MALT: CURVATURE-AIDED MUON VIA LIGHTWEIGHT DIAGONAL PRECONDITIONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Muon has recently emerged as a promising alternative to AdamW for pretraining language models. By orthogonalizing momentum matrices using Newton–Schulz iterations, Muon mitigates the influence of gradient anisotropy by distributing the update more evenly across all singular directions, and enjoys an improved learning efficiency empirically. However, this operation does not explicitly account for the curvature of the loss landscape, and Muon may therefore still sensitive to curvature anisotropy. We bridge this gap by proposing **MALT**(**M**uon **A**ugmented by **L**ightweight **T**wo-sided preconditioning), which uses lightweight diagonal preconditioners to reduce the influence of curvature anisotropy to Muon. Specifically, MALT constructs two-sided diagonal preconditioners with low memory and computational overhead to capture a diagonal approximation of the curvature of the loss landscape. It then orthogonalizes the diagonally preconditioned momentum using Newton–Schulz iterations and maps the result back to define the update direction, while norm grafting controls the update magnitude. Convergence guarantees are provided for **MALT** in the stochastic non-convex setting. To improve its robustness to stochastic gradient noise, we further proposed **MALTER** (**MALT** with noise Adaptive st**E**psize **R**escaling). Experiments on GPT-2 Small, Medium, and Large pretraining show that the proposed methods outperform Muon while maintaining nearly the same memory footprint and wall-clock time.

1 INTRODUCTION

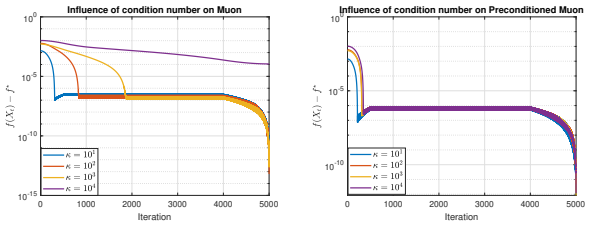
Recent progress in large language models (LLMs) has substantially accelerated the development of modern artificial intelligence (Bommasani et al., 2021; Zhao et al., 2023; Minaee et al., 2024). Models such as GPT (OpenAI, 2026), DeepSeek (DeepSeek-AI, 2025), LLaMA (Meta AI, 2025), and Gemini (Gemini Team, Google, 2025) have demonstrated strong capabilities across a wide range of tasks. However, as scaling laws (Kaplan et al., 2020) show that model performance is strongly correlated with model size, dataset size, and training compute, further improving their capability through conventional training routines can leads to substantially increased costs in training time, GPU memory (Rajbhandari et al., 2020), communication (Narayanan et al., 2021), and energy consumption (Patterson et al., 2021). As a result, developing efficient optimizers becomes a critical component to reduce these training costs for large models (Ranganath, 2026; Liu et al., 2025).

For decades, Adam (Kingma & Ba, 2015) and its variant AdamW (Loshchilov & Hutter, 2019) have remained the default optimizers for training deep neural networks and LLMs. These algorithms modify SGD by replacing the stochastic gradient with its exponential moving average to stabilize the update and rescaling it elementwise using its second-moment estimate. Although the precise role of adaptive rescaling in improving training efficiency is not yet fully understood, it is commonly interpreted as diagonal preconditioning that partially compensates for anisotropic curvature in the local loss landscape (Cohen et al., 2022; 2025) Despite their popularity, efforts in understanding and improving upon Adam-type optimizers never stopped. One criticism is that model weights are flattened as a single vector in the updates, whereas many neural network parameters are naturally organized as matrices or higher-order tensors. This mismatch has motivated the development of optimizers such as Shampoo (Gupta et al., 2018) and SOAP (Vyas et al., 2025), which construct preconditioners that exploit the matrix structure of the parameters. Muon (Jordan et al., 2024), by contrast, approaches the problem from a different perspective. Empirical evidence suggests that the gradients or momentum matrices arising during neural network training can exhibit highly unbalanced

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.

singular-value spectra. Consequently, an update may be dominated by a small number of leading singular directions (Lau et al., 2025; Huang et al., 2026), leaving the parameter space inefficiently updated in each iteration and potentially slow down convergence. Muon addresses this issue by using the Newton–Schulz iterations to approximately orthogonalize the momentum matrix, thereby flattening its singular spectrum and producing a nearly isotropic update direction.

As the readers may have already noticed, the above two lines of work are proposed to address different notions of anisotropy, with the former focusing on the curvature of the loss and the latter on the matrix-valued gradients (Lau et al., 2025). More specifically, curvature anisotropy describes how sharp or flat the loss landscape is across different directions, and is reflected by the eigenvalue spectra of the Hessian matrix of the loss (Nocedal & Wright, 2006; Ghorbani et al., 2019; Sagun et al., 2018). High curvature anisotropy is known to slow down the convergence of first order optimization methods, whose rate depends directly on the condition number of the Hessian matrix (Nocedal & Wright, 2006). Muon reduces anisotropy in each momentum matrix in its updates, but does not explicitly address curvature anisotropy of the loss. To understand how it will affect the training efficiency, we test the performance of Muon using a synthetic example whose curvature anisotropy can be controlled. Specifically, we consider solving a matrix quadratic regression problem $\min_{\boldsymbol{X}} f(\boldsymbol{X}) := \frac{1}{2}\|\boldsymbol{AXB} - \boldsymbol{C}\|_F^2$ under a warmup-stable-decay (WSD) learning rate scheduling, one that is widely adopted in training modern language models (Ibrahim et al., 2024; Wen et al., 2025). We observe that as the Hessian condition number of f increases, the convergence of Muon becomes substantially slower, as shown in Fig. 1 (a).



(a) Muon (b) Preconditioned Muon

Figure 1: Muon with and without preconditioning under the same WSD schedule. See Appendix C for details.

The sensitivity of Muon to curvature anisotropy motivates us to propose a new optimizer, MALT, to incorporate the curvature information into Muon by preconditioning. As demonstrated in Fig. 1 (b), preconditioned Muon is less sensitive to the condition number and converges substantially faster than Muon for ill-conditioned losses. Our contributions are summarized as follows:

- We revisit preconditioning for gradient methods and interpret it as steepest descent in a transformed space induced by the preconditioner, where the loss becomes more isotropic locally. Building on this viewpoint, we propose to integrate preconditioning and orthogonalization by first mapping the the gradient momentum to a preconditioned space built based on a local quadratic approximation of the loss. The transformed momentum is then orthogonalized and mapped back to the original parameter space to generate the update direction. This allows the algorithm to address curvature and gradient anisotropy simultaneously.
- Based on the proposed approach, we develop a new optimizer, MALT, that incorporates two lightweight diagonal matrices as preconditioners and, similar to Muon, use Newton–Schulz for orthogonalization. The preconditioners are constructed as the exponential moving averages of row- and column-wise squared gradient norms, with the goal of preserving matrix structure without introducing excessive extra memory and computational cost. To stabilize the update scale, MALT further uses the norm grafting (Agarwal et al., 2020) to rescale update magnitude. Convergence analysis of MALT is provided for nonconvex losses under a stochastic setting.
- To robustify MALT against stochastic gradient noise, we propose an adaptive variant of it named MALTER. MALTER introduces a single scalar stepsize for the grafted update direction, which is adaptively adjusted using norm-based Adam-type noise adaptation in preconditioned space. Compared to MALT, MALTER shows improved empirical performance with negligible extra memory and computation overhead.
- We evaluate MALT and MALTER on pretraining GPT-2 Small, Medium, and Large models. Both methods consistently outperform Muon while costing nearly the same memory footprint and wall-clock time. MALT consistently improves over Muon, while MALTER provides further gains over MALT, suggesting that noise adaptive stepsize control and preconditioned orthogonalization are both important and complementary mechanisms for optimizer design.

