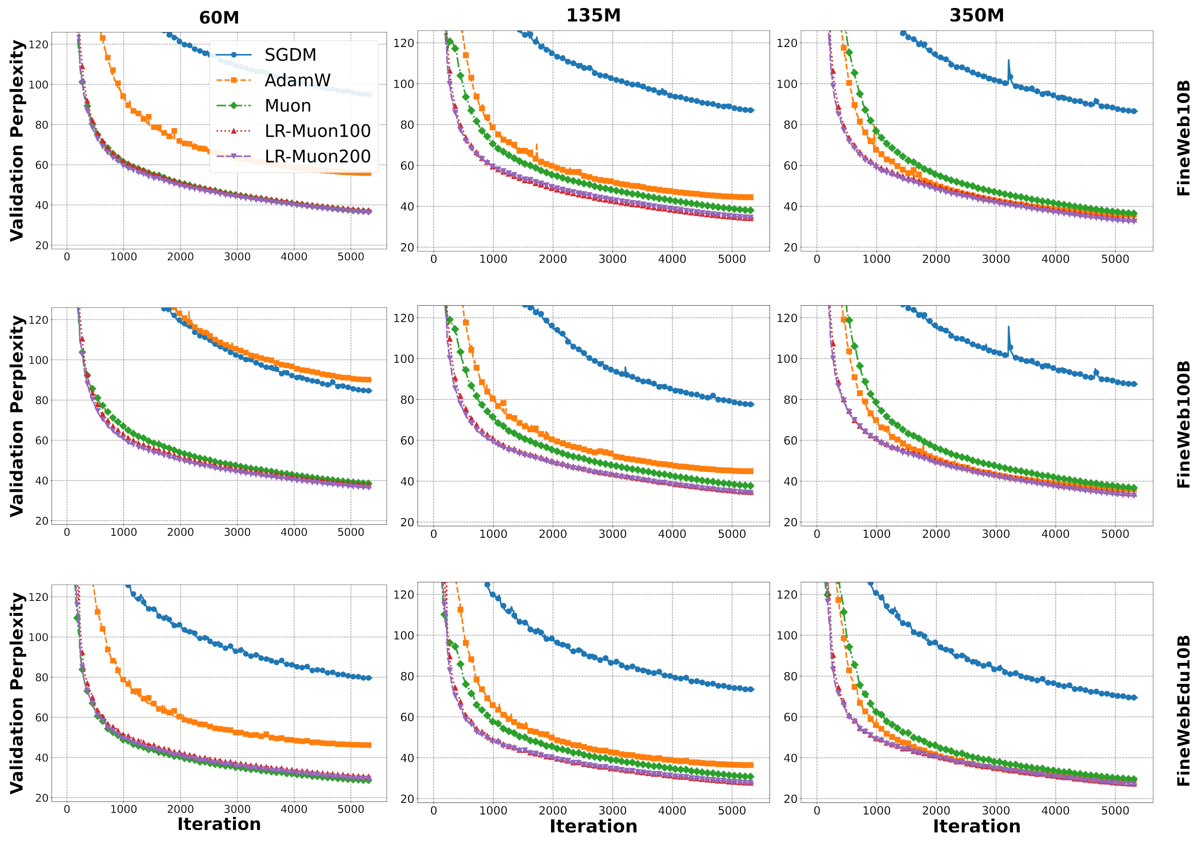

Figure 8: Comparison of validation perplexity and computational time for all competing methods in training LLaMA models.

perplexity achieved by AdamW and SGDM. We also observe that the computational time of our low-rank Muon improves upon that of vanilla Muon, though all Muon-type methods remain slower than SGDM and AdamW. These observations demonstrate that our low-rank orthogonalization produces more generalizable LLaMA models across all tested model sizes and datasets. Moreover, it reduces computational time compared to the Newton-Schulz iterations for orthogonalization, although the improvement is less significant than the gains discussed in Sections 3.1 and 4.1. This is because most of the computational time is dominated by the forward and backward passes of the neural networks.

We plot the convergence behavior of validation perplexity versus training iterations in Figure 8. From this figure, we observe that for the model size of 60M, we observe that all Muon-type methods significantly outperforms AdamW and SGDM, while our low-rank Muon has similar performance compared to the vanilla Muon. For the 135M and 350M model size, we observe that our low-rank Muon significantly outperforms the vanilla Muon and other training methods. These observations support that our low-rank Muon consistently outperforms all other methods for training LLaMA models.

Based on the above observations, we conclude that our low-rank Muon improves upon existing methods for training LLaMA models across the model sizes and datasets we tested.

5 Proof of the main results

In this section, we provide the proofs of our main results presented in Section 3, specifically Theorems 1 to 4.

We first provide three technical lemmas, whose proofs can be found in [24, Lemmas 1 to 3] and are therefore omitted here. The next lemma provides an expansion for the α-power of the Frobenius norm, generalizing the well-known identity $\|U+V\|_F^2 = \|U\|_F^2 + 2\langle U, V\rangle + \|V\|_F^2$ and inequality $\|U+V\|_F^2 \le (1+c)\|U\|_F^2 + (1+1/c)\|V\|_F^2$ for all $U, V \in \mathbb{R}^{m\times n}$ and $c > 0$.

Lemma 1. *For any $\alpha \in (1,2]$, it holds that*

$$\|U+V\|_F^\alpha \le \|U\|_F^\alpha + \alpha\|U\|_F^{\alpha-2}\langle U, V\rangle + 2\|V\|_F^\alpha \qquad \forall U, V \in \mathbb{R}^{m\times n}, \tag{18}$$

$$\|U+V\|_F^\alpha \le (1+c)\|U\|_F^\alpha + (2 + (\alpha-1)^{\alpha-1}c^{1-\alpha})\|V\|_F^\alpha \qquad \forall U, V \in \mathbb{R}^{m\times n}, c > 0. \tag{19}$$

The next lemma provides an estimation of the partial sums of series.

Lemma 2. *Let $\zeta(\cdot)$ be a convex univariate function. Then we have $\sum_{r=a}^b \zeta(r) \le \int_{a-1/2}^{b+1/2}\zeta(\tau)\mathrm{d}\tau$ for any integers a, b satisfying $[a-1/2, b+1/2] \subset \mathrm{dom}\,\zeta$. Consequently, one has*

$$\sum_{r=a}^b \frac{1}{r^\beta} \le \begin{cases} \ln\left(b+\frac{1}{2}\right) - \ln\left(a-\frac{1}{2}\right) & \text{if } \beta = 1, \\ \frac{1}{1-\beta}\left(\left(b+\frac{1}{2}\right)^{1-\beta} - \left(a-\frac{1}{2}\right)^{1-\beta}\right) & \text{if } \beta \in (0,1)\cup(1,+\infty). \end{cases} \tag{20}$$

We next give a lemma that will be used to derive complexity bounds for our methods.