2 RELATED WORKS

Muon and its variants. Recent Muon variants mainly improve the algorithm from three aspects: orthogonalization, update scaling, and row and column structure. The first line improves orthogonalization through Newton–Schulz implementations optimized for modern hardware (Zhang et al., 2026a), efficient approximations of the polar factor (Grishina et al., 2025; Amsel et al., 2025), low rank parametrizations (Refael et al., 2026; He et al., 2025), or momentum equilibration before orthogonalization to improve conditioning and Newton–Schulz accuracy (Chang et al., 2026). The second line studies update magnitude through interpretations of spectral norm constraint optimization (Lau et al., 2025; Nguyen et al., 2026) or adaptive scaling based on RMSProp, AdaGrad, Adam, and trust region methods (Si et al., 2025; Song et al., 2026; Li et al., 2026; Zhang et al., 2026b; Liu et al., 2026; Cheng et al., 2026). The third line addresses norm imbalance across rows and columns (Li et al., 2025; Zhang et al., 2026c), which may contribute to inactive neurons (Dewulf et al., 2026), using normalization after orthogonalization, decomposition into row magnitudes and directions (Lion et al., 2026), or normalization applied separately to each row (Deng et al., 2026; Yuan et al., 2026). However, these methods do not explicitly account for curvature anisotropy in the loss landscape.

Preconditioning for mitigating curvature anisotropy. Newton's method uses the Hessian to precondition gradients and can accelerate convergence, but computing, storing, and inverting the Hessian is prohibitive at scale (Nesterov, 2018; Lau et al., 2025). Practical methods therefore rely on cheaper curvature surrogates. AdaGrad (Duchi et al., 2011), RMSProp (Tieleman & Hinton, 2012), and Adam (Kingma & Ba, 2015) use elementwise gradient second moments as diagonal preconditioners, while Adafactor (Shazeer & Stern, 2018), SM3 (Anil et al., 2019), and Adam-mini (Zhang et al., 2025) reduce the associated memory through factorization, compression, or block sharing. AdaHessian (Yao et al., 2021) and Sophia (Liu et al., 2024) use estimates of the Hessian diagonal, whereas K-FAC (Martens & Grosse, 2015) approximates the Fisher information with Kronecker products. Matrix-aware methods such as Shampoo (Gupta et al., 2018), CASPR (Duvvuri et al., 2024), and SOAP (Vyas et al., 2025) instead construct structured preconditioners that exploit matrix or tensor structure. However, these methods primarily address curvature adaptation and do not explicitly account for anisotropy in matrix gradients (Lau et al., 2025). Although recent analyses show that Muon can itself provide implicit preconditioning benefits in special instances, including least square (Paquette et al., 2026), matrix factorization (Ma et al., 2026), phase retrieval (Braun et al., 2026), and deep learning (Davis & Drusvyatskiy, 2025). However, these benefits are problem-dependent and do not amount to an explicitly approximate local curvature of loss.

Recent works FISMO (Xu et al., 2026) and Mousse (Zhang et al., 2026d) also proposed curvature-aware Muon variants. In particular, FISMO incorporates anisotropic geometry through a Fisher metric with a Kronecker factorization and formulates the orthogonalized update as a trust region problem under this Fisher induced metric. Mousse instead uses preconditioner with a Kronecker structure in Shampoo, to construct a whitened space in which Muon updates are performed. Newton–Muon (Du & Su, 2026) is also closely related to our work, as it derives a curvature-aware Muon update from a Newton type surrogate and introduces a right preconditioner based on the second moment of the layer inputs. Unlike these methods, MALT uses diagonal preconditioners with low computational and memory overhead and updates them at every optimization step. This preconditioner preserves the row and column structure of matrix parameters while avoiding the storage of dense curvature factors and costly matrix operations such as eigendecomposition and matrix inversion.

3 MALT AND MALTER

We consider training an LLM formulated as solving the following unconstrained problem:

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

where $f:\mathbb{R}^{m\times n}\to\mathbb{R}$ is loss function and $\boldsymbol{X}\in\mathbb{R}^{m\times n}$ is a matrix parameter to be optimized. Let $\mathrm{Orth}\left(\boldsymbol{X}\right):=\boldsymbol{U}\boldsymbol{V}^\top$ denote the orthogonalization of matrix $\boldsymbol{X}\in\mathbb{R}^{m\times n}$, with $\boldsymbol{X}=\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^\top$ being its reduced singular value decomposition (SVD). The spectral gradient descent algorithm (SpecGD), which can be regarded as a simplified version of the Muon (Jordan et al., 2024), updates $\boldsymbol{X}$ as

$$\boldsymbol{X}_t=\boldsymbol{X}_{t-1}-\eta_t\,\mathrm{Orth}\left(\nabla f\left(\boldsymbol{X}_{t-1}\right)\right), \tag{2}$$

where $\nabla f(\boldsymbol{X}_{t-1})$ is the loss's gradient at $\boldsymbol{X}_{t-1}$. Compared to the gradient descent, SpecGD flattens the singular value spectra of the gradient and avoids the update being dominated by just a few leading eigen directions. To reduce the cost for training large models, Muon improves SpecGD by replacing the gradient with a momentum term that is the exponential average of past stochastic gradients, and approximately computes the orthogonalization step using the Newton–Schulz iterations.

3.1 Muon in A Preconditioned Space

On one hand, the Muon update (2) does not explicitly incorporate curvature information of the loss landscape, which can make Muon sensitive to curvature anisotropy as demonstrated in Fig. 1. On the other hand, it is widely observed that the Hessian of the loss function is ill-conditioned, both for training neural networks in general (Sagun et al., 2016; 2017; Ben Arous et al., 2024) and for language models specifically (Shen et al., 2025; Zhang et al., 2024; Tang et al., 2025).