Lemma 3. *Let $\beta \in (0,1)$ and $u \in (0, e^{-1})$ be given. Then, it holds that $v^{-\beta}\ln v \le 2u\beta^{-1}$ for all $v \ge (u^{-1}\ln(u^{-1}))^{1/\beta}$.*

We next establish a descent property for f along a matrix-signed direction.

Lemma 4. *Suppose that Assumption 1 holds. Let $X, M \in \mathbb{R}^{m\times n}$ and $\eta > 0$ be given, and let $X^+ = X - \eta\mathrm{msgn}(M)$. Then we have*

$$f(X^+) \le f(X) - \eta\|\nabla f(X)\|_* + 2\eta\|\nabla f(X) - M\|_* + \frac{L_*\eta^2}{2}, \tag{21}$$

where L_* is given in Assumption 1.

Proof. By the definition of the matrix-sign function, one has $\|\mathrm{msgn}(M)\| \leq 1$, and $\langle M, \mathrm{msgn}(M)\rangle = \|M\|_*$. It then follows from these and (4) with $Y = X^+$ that

$$\begin{aligned}
f(X^+) &\overset{(4)}{\leq} f(X) + \langle \nabla f(X), X^+ - X\rangle + \frac{L_*}{2}\|X^+ - X\|^2 \\
&= f(X) + \langle M, X^+ - X\rangle + \langle \nabla f(X) - M, X^+ - X\rangle + \frac{L_*}{2}\|X^+ - X\|^2 \\
&\leq f(X) - \eta\langle M, \mathrm{msgn}(M)\rangle + \eta\|\nabla f(X) - M\|_*\|\mathrm{msgn}(M)\| + \frac{L_*\eta^2}{2}\|\mathrm{msgn}(M)\|^2 \\
&= f(X) - \eta\|M\|_* + \eta\|\nabla f(X) - M\|_* + \frac{L_*\eta^2}{2} \\
&\leq f(X) - \eta\|\nabla f(X)\|_* + 2\eta\|\nabla f(X) - M\|_* + \frac{L_*\eta^2}{2},
\end{aligned}$$

where the second inequality is due to $X^+ = X - \eta\mathrm{msgn}(M)$ and the trace Hölder inequality, the second equality is due to $\|\mathrm{msgn}(M)\|_* \leq 1$ and $\langle M, \mathrm{msgn}(M)\rangle = \|M\|_*$, and the last inequality follows from the triangular inequality. Hence, the conclusion (21) holds as desired. □

5.1 Proof of the main results in Section 3.1

In this subsection, we prove Theorem 1.

Proof of Theorem 1. The proof of (5) can be found in [22, Theorem 10.5]. Next, we prove (6). Let $Q^T M = U_Q \Sigma V^T$ be the reduced SVD of $Q^T M$ and U_Q, V, and Q have orthogonal columns, and Σ is diagonal. Denote $U = QU_Q$. Since Q and U_Q have orthogonal columns, we obtain that U also has orthogonal columns; thus, $QQ^T M = U\Sigma V^T$ is a reduced SVD of $QQ^T M$. Therefore, we have $\mathrm{msgn}(QQ^T M) = UV^T = QU_Q V^T = Q\mathrm{msgn}(Q^T M)$. □

5.2 Proof of the main results in Section 3.2

In this subsection, we provide proofs of Theorems 2 and 3.

Proof of Theorem 2. Recall from Algorithm 2 that $M_O^k = \mathrm{msgn}(M_Q^k)$ for all $k \geq 0$. Using this and Lemma 4 with $(X, X^+, M, \eta) = (X^k, X^{k+1}, M_Q^k, \eta_k)$, we obtain that for all $k \geq 0$,

$$f(X^{k+1}) \leq f(X^k) - \eta_k\|\nabla f(X^k)\|_* + 2\eta_k\|\nabla f(X^k) - M_Q^k\|_* + \frac{L_*\eta_k^2}{2}. \tag{22}$$

Summing (22) over $k = 0, \ldots, K-1$ and using $f(X^K) \geq f_{\mathrm{low}}$, we obtain that for all $K \geq 1$,

$$\begin{aligned}
f_{\mathrm{low}} \leq f(X^K) &\overset{(22)}{\leq} f(X^0) - \sum_{k=0}^{K-1} \eta_k\|\nabla f(X^k)\|_* + \sum_{k=0}^{K-1}\Big(2\eta_k\|\nabla f(X^k) - M_Q^k\|_* + \frac{L_*\eta_k^2}{2}\Big) \\
&\leq f(X^0) - \eta_{K-1}\sum_{k=0}^{K-1}\|\nabla f(X^k)\|_* + \sum_{k=0}^{K-1}\Big(2\eta_k\|\nabla f(X^k) - M_Q^k\|_* + \frac{L_*\eta_k^2}{2}\Big),
\end{aligned} \tag{23}$$

where the last inequality is due to the fact that $\{\eta_k\}$ is nonincreasing. Rearranging this inequality and using (7), we obtain that for all $K \geq 3$,

$$\frac{1}{K}\sum_{k=0}^{K-1}\|\nabla f(X^k)\|_* \overset{(23)}{\leq} \frac{f(X^0) - f_{\mathrm{low}}}{K\eta_{K-1}} + \frac{1}{K\eta_{K-1}}\sum_{k=0}^{K-1}\Big(2\eta_k\|\nabla f(X^k) - M_Q^k\|_* + \frac{L_*\eta_k^2}{2}\Big)$$

$$\overset{(7)}{=} \frac{f(X^0) - f_{\text{low}}}{K^{1/2}} + \frac{1}{K^{1/2}} \sum_{k=0}^{K-1} \Big(\frac{2\|\nabla f(X^k) - M_Q^k\|_*}{(k+1)^{1/2}} + \frac{L_*}{2(k+1)} \Big)$$
$$\leq \frac{f(X^0) - f_{\text{low}} + L_* \ln K}{K^{1/2}} + \frac{2}{K^{1/2}} \sum_{k=0}^{K-1} \frac{\|\nabla f(X^k) - M_Q^k\|_*}{(k+1)^{1/2}},$$

where the last inequality follows from $\sum_{k=0}^{K-1} 1/(k+1) \leq \ln(2K+1) \leq 2\ln K$ for all $K \geq 3$ due to (20). Hence, the conclusion (8) holds as desired. □

Proof of Theorem 3. Using (9), (10), (11), and the same arguments as for proving (8), we obtain that for all $K \geq 3$,

$$\begin{aligned}
\min_{0\leq k\leq K-1}\{\|\nabla f(X^k)\|\} &\overset{(8)(9)}{\leq} \frac{(f(X^0) - f_{\text{low}} + L_*)\ln K}{K^{1/2}} + \frac{2}{K^{1/2}} \sum_{k=0}^{K-1} \frac{\delta_k}{(k+1)^{1/2}} \\
&\overset{(11)}{=} \frac{(f(X^0) - f_{\text{low}} + L_*)\ln K}{K^{1/2}} + \frac{2}{K^{1/2}} \sum_{k=0}^{K-1} \frac{1}{k+1} \\
&\leq \frac{(f(X^0) - f_{\text{low}} + L_* + 4)\ln K}{K^{1/2}} \overset{(10)}{=} \frac{U_{\text{gd}} \ln K}{K^{1/2}},
\end{aligned} \tag{24}$$

where the last inequality follows from $\sum_{k=0}^{K-1} 1/(k+1) \leq \ln(2K+1) \leq 2\ln K$ for all $K \geq 3$ due to (20). In addition, by Lemma 3 with $(\beta, u, v) = (1/2, \epsilon/(4U_{\text{gd}}), K)$, one can see that

$$K^{-1/2} \ln K \leq \frac{\epsilon}{U_{\text{gd}}} \qquad \forall K \geq \Big(\frac{4U_{\text{gd}}}{\epsilon} \ln\Big(\frac{4U_{\text{gd}}}{\epsilon}\Big)\Big)^2,$$

which along with (24) implies that Theorem 3 holds.

□

5.3 Proof of the main results in Section 3.3

In this subsection, we begin by establishing some technical lemmas and use them to prove Theorem 4. For convenience, we define a sequence of potentials for Algorithm 4 as follows:

$$\mathcal{P}_k := f(X^k) + p_k \|\nabla f(X^k) - M^k\|_F^\alpha \qquad \forall k \geq 0, \tag{25}$$

where the sequence $\{(X^k, M^k)\}$ is generated by Algorithm 4, and $\{p_k\}$ is a sequence of positive scalars that will be specified separately later. The following lemma presents a recurrence relation for the estimation error of the gradient estimators $\{M^k\}$ generated by Algorithm 4.

Lemma 5. *Suppose that Assumptions 1 and 2 hold. Let $\{(X^k, M^k)\}$ be generated by Algorithm 4 with input parameters $\{(\eta_k, \theta_k)\}$. Then we have*

$$\mathbb{E}_{\xi^{k+1}}[\|M^{k+1} - \nabla f(X^{k+1})\|_F^\alpha] \leq (1-\theta_k)\|M^k - \nabla f(X^k)\|_F^\alpha + 3L_*^\alpha \theta_k^{1-\alpha}\eta_k^\alpha + 2\sigma^\alpha\theta_k^\alpha \qquad \forall k \geq 0, \tag{26}$$

where L_* is given in Assumption 1, and σ and α are given in Assumption 2.

Proof. Fix any $k \geq 0$. It follows from (12) that

$$\begin{aligned}
&M^{k+1} - \nabla f(X^{k+1}) \overset{(12)}{=} (1-\theta_k)M^k + \theta_k G(X^{k+1};\xi^{k+1}) - \nabla f(X^{k+1}) \\
&= (1-\theta_k)(M^k - \nabla f(X^k)) + (1-\theta_k)(\nabla f(X^k) - \nabla f(X^{k+1})) + \theta_k(G(X^{k+1};\xi^{k+1}) - \nabla f(X^{k+1})).
\end{aligned} \tag{27}$$

Notice from (13) that $M_O^k = \mathrm{msgn}(M_Q^k)$, which along with (14) implies that $\|X^{k+1}-X^k\| = \eta_k\|\mathrm{msgn}(M_Q^k)\| \leq \eta_k$. Also, it follows from Assumption 1 that $\mathbb{E}_{\xi^{k+1}}[G(X^{k+1};\xi^{k+1})-\nabla f(X^{k+1})] = 0$, $\mathbb{E}_{\xi^{k+1}}[\|G(X^{k+1};\xi^{k+1})-\nabla f(X^{k+1})\|_F^\alpha] \leq \sigma^\alpha$, and $\|\nabla f(X^k) - \nabla f(X^{k+1})\|_F \leq \|\nabla f(X^k) - \nabla f(X^{k+1})\|_* \leq L_*\eta_k$. Using these, (18), (19), and (27), we obtain that for all $c > 0$,

$$
\begin{aligned}
&\mathbb{E}_{\xi^{k+1}}[\|M^{k+1} - \nabla f(X^{k+1})\|_F^\alpha] \\
&\overset{(27)}{=} \mathbb{E}_{\xi^{k+1}}[\|(1-\theta_k)(M^k - \nabla f(X^k)) + (1-\theta_k)(\nabla f(X^k) - \nabla f(X^{k+1})) + \theta_k(G(X^{k+1};\xi^{k+1}) - \nabla f(X^{k+1}))\|_F^\alpha] \\
&\overset{(18)}{\leq} \|(1-\theta_k)(M^k - \nabla f(X^k)) + (1-\theta_k)(\nabla f(X^k) - \nabla f(X^{k+1}))\|_F^\alpha + 2\mathbb{E}_{\xi^{k+1}}[\|\theta_k(G(X^{k+1};\xi^{k+1}) - \nabla f(X^{k+1}))\|_F^\alpha] \\
&\overset{(19)}{\leq} (1+c)(1-\theta_k)^\alpha\|M^k - \nabla f(X^k)\|_F^\alpha + (2 + (\alpha-1)^{\alpha-1}c^{1-\alpha})(1-\theta_k)^\alpha\|\nabla f(X^k) - \nabla f(X^{k+1})\|_F^\alpha + 2\sigma^\alpha\theta_k^\alpha \\
&\leq (1+c)(1-\theta_k)^\alpha\|M^k - \nabla f(X^k)\|_F^\alpha + L_*^\alpha(2 + (\alpha-1)^{\alpha-1}c^{1-\alpha})(1-\theta_k)^\alpha\eta_k^\alpha + 2\sigma^\alpha\theta_k^\alpha, \qquad (28)
\end{aligned}
$$

where the first inequality is due to (18) and $\mathbb{E}_{\xi^{k+1}}[G(X^{k+1};\xi^{k+1})-\nabla f(X^{k+1})] = 0$, the second inequality is due to (19) and $\mathbb{E}_{\xi^{k+1}}[\|G(X^{k+1};\xi^{k+1}) - \nabla f(X^{k+1})\|_F^\alpha] \leq \sigma^\alpha$, and the last inequality is due to $\|\nabla f(X^k) - \nabla f(X^{k+1})\|_F \leq L_*\eta_k$.