This contrast motivates us to explore the possibility of improving Muon by injecting into it the local curvature information of the loss. To do so, we start by introducing the preconditioned gradient algorithm as steepest descent on a transformed space. To minimize a loss f that is possibly nonconvex, we introduce a surrogate function for it that aims to capture its curvature locally at $\boldsymbol{X}_t$ defined as:

$$f(\boldsymbol{X}_t + \boldsymbol{\Delta}) \approx f(\boldsymbol{X}_t) + \langle \nabla f(\boldsymbol{X}_t), \boldsymbol{\Delta} \rangle + \frac{1}{2} \langle \boldsymbol{\Delta}, \mathcal{B}_t(\boldsymbol{\Delta}) \rangle , \tag{3}$$

where $\boldsymbol{\Delta}_t$ is the small local perturbation that the above quadratic surrogate function can locally approximately well at $\boldsymbol{X}_t$. $\mathcal{B}_t : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ is a positive semi-definite linear mapping so that the quadratic surrogate function is convex. The approximation resembles the second-order Taylor expansion of f at $\boldsymbol{X}_t$. However, as the loss functions for DNN training are typically nonconvex, applying the exact expansion with $\mathcal{B}_t$ being the Hessian can make the surrogate unbounded below (Dauphin et al., 2014; Ghorbani et al., 2019). In addition, for large-scale problems with billions of parameters, computing, storing and inverting the Hessian are all unafforable (Lau et al., 2025). Therefore, $\mathcal{B}_t$ in (3) is understood as an replacement of the Hessian, known as a preconditioner, so that the second order term approximately captures the curvature of f locally at $\boldsymbol{X}_t$. The update direction is then obtained by minimizing the surrogate with respect to $\boldsymbol{\Delta}$:

$$\boldsymbol{\Delta}_t := \underset{\boldsymbol{\Delta}\in\mathbb{R}^{m\times n}}{\arg\min} \langle \nabla f(\boldsymbol{X}_t), \boldsymbol{\Delta} \rangle + \frac{1}{2} \langle \boldsymbol{\Delta}, \mathcal{B}_t(\boldsymbol{\Delta}) \rangle . \tag{4}$$

To understand the role of $\mathcal{B}_t$, we denote $\mathcal{B}_t^{1/2}$ as positive semi-definite square root of $\mathcal{B}_t$ and define

$$\widetilde{\boldsymbol{\Delta}} := \mathcal{B}_t^{1/2}(\boldsymbol{\Delta}), \tag{5}$$

as new variable in preconditioned space. Substituting $\widetilde{\boldsymbol{\Delta}}$ into the quadratic term of (4) yields

$$\langle \boldsymbol{\Delta}, \mathcal{B}_t(\boldsymbol{\Delta}) \rangle = \left\| \widetilde{\boldsymbol{\Delta}} \right\|_F^2 . \tag{6}$$

That is, the linear transform $\mathcal{B}_t^{1/2}$ induces a preconditioned space with coordinates $\widetilde{\boldsymbol{\Delta}}$, under which the surrogate function becomes isotropic. The update direction in the new space is given by:

$$\hat{\boldsymbol{\Delta}}_t := \underset{\widetilde{\boldsymbol{\Delta}}\in\mathbb{R}^{m\times n}}{\arg\min} \left\langle \mathcal{B}_t^{-1/2} \left(\nabla f(\boldsymbol{X}_t) \right), \widetilde{\boldsymbol{\Delta}} \right\rangle + \frac{1}{2} \left\| \widetilde{\boldsymbol{\Delta}} \right\|_F^2 , \tag{7}$$

with $\mathcal{B}_t^{-1/2} \left(\nabla f(\boldsymbol{X}_t) \right)$ being the gradient in the preconditioned space. The closed form solution of (7) is $\hat{\boldsymbol{\Delta}}_t := \mathcal{B}_t^{-1/2} \left(\nabla f(\boldsymbol{X}_t) \right)$, which is actually the steepest GD direction in preconditioned space.

So far we have seen under the linear transform (5), the quadratic surrogate of f is isotropic in the corresponding preconditioned space. However, the singular value spectra of the matrix-valued gradient $\mathcal{B}_t^{-1/2} \left(\nabla f(\boldsymbol{X}_t) \right)$ may still be isotropic and result in insufficient updates per iteration. To address this limitation, we apply the idea of Muon and orthogonalize the preconditioned gradient as

$$\widetilde{\boldsymbol{\Delta}}_t := \mathrm{Orth} \left(\mathcal{B}_t^{-1/2} \left(\nabla f(\boldsymbol{X}_t) \right) \right) . \tag{8}$$

Algorithm 1 MALT (**M**uon **A**ugmented by **L**ightweight **T**wo-sided preconditioning)

Input: learning rate η, momentum $\mu_1, \mu_2 \in [0, 1)$, batch size B and damping constant ϵ.
Initialize: $\boldsymbol{W}_0 \in \mathbb{R}^{m\times n}, \boldsymbol{M}_0 = \boldsymbol{0}_{m\times n}, \boldsymbol{l}_0 = \boldsymbol{0}_m, \boldsymbol{r}_0 = \boldsymbol{0}_n$.

for $t = 1, \ldots, T$ **do**
  Sample a minibatch $|\mathcal{I}_t| = B$ and calculate stochastic gradient $\boldsymbol{G}_t = \frac{\sum_{i\in\mathcal{I}_t} \nabla f(\boldsymbol{W}_{t-1};\xi_i)}{B}$
  $\boldsymbol{M}_t \leftarrow \mu_1 \boldsymbol{M}_{t-1} + (1-\mu_1)\boldsymbol{G}_t$
  $\boldsymbol{l}_t \leftarrow \mu_2 \boldsymbol{l}_{t-1} + (1-\mu_2)\left[\|\boldsymbol{G}_t[1,:]\|_F^2; \cdots; \|\boldsymbol{G}_t[m,:]\|_F^2\right]$
  $\boldsymbol{r}_t \leftarrow \mu_2 \boldsymbol{r}_{t-1} + (1-\mu_2)\left[\|\boldsymbol{G}_t[:,1]\|_F^2; \cdots; \|\boldsymbol{G}_t[:,n]\|_F^2\right]$
  $\boldsymbol{L}_t \leftarrow \mathrm{Diag}\left((\boldsymbol{l}_t+\epsilon)^{-1/8}\right)$, $\boldsymbol{R}_t \leftarrow \mathrm{Diag}\left((\boldsymbol{r}_t+\epsilon)^{-1/8}\right)$
  $\widetilde{\boldsymbol{M}}_t \leftarrow \boldsymbol{L}_t \boldsymbol{M}_t \boldsymbol{R}_t$
  $\boldsymbol{O}_t \leftarrow \mathrm{Orth}(\widetilde{\boldsymbol{M}}_t)$
  $\boldsymbol{D}_t \leftarrow \boldsymbol{L}_t \boldsymbol{O}_t \boldsymbol{R}_t$
  $\boldsymbol{W}_t \leftarrow \boldsymbol{W}_{t-1} - \eta \frac{\|\boldsymbol{O}_t\|_F}{\|\boldsymbol{D}_t\|_F+\epsilon}\boldsymbol{D}_t$
end for
return $\boldsymbol{W}_T$

The update direction in preconditioned space is then mapped back to the original space based on (5), leading to the final update direction given by

$$\boldsymbol{\Delta}_t = \mathcal{B}_t^{-1/2}\left(\widetilde{\boldsymbol{\Delta}}_t\right) = \mathcal{B}_t^{-1/2}\left(\mathrm{Orth}\left(\mathcal{B}_t^{-1/2}\left(\nabla f(\boldsymbol{X}_t)\right)\right)\right). \tag{9}$$

curvature-aware preconditioning to reduce curvature anisotropy, then orthogonalizes the preconditioned gradient in the preconditioned space to reduce gradient anisotropy, and finally maps the resulting direction back to the original parameter space to obtain the final update direction.

3.2 Lightweight Diagonal Preconditioner

To complete the picture, it remains to specify the preconditioner $\boldsymbol{\mathcal{B}}_t$. In the context of training LLMs, a key requirement is that it should be easy to compute and cheap to store so that the cost of applying preconditioning per iteration does not override its benefits in reducing the iteration complexity. To this end, we combine the idea of matrix-structure-aware preconditioning in Shampoo (Gupta et al., 2018) and diagonal approximation in Adam (Kingma & Ba, 2015), and propose the following preconditioning procedure.

Let $\boldsymbol{G}_t \in \mathbb{R}^{m\times n}$ be a stochastic gradient of f at iteration t. We maintain the exponential moving averages (EMA) of the squared norms of its rows and columns as:

$$(\boldsymbol{l}_t)_i = \mu_2(\boldsymbol{l}_{t-1})_i + (1-\mu_2)\,\|(\boldsymbol{G}_t)_{i:}\|_2^2, \quad \forall i \in [m], \tag{10}$$

$$(\boldsymbol{r}_t)_j = \mu_2(\boldsymbol{r}_{t-1})_j + (1-\mu_2)\,\|(\boldsymbol{G}_t)_{:j}\|_2^2, \quad \forall j \in [n], \tag{11}$$

where $(\boldsymbol{l}_t)_i$ and $(\boldsymbol{r}_t)_j$ denotes respectively the i- and j-th element of the vector $\boldsymbol{l}_t \in \mathbb{R}^m$ and $(\boldsymbol{r}_t)_j \in \mathbb{R}^n$; and $(\boldsymbol{G}_t)_{i:}$ and $(\boldsymbol{G}_t)_{:j}$ denotes respectively the i-th row and j-th column of $\boldsymbol{G}_t$. Then, the preconditioner is defined using two diagonal matrices

$$\boldsymbol{P}_t = \mathrm{Diag}\left((\boldsymbol{l}_t+\epsilon)^{-1/8}\right), \quad \boldsymbol{Q}_t = \mathrm{Diag}\left((\boldsymbol{r}_t+\epsilon)^{-1/8}\right) \tag{12}$$

as

$$\boldsymbol{\mathcal{B}}_t(\boldsymbol{G}) := \boldsymbol{P}_t \boldsymbol{G} \boldsymbol{Q}_t, \quad \forall\, G \in \mathbb{R}^{m\times n}, \tag{13}$$

where $\epsilon > 0$ is a small damping constant for numerical stability. Integrating the above preconditioner into (9) leads to MALT, as summarized in Algorithm 1.

The motivation for this preconditioner is fourfold. **1) Curvature-aid adaptive scaling.** The vectors $\boldsymbol{l}_t$ and $\boldsymbol{r}_t$ track the EMAs of squared gradient norms across the rows and columns, i.e., the diagonal parts of $\boldsymbol{G}_t\boldsymbol{G}_t^\top$ and $\boldsymbol{G}_t^\top\boldsymbol{G}_t$ over time. The intuition is that large entries in $\boldsymbol{l}_t$ or $\boldsymbol{r}_t$ suggest stronger local sensitivity or variation of the loss along the corresponding rows or column directions. $\boldsymbol{P}_t$ and

$\boldsymbol{Q}_t$ then suppress updates along such rows or columns and allow relatively larger updates along rows or columns with smaller accumulated squared gradient norm. **2) Matrix structure awareness.** Elementwise adaptive methods assign a separate scaling factor to every parameter entry. In contrast, MALT scales a matrix $\boldsymbol{Z} \in \mathbb{R}^{m\times n}$ according to $(\boldsymbol{P}_t \boldsymbol{Z} \boldsymbol{Q}_t)_{ij} = (\boldsymbol{P}_t)_{ii} \boldsymbol{Z}_{ij} (\boldsymbol{Q}_t)_{jj}$. Thus, all entries in the same row share the factor $(\boldsymbol{P}_t)_{ii}$, while all entries in the same column share the factor $(\boldsymbol{Q}_t)_{jj}$. Each entry is therefore scaled using information from both its row and its column, rather than from that entry alone. **3) Memory efficiency.** For each matrix parameter in $\mathbb{R}^{m\times n}$, MALT only stores two vectors, $\boldsymbol{l}_t \in \mathbb{R}^m$ and $\boldsymbol{r}_t \in \mathbb{R}^n$, resulting in $\mathcal{O}(m+n)$ additional optimizer state memory. This overhead is negligible compared with the $\mathcal{O}(mn)$ memory required for the first-order momentum and is substantially lower than storing dense matrix preconditioning factors. **4) Structured curvature approximation.** Existing studies (Zhang et al., 2024; 2025; Dong et al., 2025) have shown that the Hessian of neural networks and Transformers often has a near block diagonal structure. This suggests that useful curvature information can be captured without forming a full dense Hessian. Motivated by this, MALT uses separable row and column gradient norms as a lightweight structured approximation to local curvature, reasonably providing a middle ground between the elementwise scaling of Adam and expensive dense matrix preconditioning.

3.3 Stepsize Scaling: Norm Grafting and Noise Adaptation

In MALT, the preconditioned update $\boldsymbol{D}_t = \boldsymbol{L}_t \boldsymbol{O}_t \boldsymbol{R}_t$ can become overly large when some row or column gradient norms are small, because $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$ contain inverse eighth roots. This may require a sufficiently small learning rate for stable training. We therefore use grafting (Agarwal et al., 2020; Anil et al., 2020) to decouple update direction from magnitude: $\boldsymbol{D}_t$ determines the direction, while $\|\boldsymbol{O}_t\|_F$ determines its magnitude through $\frac{\|\boldsymbol{O}_t\|_F}{\|\boldsymbol{D}_t\|_F+\epsilon}\boldsymbol{D}_t$.

Grafting stabilizes the update magnitude but does not adapt it to stochastic gradient noise. Existing work suggests that noise adaptation can be as important as optimizing the update direction (Frans et al., 2025) in improving training efficiency. Successful examples include Adam improves over Signum (Bernstein et al., 2018), SOAP improves over SPlus (Frans et al., 2026), and AdaMuon improves over Muon (Si et al., 2025). We therefore apply this technique to MALT, resulting in MALTER as presented in Algorithm 2 of Appendix. MALTER follows the idea of NAMO (Zhang et al., 2026b), and adjust the update direction by a factor that estimates the signal-to-noise ratio of the gradient in the preconditioned space. It forms $\widetilde{\boldsymbol{M}}_t = \boldsymbol{L}_t \boldsymbol{M}_t \boldsymbol{R}_t$ and maintains

$$\nu_t = \mu_2 \nu_{t-1} + (1-\mu_2)\|\boldsymbol{L}_t \boldsymbol{G}_t \boldsymbol{R}_t\|_F^2. \tag{14}$$

The adaptation factor is then computed as:

$$\alpha_t = \frac{\sqrt{1-\mu_2^t}}{1-\mu_1^t+\epsilon}\frac{\|\widetilde{\boldsymbol{M}}_t\|_F}{\sqrt{\nu_t}+\epsilon}. \tag{15}$$

Thus, MALTER adapts the grafted update to gradient noise while retaining the preconditioned orthogonalization used to address curvature and gradient anisotropy with negligible cost.