When $\theta_k = 1$, (26) clearly holds. For $\theta_k \in (0,1)$, letting $c = (1-\theta_k)^{1-\alpha} - 1$ in (28), and using the fact that $\alpha \in (1,2]$, we have

$$
\begin{aligned}
c^{1-\alpha} &= ((1-\theta_k)^{1-\alpha} - 1)^{1-\alpha} = \Big(\frac{1}{(1-\theta_k)^{\alpha-1}} - 1\Big)^{1-\alpha} \leq \Big(\frac{1}{1-(\alpha-1)\theta_k} - 1\Big)^{1-\alpha} \\
&= \Big(\frac{1-(\alpha-1)\theta_k}{(\alpha-1)\theta_k}\Big)^{\alpha-1} \leq ((\alpha-1)\theta_k)^{1-\alpha},
\end{aligned}
$$

where the first inequality follows from $(1-\tau)^\beta \leq 1 - \beta\tau$ for all $\tau \in (-\infty, 1)$ and $\beta \in [0,1]$. Combining this inequality with (28), one has

$$
\mathbb{E}_{\xi^{k+1}}[\|M^{k+1} - \nabla f(X^{k+1})\|_F^\alpha] \leq (1-\theta_k)\|M^k - \nabla f(X^k)\|_F^\alpha + L_*^\alpha(2 + \theta_k^{1-\alpha})(1-\theta_k)^\alpha\eta_k^\alpha + 2\sigma^\alpha\theta_k^\alpha,
$$

which along with $\theta_k \in (0,1]$ and $\alpha \in (1,2]$ implies that (26) holds as desired. □

The following lemma establishes a descent property for the potential sequence $\{\mathcal{P}_k\}$ defined below.

Lemma 6. *Suppose that Assumptions 1 and 2 hold. Let $\{(X^k, M^k)\}$ be generated by Algorithm 4 with input parameters $\{(\eta_k, \theta_k, \delta_k)\}$, L_* be given in Assumption 1, σ and α be given in Assumption 2, and $\varrho := \min\{m, n\}$, and let $\{\mathcal{P}_k\}$ be defined in (25) for $\{(X^k, M^k)\}$ and any nonincreasing positive sequence $\{p_k\}$. Then, it holds that*

$$
\begin{aligned}
\mathbb{E}_{\xi^{k+1}}[\mathcal{P}_{k+1}] \leq \mathcal{P}_k &- \eta_k\|\nabla f(X^k)\|_* + 2\eta_k\delta_k + \frac{L_*\eta_k^2}{2} \\
&+ \frac{(\alpha-1)(2\varrho^{1/2}\eta_k)^{\alpha/(\alpha-1)}}{\alpha^{\alpha/(\alpha-1)}(\theta_k p_k)^{1/(\alpha-1)}} + 3L_*^\alpha\theta_k^{1-\alpha}\eta_k^\alpha p_k + 2\sigma^\alpha\theta_k^\alpha p_k \qquad \forall k \geq 0. \qquad (29)
\end{aligned}
$$

Proof. Fix any $k \geq 0$. Recall from Algorithm 4 that $M_O^k = \mathrm{msgn}(M_Q^k)$ and $\|M^k - M_Q^k\|_* \leq \delta_k$. Using these and (21) with $(X^+, X, M, \eta) = (X^{k+1}, X^k, M_Q^k, \eta_k)$, we obtain that

$$
\begin{aligned}
f(X^{k+1}) &\leq f(X^k) - \eta_k\|\nabla f(X^k)\|_* + 2\eta_k\|\nabla f(X^k) - M_Q^k\|_* + \frac{L_*\eta_k^2}{2} \\
&\leq f(X^k) - \eta_k\|\nabla f(X^k)\|_* + 2\eta_k\|\nabla f(X^k) - M^k\|_* + 2\eta_k\delta_k + \frac{L_*\eta_k^2}{2}. \qquad (30)
\end{aligned}
$$

Combining this with (25) and (26), we obtain that

$$\begin{aligned}
\mathbb{E}_{\xi^{k+1}}[\mathcal{P}_{k+1}] &\overset{(25)}{=} \mathbb{E}_{\xi^{k+1}}[f(X^{k+1}) + p_{k+1}\|\nabla f(X^{k+1}) - M^{k+1}\|_F^\alpha] \\
&\overset{(26)(30)}{\leq} f(X^k) - \eta_k\|\nabla f(X^k)\|_* + 2\eta_k\|\nabla f(X^k) - M^k\|_* + 2\eta_k\delta_k + \frac{L_*\eta_k^2}{2} \\
&\quad + (1-\theta_k)p_{k+1}\|M^k - \nabla f(X^k)\|_F^\alpha + 3L_*^\alpha\theta_k^{1-\alpha}\eta_k^\alpha p_{k+1} + 2\sigma^\alpha\theta_k^\alpha p_{k+1} \\
&\leq f(X^k) - \eta_k\|\nabla f(X^k)\|_* + 2\eta_k\varrho^{1/2}\|\nabla f(X^k) - M^k\|_F + 2\eta_k\delta_k + \frac{L_*\eta_k^2}{2} \\
&\quad + (1-\theta_k)p_k\|M^k - \nabla f(X^k)\|_F^\alpha + 3L_*^\alpha\theta_k^{1-\alpha}\eta_k^\alpha p_k + 2\sigma^\alpha\theta_k^\alpha p_k, \qquad (31)
\end{aligned}$$