4 Convergence Analysis of MALT

This section provides the convergence analysis of MALT, under the following standard assumptions on loss function (1). With a slight abuse of notation, we denote a stochastic gradient of f computed on sample ξ as $\nabla f(\boldsymbol{W};\xi)$.

Assumption 1. *(Lower boundedness). The loss function $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is lower bounded, i.e.,* $\inf_{\boldsymbol{W}\in\mathbb{R}^{m\times n}} f(\boldsymbol{W}) := f^\star > -\infty$.

Assumption 2. *(L-smoothness) The loss function $f : \mathbb{R}^{m\times n} \to \mathbb{R}$ is differentiable and there exists a constant $L > 0$ such that $\|\nabla f(\boldsymbol{W}_1) - \nabla f(\boldsymbol{W}_2)\|_F \le L\|\boldsymbol{W}_1 - \boldsymbol{W}_2\|_F$.*

Assumption 3. *(Bounded variance). Let ξ be a random sample that is independent of $\boldsymbol{W} \in \mathbb{R}^{m\times n}$, we assume $\nabla f(\boldsymbol{W};\xi)$ is an unbiased stochastic estimator of gradient $\nabla f(\boldsymbol{W})$ and have a bounded variance, i.e. $\mathbb{E}[\nabla f(\boldsymbol{W};\xi)] = \nabla f(\boldsymbol{W}), \mathbb{E}\left[\|\nabla f(\boldsymbol{W};\xi) - \nabla f(\boldsymbol{W})\|_F^2\right] \le \sigma^2$.*

Based on the above assumptions, we have the convergence guarantee for the proposed MALT.

Theorem 1. *Without loss of generality, we assume $m \geq n$ and denote $\Delta_{0,1} := f(\boldsymbol{W}_0) - f^\star, \Delta_{0,2} := \|\nabla f(\boldsymbol{W}_0)\|_F$. If Assumption 1-3 and $\eta = \frac{\epsilon^{\frac{5}{8}}}{n^{\frac{3}{4}}}\sqrt{\frac{2\Delta_{0,1}(1-\mu_1)}{5LT}}$, $\quad 1-\mu_1 = \min\left\{\epsilon^{\frac{5}{8}}\sqrt{\frac{5LB\Delta_{0,1}}{2Tn^{\frac{1}{2}}\sigma^2}}, 1\right\}$ are satisfied, then $\forall \epsilon \in (0,1)$ the iterates generated by MALT (cf. Algorithm 1) satisfy*

$$\min_{t=1,\cdots,T} \min\left\{\mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-1})\|_F^{\frac{1}{2}}\right], \mathbb{E}^2\left[\|\nabla f(\boldsymbol{W}_{t-1})\|_F^{\frac{1}{2}}\right]\right\} \lesssim \sqrt[4]{\frac{L\Delta_{0,1}\sigma^2}{TB}} + \frac{\Delta_{0,2}\sigma}{\sqrt{TLB\Delta_{0,1}}} + \sqrt{\frac{L\Delta_{0,1}}{T}} + \frac{\Delta_{0,2}}{T}. \tag{16}$$

Compared with standard Muon convergence bounds, the stationarity measure in Theorem 1 is different from the conventional choices such as the Euclidean norm $\mathbb{E}\left[\|\nabla f(\boldsymbol{W}_t)\|_F\right]$ (Nagashima & Iiduka, 2026; Zhang et al., 2026b) or nuclear norm $\mathbb{E}\left[\|\nabla f(\boldsymbol{W}_t)\|_*\right]$ (Shen et al., 2025). When we seek a high-accuracy solution such that $\mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-1})\|_F^{1/2}\right]$ is sufficiently small, the convergence metric is dominated by $\mathbb{E}^2\left[\|\nabla f(\boldsymbol{W}_{t-1})\|_F^{1/2}\right]$. This implies that the complexity of finding a point satisfying $\mathbb{E}^2\left[\|\nabla f(\boldsymbol{W}_{t-1})\|_F^{1/2}\right] \leq \delta$ is $\mathcal{O}\left(\frac{L\Delta_{0,1}\sigma^2}{B}\delta^{-4}\right)$. This matches the δ^{-4} accuracy dependence of existing stochastic non-convex analyses of Muon (Shen et al., 2025; Nagashima & Iiduka, 2026), while incurring a worse polynomial dependence on the matrix dimension n. However, since $\mathbb{E}^2\left[\|\nabla f(\boldsymbol{W}_{t-1})\|_F^{1/2}\right] \leq \mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-1})\|_F\right]$, this convergence metric indicates that our convergence guarantee is slightly weaker than those in existing results.

5 Experiments

We evaluate MALT and MALTER on GPT-2 pretraining and compare them with AdamW (Kingma & Ba, 2015) and Muon (Jordan et al., 2024) across different model scales in Section 5.1. We evaluate the time and memory overhead of MALT and MALTER, as well as other baselines in Section 5.2. We provide ablation studies in Section 5.3. We further compare with matrix optimizers SOAP (Vyas et al., 2025) and Mousse (Zhang et al., 2026d) in Section 5.4. Our experiment setting is specified below.

We train GPT-2 Small (124M), Medium (355M), and Large (774M) using the OpenWebText (Gokaslan et al., 2019) dataset, based on nanoGPT implementation (Karpathy, 2022). Following the standard Muon parameter grouping, Muon, SOAP, Mousse, MALT, and MALTER are applied to matrix parameters, while AdamW is used for vectors, scalars, embeddings, and output heads. Unless otherwise specified, we use a global batch size of 0.492M tokens per step and a warmup-stable learning rate schedule. For each optimizer, we perform a 10K-step learning-rate grid search, and select the learning rate that achieves the lowest validation loss. The selected learning rate is then used for the corresponding long-run experiment. GPT-2 Small, Medium, and Large are trained for 50K, 10K, and 18K steps, respectively. Complete learning-rate grids, hyperparameters, training protocols, and hardware details are provided in Appendix D.1.

5.1 Main Results

MALT and MALTER consistently outperform Muon in GPT-2 pretraining. Table 1 shows that MALT lowers Muon's validation loss by 0.0114, 0.0108, and 0.0104 on GPT-2 Small, Medium, and Large, respectively. MALTER lowers Muon's validation loss to 0.0241, 0.0277, and 0.0139, achieving the best result at every scale. Fig. 2 shows that MALT and MALTER consistently achieve lower validation loss than AdamW and Muon over iterations. Across the three model sizes, MALT matches Muon's final validation loss with **28.0%**, **5.0%**, and **5.6%** fewer steps, respectively, while MALTER requires **46.0%**, **15.0%**, and **16.7%** fewer steps than Muon.

5.2 Memory and Computation Overhead Analysis

MALT and MALTER consume comparable optimizer state memory as Muon. Table 2 reports the optimizer-state memory, optimizer-step time, and end-to-end training-step time comparison on

Model Size	Metric	AdamW	Muon	MALT	MALTER
Small (124M)	Train Loss	3.0467	3.0298	3.0193	**3.0063**
	Val Loss	3.0540	3.0472	3.0358	**3.0231**
Medium (355M)	Train Loss	2.9851	2.9599	2.9490	**2.9282**
	Val Loss	2.9971	2.9719	2.9611	**2.9442**
Large (774M)	Train Loss	2.8089	2.7695	2.7581	**2.7550**
	Val Loss	2.8267	2.7864	2.7760	**2.7725**

Table 1: Final training and validation losses achieved by different optimizers across model scales under batchsize of 0.492M tokens. The maximum numbers of training steps for the small, medium, and large models are 50K, 10K, and 18K, respectively.