where the second inequality follows from $\|U\|_* \leq \varrho^{1/2}\|U\|_F$ for all $U \in \mathbb{R}^{m\times n}$, and the fact that $\{p_k\}$ is nonincreasing. In addition, letting $\alpha' = \alpha/(\alpha-1)$ and using the Young's inequality, we obtain that

$$\begin{aligned}
2\eta_k\varrho^{1/2}\|\nabla f(X^k) - M^k\|_F &\leq \frac{\big((\alpha\theta_k p_k)^{1/\alpha}\|\nabla f(X^k) - M^k\|_F\big)^\alpha}{\alpha} + \frac{(2\varrho^{1/2}\eta_k/(\alpha\theta_k p_k)^{1/\alpha})^{\alpha'}}{\alpha'} \\
&= \theta_k p_k\|\nabla f(X^k) - M^k\|_F^\alpha + \frac{(\alpha-1)(2\varrho^{1/2}\eta_k)^{\alpha/(\alpha-1)}}{\alpha^{\alpha/(\alpha-1)}(\theta_k p_k)^{1/(\alpha-1)}}.
\end{aligned}$$

This along with (31) implies that

$$\begin{aligned}
\mathbb{E}_{\xi^{k+1}}[\mathcal{P}_{k+1}] \leq f(X^k) + p_k\|\nabla f(X^k) - M^k\|_F^\alpha - \eta_k\|\nabla f(X^k)\|_* + 2\eta_k\delta_k + \frac{L_*\eta_k^2}{2} \\
+ \frac{(\alpha-1)(2\varrho^{1/2}\eta_k)^{\alpha/(\alpha-1)}}{\alpha^{\alpha/(\alpha-1)}(\theta_k p_k)^{1/(\alpha-1)}} + 3L_*^\alpha\theta_k^{1-\alpha}\eta_k^\alpha p_k + 2\sigma^\alpha\theta_k^\alpha p_k.
\end{aligned}$$

The conclusion (29) then follows from this and (25). $\square$

We are now ready to prove Theorem 4.

Proof of Theorem 4. Let $\{(X^k, M^k)\}$ be generated by Algorithm 4 with $\{(\eta_k, \theta_k, \delta_k)\}$ given in (16), and $\{\mathcal{P}_k\}$ be defined in (25) with such $\{(X^k, M^k)\}$ and the following $\{p_k\}$:

$$p_k = (k+1)^{(\alpha^2-3\alpha+2)/(3\alpha-2)} \qquad \forall k \geq 0. \qquad (32)$$

Since $\alpha \in (1, 2]$, one can see that $\{p_k\}$ is nonincreasing. Also, observe from (16) that $\{\eta_k\} \subset (0, 1]$ and $\{\theta_k\} \subset (0, 1]$. Hence, $\{(\eta_k, \theta_k, p_k)\}$ satisfies the assumptions in Lemma 6 and Algorithm 4. In addition, by (25) and (32), one has that

$$\mathbb{E}[\mathcal{P}_0] = f(X^0) + p_0\mathbb{E}[\|M^0 - \nabla f(X^0)\|_F^\alpha] = f(X^0) + \mathbb{E}[\|G(X^0;\xi^0) - \nabla f(X^0)\|_F^\alpha] \leq f(X^0) + \sigma^\alpha, \quad (33)$$
$$\mathbb{E}[\mathcal{P}_K] = \mathbb{E}[f(X^K) + p_K\|M^K - \nabla f(X^K)\|_F^\alpha] \geq \mathbb{E}[f(X^K)] \geq f_{\text{low}}. \quad (34)$$

Taking expectation on both sides of (29) with respect to $\{\xi^i\}_{i=0}^{k+1}$, we have

$$\begin{aligned}
\mathbb{E}[\mathcal{P}_{k+1}] &\leq \mathbb{E}[\mathcal{P}_k] - \eta_k\mathbb{E}[\|\nabla f(X^k)\|_*] \\
&\quad + 2\eta_k\delta_k + \frac{L_*\eta_k^2}{2} + \frac{(\alpha-1)(2\varrho^{1/2}\eta_k)^{\alpha/(\alpha-1)}}{\alpha^{\alpha/(\alpha-1)}(\theta_k p_k)^{1/(\alpha-1)}} + 3L_*^\alpha\theta_k^{1-\alpha}\eta_k^\alpha p_k + 2\sigma^\alpha\theta_k^\alpha p_k \qquad \forall k \geq 0.
\end{aligned}$$

Summing up this inequality over $k = 0, \ldots, K-1$, and using (33) and (34), we obtain that for all $K \geq 1$,

$$f_{\text{low}} \overset{(34)}{\leq} \mathbb{E}[\mathcal{P}_K]$$

$$\leq \mathbb{E}[\mathcal{P}_0] - \sum_{k=0}^{K-1} \eta_k \mathbb{E}[\|\nabla f(X^k)\|_*]$$
$$+ \sum_{k=0}^{K-1} \Big(2\eta_k\delta_k + \frac{L_*\eta_k^2}{2} + \frac{(\alpha-1)(2\varrho^{1/2}\eta_k)^{\alpha/(\alpha-1)}}{\alpha^{\alpha/(\alpha-1)}(\theta_k p_k)^{1/(\alpha-1)}} + 3L_*^\alpha \theta_k^{1-\alpha}\eta_k^\alpha p_k + 2\sigma^\alpha\theta_k^\alpha p_k\Big)$$
$$\overset{(33)}{\leq} f(X^0) + \sigma^\alpha - \eta_{K-1}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(X^k)\|_*]$$
$$+ \sum_{k=0}^{K-1} \Big(2\eta_k\delta_k + \frac{L_*\eta_k^2}{2} + \frac{(\alpha-1)(2\varrho^{1/2}\eta_k)^{\alpha/(\alpha-1)}}{\alpha^{\alpha/(\alpha-1)}(\theta_k p_k)^{1/(\alpha-1)}} + 3L_*^\alpha \theta_k^{1-\alpha}\eta_k^\alpha p_k + 2\sigma^\alpha\theta_k^\alpha p_k\Big), \tag{35}$$

where the last inequality follows from (33) and the fact that $\{\eta_k\}$ is nonincreasing. Rearranging the terms in (35), and using (15), (16), and (32), we obtain that for all $K \geq 3$,

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(X^k)\|_*] \overset{(35)}{\leq} \frac{f(X^0) - f_{\text{low}} + \sigma^\alpha}{K\eta_{K-1}}$$
$$+ \frac{1}{K\eta_{K-1}}\sum_{k=0}^{K-1}\Big(2\eta_k\delta_k + \frac{L_*\eta_k^2}{2} + \frac{(\alpha-1)(2\varrho^{1/2}/\alpha)^{\alpha/(\alpha-1)}\eta_k^{\alpha/(\alpha-1)}}{(\theta_k p_k)^{1/(\alpha-1)}} + 3L_*^\alpha\theta_k^{1-\alpha}\eta_k^\alpha p_k + 2\sigma^\alpha\theta_k^\alpha p_k\Big)$$
$$\overset{(16)(32)}{=} \frac{f(X^0) - f_{\text{low}} + \sigma^\alpha}{K^{(\alpha-1)/(3\alpha-2)}}$$
$$+ \frac{1}{K^{(\alpha-1)/(3\alpha-2)}}\sum_{k=0}^{K-1}\Big(\frac{L_*}{2(k+1)^{2(2\alpha-1)/(3\alpha-2)}} + \frac{2 + (\alpha-1)(2\varrho^{1/2}/\alpha)^{\alpha/(\alpha-1)} + 3L_*^\alpha + 2\sigma^\alpha}{k+1}\Big)$$
$$\leq \frac{(f(X^0) - f_{\text{low}} + \sigma^\alpha + 2L_* + 4 + 2(\alpha-1)(2\varrho^{1/2}/\alpha)^{\alpha/(\alpha-1)} + 6L_*^\alpha + 4\sigma^\alpha)\ln K}{K^{(\alpha-1)/(3\alpha-2)}} \overset{(15)}{=} \frac{U_{\text{mn}}\ln K}{K^{(\alpha-1)/(3\alpha-2)}},$$

where the second inequality follows from $\sum_{k=0}^{K-1} 1/(k+1) \leq 2\ln K$ due to (20) and $K \geq 3$, and $\sum_{k=0}^{K-1} 1/(k+1)^{2(2\alpha-1)/(3\alpha-2)} \leq (3\alpha-2)2^{\alpha/(3\alpha-2)}/\alpha < 4$ due to (20) and $(3\alpha-2)/\alpha \in (1,2]$. Recall that ι_K is uniformly selected from $\{0,\ldots,K-1\}$. It then follows from this and the above inequality that

$$\mathbb{E}[\|\nabla f(X^{\iota_K})\|_*] = \frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}[\|\nabla f(X^k)\|_*] \leq \frac{U_{\text{mn}}\ln K}{K^{(\alpha-1)/(3\alpha-2)}} \qquad \forall K \geq 3. \tag{36}$$

In addition, by Lemma 3 with $(\beta, u, v) = ((\alpha-1)/(3\alpha-2), (\alpha-1)\epsilon/(2(3\alpha-2)U_{\text{mn}}), K)$, one can see that

$$K^{-(\alpha-1)/(3\alpha-2)}\ln K \leq \frac{\epsilon}{U_{\text{mn}}} \qquad \forall K \geq \Big(\frac{2(3\alpha-2)U_{\text{mn}}}{(\alpha-1)\epsilon}\ln\Big(\frac{2(3\alpha-2)U_{\text{mn}}}{(\alpha-1)\epsilon}\Big)\Big)^{(3\alpha-2)/(\alpha-1)},$$

which together with (36) implies that Theorem 4 holds. □

6 Concluding remarks

In this paper, we introduce a new low-rank orthogonalization approach as a lightweight substitute for the popular orthogonalization approach by Newton-Schulz iterations. Based on this low-rank orthogonalization, we propose low-rank Muon and illustrate its advantages over existing training methods through experiments on GPT-2 and LLaMA pretraining. Theoretically, we also establish complexity bounds for the proposed low-rank Muon under heavy-tailed noise.

A key ingredient of our low-rank orthogonalization is the construction of a column-orthogonal matrix Q, inspired by low-rank matrix approximation techniques. A valuable future research direction is to explore alternative strategies for constructing Q. Some such alternatives are presented in Appendix A.

References

[1] K. Ahn and B. Xu. Dion: A communication-efficient optimizer for large models. *arXiv preprint arXiv:2504.05295*, 2025.

[2] K. Ahn, B. Xu, N. Abreu, and J. Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

[3] E. AI, I. Shah, A. M. Polloreno, K. Stratos, P. Monk, A. Chaluvaraju, A. Hojel, A. Ma, A. Thomas, A. Tanwer, D. J. Shah, K. Nguyen, K. Smith, M. Callahan, M. Pust, M. Parmar, P. Rushton, P. Mazarakis, R. Kapila, S. Srivastava, S. Singla, T. Romanski, Y. Vanjani, and A. Vaswani. Practical efficiency of Muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

[4] K. An, Y. Liu, R. Pan, S. Ma, D. Goldfarb, and T. Zhang. ASGO: Adaptive structured gradient optimization. *arXiv preprint arXiv:2503.20762*, Mar. 2025. Preprint.

[5] R. Anil, V. Gupta, T. Koren, K. Regan, and Y. Singer. Scalable second order optimization for deep learning. *arXiv preprint arXiv:2002.09018*, 2020.

[6] J. Bernstein and L. Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

[7] F. Bian, J.-F. Cai, and R. Zhang. A preconditioned Riemannian gradient descent algorithm for low-rank matrix recovery. *SIAM Journal on Matrix Analysis and Applications*, 45(4):2075–2103, 2024.

[8] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang. On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258, 2021.

[9] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, pages 177–186. Springer, 2010.

[10] D. Carlson, V. Cevher, and L. Carin. Stochastic spectral descent for restricted Boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 111–119, 2015.

[11] D. Carlson, Y.-P. Hsieh, E. Collins, L. Carin, and V. Cevher. Stochastic spectral descent for discrete graphical models. *IEEE Journal of Selected Topics in Signal Processing*, 10(2):296–311, 2015.

[12] D. E. Carlson, E. Collins, Y.-P. Hsieh, L. Carin, and V. Cevher. Preconditioned spectral descent for deep learning. In *Advances in neural information processing systems*, volume 28, 2015.

[13] Y. Carmon, J. C. Duchi, O. Hinder, and A. Sidford. Lower bounds for finding stationary points I. *Mathematical Programming*, 184(1):71–120, 2020.

[14] F. L. Cesista. Muon and a selective survey on steepest descent in Riemannian and non-Riemannian manifolds, 2025.