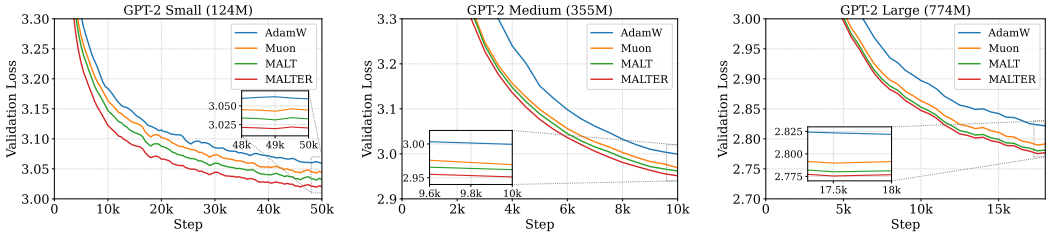


Figure 2: Validation losses for GPT-2 Small, Medium, and Large trained for 50K, 10K, and 18K steps with batchsize 0.492M tokens. MALT consistently improves over Muon, while MALTER achieves the best overall performance.

GPT-2 Medium (355M). The training-step time includes the time of forward pass, backward pass and optimizer step. It reflects the end-to-end time consumption of one iteration. As shown in Table 2, MALT and MALTER incur negligible optimizer state memory compared with Muon, while showing substantially less memory than SOAP and Mousse. For each matrix parameter, MALT adds only two vector states to Muon, and MALTER introduces one additional scalar stepsize. In contrast, SOAP and Mousse maintain matrix-valued preconditioning states, resulting in approximately 6 to 7 times the optimizer-state memory of Muon.

MALT and MALTER have comparable training step time as Muon. As shown in Table 2, MALT and MALTER introduce additional optimizer step time but achieve comparable training-step time to Muon, whereas SOAP and Mousse have much higher optimizer-step costs due to expensive matrix operations. Mousse requires 2 eigendecompositions every 10 steps, while SOAP uses QR-based basis updates, both of which result in noticeably higher optimizer-step time. All measurements are conducted on the same single H100 GPU. Detailed experimental settings are provided in Appendix D.

5.3 Ablation Studies

We conduct ablation studies on GPT-2 Small to see the effect of different placement of diagonal preconditioners relative to Newton–Schulz (NS) orthogonalization and to isolate the gain of diagonal preconditioning from the adaptive scalar stepsize.

Placement of diagonal preconditioners. We first compare MALT with two variants. MALT (Pre-NS only) applies the preconditioners before orthogonalization, with update $\mathrm{Orth}(\boldsymbol{L}_t\boldsymbol{M}_t\boldsymbol{R}_t)$. MALT (Post-NS only) applies the preconditioners only after orthogonalization, with update $\boldsymbol{L}_t\,\mathrm{Orth}(\boldsymbol{M}_t)\boldsymbol{R}_t$. In contrast, MALT applies preconditioners both before and after the NS operation. As shown in Fig. 4, both variants slightly outperform Muon but remain worse than MALT.

Effect of diagonal preconditioning and adaptive scalar stepsize. We compare Muon, MALT and MALTER with NAMO (Zhang et al., 2026b) which is Muon equipped with the adaptive scalar stepsize. As shown in Fig. 4, NAMO substantially reduces the validation loss by adding the scalar adaptive stepsize. MALTER, which is MALT with scalar adaptive stepsize, further achieves the lowest validation loss. This indicate that both diagonal preconditioning and adaptive stepsize contribute to the performance gain of MALTER.

Optimizer	**Opt. State Memory** (GB)	**Opt. Step Time** (ms)	**Training Step Time** (s)
AdamW	2.837	4.098	2.606
Muon	1.629	36.77	2.629
SOAP	12.10	314.3	2.909 (10.7% ↑)
Mousse	10.89	842.7	3.434 (30.6% ↑)
MALT	1.630	51.01	2.647 (0.7% ↑)
MALTER	1.630	62.25	2.650 (0.8% ↑)

Table 2: Per-iteration mmemory and computation overhead during GPT-2 Medium pretraining with batchsize 0.492M tokens. Percentages denote the relative increase over Muon.

Optimizer	**Small**	**Medium**
Muon	3.1630	2.9719
SOAP	3.1880	2.9840
Mousse	3.1547	2.9764
MALT	3.1463	2.9611
MALTER	**3.1223**	**2.9442**

Table 3: Final validation loss of matrix optimizers pretraining GPT-2 Small and Medium models for 10K steps with batchsize 0.492M tokens.

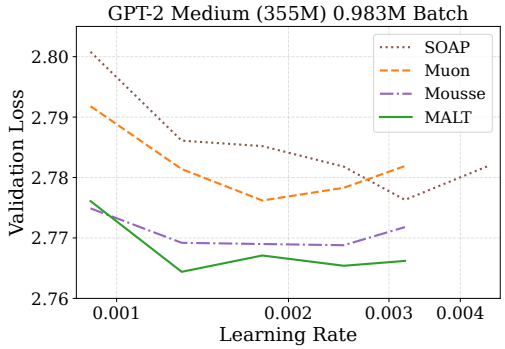


Figure 3: Validation losses of pretraining GPT-2 Medium model for 10K steps under the 0.983M batchsize with WSD learning rate schedule.

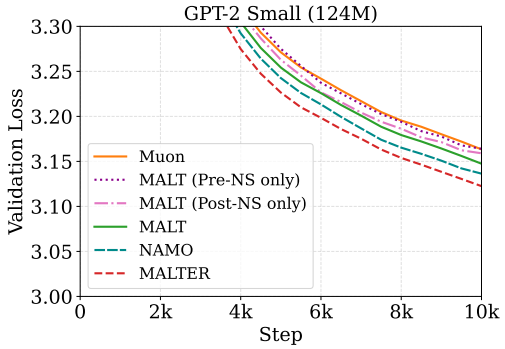


Figure 4: Ablation studies of MALT and MALTER on GPT-2 Small model for 10K steps under the 0.492M tokens batchsize setting.

5.4 Comparison with SOAP and Mousse: Influence of Batchsize

Table 3 compares MALT and MALTER with SOAP and Mousse under the same setting used in our main experiments. While Mousse slightly improves over Muon on GPT-2 Small, neither SOAP nor Mousse consistently outperforms Muon across the two model scales. In contrast, MALT and MALTER achieve lower validation loss on both models. We observe that the batchsize of 0.492M tokens may not fully reflect the potential of SOAP and Mousse. One possible explaination is that SOAP and Mousse are more sensitive to gradient noise under smaller batchsizes. The SOAP (Vyas et al., 2025) notes similar phenomenon that SOAP performs better under larger batchsizes.

This motivates us to further compare SOAP and Mousse with our methods and under a larger batchsize. We therefore increase the global batch size to 0.983M tokens. Similar to the Mousse (Zhang et al., 2026d), we adopt a 10%/80%/10% warmup-stable-decay (WSD) learning rate schedule. As shown in Fig. 3, Mousse outperforms Muon across different learning rates, while MALT attains the lowest validation loss among the compared methods.

6 Conclusion

We proposed MALT, a lightweight curvature aware Muon variant that integrates diagonal preconditioning with orthogonalization to address curvature and gradient anisotropy at nearly the same memory and computational cost as Muon. We further introduced MALTER, which adds norm based noise adaptive stepsize control in the preconditioned space, and established a stochastic nonconvex convergence guarantee for MALT. Experiments on GPT-2 pretraining show consistent improvements over Muon across model scales, highlighting curvature preconditioning and noise adaptation as complementary mechanisms for efficient matrix optimization.

AI USE STATEMENT

In this work, we used generative AI tools for assist in the writing of proofs design or provide feedback on research methodology or experiments, implement methods, assist with translation, interpret results. We have not used generative AI tools for provide critical ingredients for proving mathematical claims, propose or refine hypotheses, help develop theoretical models or conceptual frameworks, formulate mathematical claims. And generate synthetic data sets, clean and reformat dataset, support qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for create or modify scientific figures or images, create or edit software code summarize or analyse existing literature, discover research topics or identify gaps, brainstorming, sourcing/searching for information, edit a research paper to improve readability, identify relevant literature, format references, suggest a structure for a research paper. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by authors. LLM-polished proof and words were verified by authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*, 2020.

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the muon algorithm. *arXiv preprint arXiv:2505.16932*, 2025.

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

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

Gerard Ben Arous, Reza Gheissari, Jiaoyang Huang, and Aukosh Jagannath. High-dimensional sgd aligns with emerging outlier eigenspaces. In *International Conference on Learning Representations*, volume 2024, pp. 47732–47778, 2024.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Anima Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Guillaume Braun, Han Bao, Wei Huang, and Masaaki Imaizumi. Spectral gradient descent mitigates anisotropy-driven misalignment: A case study in phase retrieval. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Da Chang, Qiankun Shi, Lvgang Zhang, Yu Li, Ruijie Zhang, Yao Lu, Yongxiang Liu, and Ganzhao Yuan. Muoneq: Balancing before orthogonalization with lightweight equilibration. *arXiv preprint arXiv:2603.28254*, 2026.

Peng Cheng, Jiucheng Zang, Qingnan Li, Liheng Ma, Yufei Cui, Yingxue Zhang, Boxing Chen, Ming Jian, and Wen Tong. Trasmuon: Trust-region adaptive scaling for orthogonalized momentum optimizers. *arXiv preprint arXiv:2602.13498*, 2026.

Jeremy M. Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E. Dahl, and Justin Gilmer. Adaptive gradient methods at the edge of stability. In *Advances in Neural Information Processing Systems*, 2022.

Jeremy M. Cohen, Alex Damian, Ameet Talwalkar, J. Zico Kolter, and Jason D. Lee. Understanding optimization in deep learning with central flows. In *International Conference on Learning Representations*, 2025.

Yann N. Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Damek Davis and Dmitriy Drusvyatskiy. When do spectral gradient updates help in deep learning? *arXiv preprint arXiv:2512.04299*, 2025.

DeepSeek-AI. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Shenyang Deng, Zhuoli Ouyang, Tianyu Pang, Zihang Liu, Ruochen Jin, Shuhua Yu, and Yaoqing Yang. Rmnp: Row-momentum normalized preconditioning for scalable matrix-based optimization. *arXiv preprint arXiv:2603.20527*, 2026.

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

Zhaorui Dong, Yushun Zhang, Jianfeng Yao, and Ruoyu Sun. Towards quantifying the hessian structure of neural networks. *arXiv preprint arXiv:2505.02809*, 2025.

Zhehang Du and Weijie Su. The newton-muon optimizer. *arXiv preprint arXiv:2604.01472*, 2026.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

Sai Surya Duvvuri, Fnu Devvrit, Rohan Anil, Cho-Jui Hsieh, and Inderjit S. Dhillon. Combining axes preconditioners through kronecker approximation for deep learning. In *International Conference on Learning Representations*, 2024.

Kevin Frans, Pieter Abbeel, and Sergey Levine. What really matters in matrix-whitening optimizers? *arXiv preprint arXiv:2510.25000*, 2025.

Kevin Frans, Sergey Levine, and Pieter Abbeel. A stable whitening optimizer for efficient neural network training. *Advances in Neural Information Processing Systems*, 38:174086–174110, 2026.

Gemini Team, Google. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2232–2241. PMLR, 2019.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. Openwebtext corpus, 2019.

Ekaterina Grishina, Matvey Smirnov, and Maxim Rakhuba. Accelerating newton-schulz iteration for orthogonalization via chebyshev-type polynomials. *arXiv preprint arXiv:2506.10935*, 2025.

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

Chuan He, Zhanwang Deng, and Zhaosong Lu. Low-rank orthogonalization for large-scale matrix optimization with applications to foundation model training. *arXiv preprint arXiv:2509.11983*, 2025.

Zhendong Huang, Hengjie Cao, Fang Dong, Ruijun Huang, Mengyi Chen, Yifeng Yang, Xin Zhang, Anrui Chen, Mingzhi Dong, Yujiang Wang, et al. Spectra: Rethinking optimizers for llms under spectral anisotropy. *arXiv preprint arXiv:2602.11185*, 2026.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. In *International Conference on Learning Representations*, 2024.

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

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

Andrej Karpathy. nanogpt. `https://github.com/karpathy/nanoGPT`, 2022. GitHub repository.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

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.

Jingru Li, Yibo Fan, and Huan Li. Variance-adaptive muon: Accelerating llm pretraining with nsr-modulated and variance-scaled momentum. *arXiv preprint arXiv:2601.14603*, 2026.

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

Kai Lion, Florian Hübler, Bingcong Li, Antonio Orvieto, and Niao He. Muown: Row-norm control for muon optimization. *arXiv preprint arXiv:2605.10797*, 2026.

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

Ziyue Liu, Ruijie Zhang, Zhengyang Wang, Yequan Zhao, Yupeng Su, Zi Yang, and Zheng Zhang. Muon ^{2}: Boosting muon via adaptive second-moment preconditioning. *arXiv preprint arXiv:2604.09967*, 2026.

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

Jianhao Ma, Yu Huang, Yuejie Chi, and Yuxin Chen. Preconditioning benefits of spectral orthogonalization in muon. *arXiv preprint arXiv:2601.13474*, 2026.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417. PMLR, 2015.

Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal ai innovation. `https://ai.meta.com/blog/llama-4-multimodal-intelligence/`, 2025. Accessed: 2026-06-07.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

Shuntaro Nagashima and Hideaki Iiduka. Improved convergence rates of muon optimizer for nonconvex optimization. *arXiv preprint arXiv:2601.19400*, 2026.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on GPU clusters using Megatron-LM. *arXiv preprint arXiv:2104.04473*, 2021.

Yurii Nesterov. *Lectures on Convex Optimization*. Springer, 2018.

Tien-Phat Nguyen, Truong Nguyen, Minh-Phuc Truong, Tuc Nguyen, James Bailey, and Trung Le. Spectral flattening is all muon needs: How orthogonalization controls learning rate and convergence. *arXiv preprint arXiv:2605.13079*, 2026.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2 edition, 2006.