[15] L. Chen, J. Li, and Q. Liu. Muon optimizes under spectral norm constraints. *arXiv preprint arXiv:2506.15054*, 2025.

[16] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[17] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183, 2006.

[18] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

[19] S. S. S. Duvvuri, F. Devvrit, R. Anil, C. Hsieh, and I. S. Dhillon. Combining axes preconditioners through Kronecker approximation for deep learning. In *International Conference on Learning Representations*, 2024.

[20] A. Glentis, J. Li, A. Han, and M. Hong. A minimalist optimizer design for LLM pretraining. *arXiv preprint arXiv:2506.16659*, 2025.

[21] V. Gupta, T. Koren, and Y. Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850, 2018.

[22] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[23] Y. Hao, Y. Cao, and L. Mou. Flora: Low-rank adapters are secretly gradient compressors. In *International Conference on Machine Learning*, 2024.

[24] C. He, Z. Lu, D. Sun, and Z. Deng. Complexity of normalized stochastic first-order methods with momentum under heavy-tailed noise. *arXiv preprint arXiv:2506.11214*, 2025.

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

[26] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, et al. Lora: Low-rank adaptation of large language models. *International Conference on Learning Representations*, 1(2):3, 2022.

[27] K. Jordan, J. Bernstein, B. Rappazzo, @fernbear.bsky.social, B. Vlado, Y. Jiacheng, F. Cesista, B. Koszarsky, and @Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline, 2024.

[28] K. Jordan, Y. Jin, V. Boza, Y. Jiacheng, F. Cecista, L. Newhouse, and J. Bernstein. Muon: An optimizer for hidden layers in neural networks. *URL https://kellerjordan. github. io/posts/muon.*

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[30] D. Kovalev. Understanding gradient orthogonalization for deep learning via non-Euclidean trust-region optimization. *arXiv preprint arXiv:2503.12645*, 2025.

[31] T. T.-K. Lau, Q. Long, and W. Su. PolarGrad: A class of matrix-gradient optimizers from a unifying preconditioning perspective. *arXiv preprint arXiv:2505.21799*, 2025.

[32] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[33] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[34] J. Li and M. Hong. A note on the convergence of Muon and further. *arXiv e-prints*, pages arXiv–2502, 2025.

[35] J. Liu, J. Su, X. Yao, Z. Jiang, G. Lai, Y. Du, Y. Qin, W. Xu, E. Lu, J. Yan, Y. Chen, H. Zheng, Y. Liu, S. Liu, B. Yin, W. He, H. Zhu, Y. Wang, J. Wang, M. Dong, Z. Zhang, Y. Kang, H. Zhang, X. Xu, Y. Zhang, Y. Wu, X. Zhou, and Z. Yang. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

[36] L. Liu, Z. Xu, Z. Zhang, H. Kang, Z. Li, C. Liang, W. Chen, and T. Zhao. COSMOS: A hybrid adaptive optimizer for memory-efficient training of LLMs. *arXiv preprint arXiv:2502.17410*, 2025.

[37] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[38] C. Ma, W. Gong, M. Scetbon, and E. Meeds. SWAN: Preprocessing SGD enables Adam-level performance on LLM training with significant memory reduction. *arXiv e-prints*, pages arXiv–2412, 2024.

[39] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen, and S. Arora. Fine-tuning language models with just forward passes. In *Advances in Neural Information Processing Systems*, volume 36, pages 53038–53075, 2023.

[40] P.-G. Martinsson. Randomized methods for matrix computations. *arXiv preprint arXiv:1607.01649*, 2016.

[41] T. Pethick, W. Xie, K. Antonakopoulos, Z. Zhu, A. Silveti-Falls, and V. Cevher. Training deep learning models with norm-constrained LMOs. *arXiv preprint arXiv:2502.07529*, 2025.

[42] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[43] A. Riabinin, E. Shulgin, K. Gruntkowska, and P. Richtárik. Gluon: Making Muon & Scion great again! (bridging theory and practice of LMO-based optimizers for LLMs). *arXiv preprint arXiv:2505.13416*, 2025.

[44] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100–1124, 2010.

[45] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[46] N. Sato, H. Naganuma, and H. Iiduka. Analysis of Muon's convergence and critical batch size. *arXiv preprint arXiv:2507.01598*, 2025.

[47] M.-E. Sfyraki and J.-K. Wang. Lions and Muons: Optimization via stochastic Frank-Wolfe. *arXiv preprint arXiv:2506.04192*, 2025.

[48] W. Shen, R. Huang, M. Huang, C. Shen, and J. Zhang. On the convergence analysis of Muon. *arXiv preprint arXiv:2505.23737*, 2025.

[49] C. Si, D. Zhang, and W. Shen. AdaMuon: Adaptive Muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025.

[50] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[51] M. Tuddenham, A. Prügel-Bennett, and J. Hare. Orthogonalising gradients to speed up neural network optimisation. *arXiv preprint arXiv:2202.07052*, 2022.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[53] N. Vyas, D. Morwani, R. Zhao, I. Shapira, D. Brandfonbrener, L. Janson, and S. M. Kakade. SOAP: Improving and stabilizing Shampoo using Adam for language modeling. In *International Conference on Learning Representations*, 2025.

[54] S. Xie, T. Wang, S. J. Reddi, S. Kumar, and Z. Li. Structured preconditioners in adaptive optimization: A unified analysis. *arXiv preprint arXiv:2503.10537*, 2025.

[55] M. D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[56] J. Zhang, S. P. Karimireddy, A. Veit, S. Kim, S. Reddi, S. Kumar, and S. Sra. Why are adaptive methods good for attention models? In *Advances in Neural Information Processing Systems*, volume 33, pages 15383–15393, 2020.

[57] J. Zhao, Z. Zhang, B. Chen, Z. Wang, A. Anandkumar, and Y. Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *International Conference on Machine Learning*, volume 235, pages 61121–61143, 2024.

A Low-rank orthogonalization procedures

In this part, we introduce two new low-rank orthogonalization methods as alternatives to Algorithm 1.

We first describe a low-rank orthogonalization method based on column selection. This method is based on the low-rank matrix approximation techniques developed in [16, 17]. For any given $M \in \mathbb{R}^{m\times n}$, it first performs column subsampling on M to select r columns, with $r \ll \varrho$, based on the size of each column, to construct a matrix $C \in \mathbb{R}^{m\times r}$. Then, this method performs a QR decomposition on C to obtain a column-orthogonal matrix $Q \in \mathbb{R}^{m\times r}$, and computes $\mathrm{msgn}(Q^T M)$. Finally, it returns $M_O = Q\mathrm{msgn}(Q^T M)$ as a low-rank approximation for $\mathrm{msgn}(M)$. Following the same arguments for proving (6), M_O is the matrix sign of QQ^TM, which is a low-rank approximation of M. Details of this method are provided in Algorithm 5.

The following theorem, adapted from [17, Theorem 4], provides the approximation error of QQ^TM.

Theorem 5. *Consider Algorithm 5 with inputs $M \in \mathbb{R}^{m\times n}$ and $r \in \mathbb{Z}_+ \cap [1, \varrho]$, where $\varrho := \min\{m, n\}$. Let $Q \in \mathbb{R}^{m\times r}$ be generated by Algorithm 5. Then, it holds that*

$$\mathbb{E}[\|(I - QQ^T)M\|_F^2] \leq \|M - [M]_r\|_F^2 + 2\|M\|_F^2.$$

Algorithm 5 A low-rank orthogonalization method based on column selection

```
**Input:** matrix $M \in \mathbb{R}^{m\times n}$, rank trial $r \in \mathbb{Z}_+ \cap [1, \varrho]$.
**Output:** approximate matrix sign $M_O \in \mathbb{R}^{m\times n}$.
Compute sampling probabilities $\{p_i\}_{i=1}^n$ as $p_i = \|M_{(i)}\|^2/\|M\|_F^2$ for $i = 1, \ldots, n$.
for $t = 1, 2, \ldots, r$ do
    Choose $i_t \in \{1, \ldots, n\}$ with probability $\mathbb{P}(i_t = i) = p_i$ for $i = 1, \ldots, n$.
    Set $C_{(i)} = M_{(i_t)}/\sqrt{rp_{i_t}}$.
end for
Perform a QR decomposition on $C$ to get a column-orthogonal Q factor $Q \in \mathbb{R}^{m\times r}$.
Return $M_O = Q\mathrm{msgn}(Q^TM)$.
```

We next propose a low-rank orthogonalization method based on power iterations. This method is based on the low-rank matrix approximation techniques developed in [22, 44]. Similar to Algorithm 1, this method also uses a Gaussian random matrix $G \in \mathbb{R}^{m\times r}$ with $r \ll \varrho$, and performs a QR decomposition on $(MM^T)^qMG$ for some $q \geq 0$ to obtain a column-orthogonal matrix $Q \in \mathbb{R}^{m\times r}$. The method then computes $\mathrm{msgn}(Q^TM) \in \mathbb{R}^{r\times n}$ and returns $M_O = Q\mathrm{msgn}(Q^TM)$ as a low-rank approximation of $\mathrm{msgn}(M)$. Following the same arguments used to prove (6), M_O represents the matrix sign of QQ^TM, which is a low-rank approximation of M. Details of this method are provided in Algorithm 6, and it can be seen that the method reduces to Algorithm 1 when $q = 0$.

Algorithm 6 A low-rank orthogonalization procedure based on power iterations

```
**Input:** matrix $M \in \mathbb{R}^{m\times n}$, rank trial $r \in \mathbb{Z}_+ \cap [1, \varrho]$, exponent $q \in \mathbb{Z}_+$.
**Output:** approximate matrix sign $M_O \in \mathbb{R}^{m\times n}$.
Generate a Gaussian random matrix $G \in \mathbb{R}^{n\times r}$.
Perform a QR decomposition on $(MM^T)^qMG$ to get a column-orthogonal Q factor $Q \in \mathbb{R}^{m\times r}$.
Return $M_O = Q\mathrm{msgn}(Q^TM)$.
```

The following theorem, adapted from [40, Section 8.2], provides the approximation error of QQ^TM.

Theorem 6. *Consider Algorithm 6 with inputs $M \in \mathbb{R}^{m\times n}$, $r \in \mathbb{Z}_+ \cap [1, \varrho]$, and $q \in \mathbb{Z}_+$, where $\varrho := \min\{m, n\}$. Let $Q \in \mathbb{R}^{m\times r}$ be generated by Algorithm 5. Then, for any $r_* \leq r - 2$, it holds that*

$$\mathbb{E}[\|(I - QQ^T)M\|^2] \leq \left(1 + \sqrt{\frac{r_*}{r - r_* - 1}} + e\sqrt{\frac{r}{r - r_*}} \cdot \sqrt{\varrho - r_*}\right)^{1/(2q+1)} \sigma_{r_*+1},$$

where σ_{r_*+1} is the $(r_* + 1)$th largest singular value of M.

B Experimental details

In this part, we present the details for our experiments on pretraining GPT-2 and LLaMA models.

Table 3 presents the hyperparameter configurations for the GPT-2 and LLaMA models. We employ hyperparameters aligned with those used in [28, 57]. In addition, we set the maximum training sequence length to 65,536 tokens, the maximum validation sequence length to 262,144 tokens, and the validation dataset size to 10,485,760 tokens.

Table 3: Hyperparameter configurations for GPT-2 and LLaMA models.

# Parameters	Hidden Dimension	Intermediate Dimension	# Heads	# Layers
60M	512	1376	8	8
130M	768	2048	12	12
350M	1024	2736	16	24
1B	2048	5461	24	32

We present the algorithmic hyperparameter search grids for each method in Table 4, following a similar choice as in [28, 31]. The best numerical performance for each training method across the grids is presented in Section 4. In addition, we apply a warm-up schedule for the learning rate during the first 70% of the training iterations, followed by a cosine annealing schedule that reduces the learning rate to 10% of its initial value.

Table 4: Algorithmic hyperparameters. Here, 'NS' stands for Newton-Schulz.

Method	Hyperparameters	Values
SGDM	batch size	2
	learning rate	{1e-4, 1e-3, 1e-2}
	weight decay	{0, 1e-3, 1e-2}
AdamW	batch size	2
	learning rate	{1e-4, 1e-3, 1e-2}
	weight decay	{0, 1e-3, 1e-2}
Muon	batch size	2
	learning rate	{1e-2, 2.5e-2, 5e-2, 1e-3}
	# NS steps	{5, 10, 20}
	weight decay	0
Low rank Muon	batch size	2
	learning rate	{1e-2, 2.5e-2, 5e-2, 1e-3}
	rank	{100, 200}
	# NS steps	{5, 10, 20}
	weight decay	0