OpenAI. GPT-5.5 System Card. `https://openai.com/index/gpt-5-5-system-card/`, 2026. Accessed: 2026-06-07.

Elliot Paquette, Noah Marshall, Lucas Benigni, Guangyuan Wang, Atish Agarwala, and Courtney Paquette. Phases of muon: When muon eclipses signsgd. *arXiv preprint arXiv:2605.09552*, 2026.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

Aditya Ranganath. Navigating LLM valley: From AdamW to memory-efficient and matrix-based optimizers. *arXiv preprint arXiv:2605.09176*, 2026.

Yehonathan Refael, Guy Smorodinsky, Tom Tirer, and Ofir Lindenbaum. Sumo: Subspace-aware moment-orthogonalization for accelerating memory-efficient llm training. *Advances in Neural Information Processing Systems*, 38:147250–147281, 2026.

Levent Sagun, Leon Bottou, and Yann LeCun. Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*, 2016.

Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Levent Sagun, Utku Evci, V. Uğur Güney, Yann Dauphin, and Léon Bottou. Empirical analysis of the hessian of over-parametrized neural networks. In *International Conference on Learning Representations Workshop*, 2018.

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.

Chongjie Si, Debing Zhang, and Wei Shen. Adamuon: Adaptive muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025.

Zitao Song, Cedar Site Bai, Zhe Zhang, Brian Bullins, and David F Gleich. Decoupling variance and scale-invariant updates in adaptive gradient descent for unified vector and matrix optimization. *arXiv preprint arXiv:2602.06880*, 2026.

Q. Y. Tang et al. Investigating the overlooked hessian structure. In *International Conference on Machine Learning*, 2025.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

Nikhil Vyas. Modded-nanogpt with soap optimizer. `https://github.com/nikhilvyas/modded-nanogpt-SOAP`, 2024. GitHub repository.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham M. Kakade. Soap: Improving and stabilizing shampoo using adam for language modeling. In *International Conference on Learning Representations*, 2025.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. In *International Conference on Learning Representations*, 2025.

Chenrui Xu, Wenjing Yan, and Ying-Jun Angela Zhang. Fismo: Fisher-structured momentum-orthogonalized optimizer. *arXiv preprint arXiv:2601.21750*, 2026.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael W. 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.

Jinghui Yuan, Jiaxuan Zou, Shuo Wang, Yong Liu, and Feiping Nie. Nora: Normalized orthogonal row alignment for scalable matrix optimizer. *arXiv preprint arXiv:2605.03769*, 2026.

Jack Zhang, Noah Amsel, Berlin Chen, and Tri Dao. Gram newton–schulz: A fast, hardware-aware newton–schulz algorithm for muon. `https://tridao.me/blog/2026/gram-newton-schulz/`, 2026a.

Minxin Zhang, Yuxuan Liu, and Hayden Schaeffer. Adam improves muon: Adaptive moment estimation with orthogonalized momentum. *arXiv preprint arXiv:2602.17080*, 2026b.

Ruijie Zhang, Yequan Zhao, Ziyue Liu, Zhengyang Wang, and Zheng Zhang. Muon+: Towards better muon via one additional normalization step. *arXiv preprint arXiv:2602.21545*, 2026c.

Yechen Zhang, Shuhao Xing, Junhao Huang, Kai Lv, Yunhua Zhou, Xipeng Qiu, Qipeng Guo, and Kai Chen. Mousse: Rectifying the geometry of muon with curvature-aware preconditioning. *arXiv preprint arXiv:2603.09697*, 2026d.

Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why transformers need adam: A hessian perspective. *Advances in neural information processing systems*, 37:131786–131823, 2024.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Diederik P. Kingma, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. In *International Conference on Learning Representations*, 2025.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Appendices

We firstly discusses the relation between the proposed MALT and existing optimizers in Appendix A. Then Appendix B gives the detailed proof for Theorem 1. The simulation setting of Fig. 1 is given in Appendix C

Algorithm 2 MALTER: Adaptive MALT

Input: learning rate η, momentum $\mu_1, \mu_2 \in [0, 1)$, batch size B and damping constant ϵ.
Initialize: $\boldsymbol{W}_0 \in \mathbb{R}^{m\times n}$, $\boldsymbol{M}_0 = \boldsymbol{0}_{m\times n}$, $\boldsymbol{l}_0 = \boldsymbol{0}_m$, $\boldsymbol{r}_0 = \boldsymbol{0}_n$, and $\nu_0 = 0$.

for $t = 1, \dots, T$ **do**
  Sample a minibatch $|\mathcal{I}_t| = B$ and compute the stochastic gradient $\boldsymbol{G}_t = \frac{1}{B}\sum_{\xi\in\mathcal{I}_t} \nabla f_i(\boldsymbol{W}_{t-1}; \xi)$.
  $\boldsymbol{M}_t \leftarrow \mu_1 \boldsymbol{M}_{t-1} + (1-\mu_1)\boldsymbol{G}_t$.
  $\boldsymbol{l}_t \leftarrow \mu_2 \boldsymbol{l}_{t-1} + (1-\mu_2)\left[|\boldsymbol{G}_t[1,:]|_2^2; \cdots; |\boldsymbol{G}_t[m,:]|_2^2\right]$.
  $\boldsymbol{r}_t \leftarrow \mu_2 \boldsymbol{r}_{t-1} + (1-\mu_2)\left[|\boldsymbol{G}_t[:,1]|_2^2; \cdots; |\boldsymbol{G}_t[:,n]|_2^2\right]$.
  $\boldsymbol{L}_t \leftarrow \mathrm{Diag}\left((\boldsymbol{l}_t + \epsilon)^{-1/8}\right)$ and $\boldsymbol{R}_t \leftarrow \mathrm{Diag}\left((\boldsymbol{r}_t + \epsilon)^{-1/8}\right)$.
  $\widetilde{\boldsymbol{M}}_t \leftarrow \boldsymbol{L}_t \boldsymbol{M}_t \boldsymbol{R}_t$
  $\boldsymbol{O}_t \leftarrow \mathrm{Orth}(\widetilde{\boldsymbol{M}}_t)$
  $\boldsymbol{D}_t \leftarrow \boldsymbol{L}_t \boldsymbol{O}_t \boldsymbol{R}_t$
  $v_t \leftarrow \mu_2 v_{t-1} + (1-\mu_2)\left\|\boldsymbol{L}_t \boldsymbol{G}_t \boldsymbol{R}_t\right\|_F^2$.
  $\alpha_t \leftarrow \eta \frac{\sqrt{1-\mu_2^t}}{1-\mu_1^t+\epsilon} \frac{\left\|\widetilde{\boldsymbol{M}}_t\right\|_F}{\sqrt{\nu_t}+\epsilon}$.
  $\boldsymbol{W}_t \leftarrow \boldsymbol{W}_{t-1} - \eta\alpha_t \frac{\|\boldsymbol{O}_t\|_F}{\|\boldsymbol{D}_t\|_F+\epsilon}\boldsymbol{D}_t$
end for
return $\boldsymbol{W}_T$.

A Relation to Existing Optimizers

This section discusses the relation between the proposed MALT and existing optimizers.

Relation to Adafactor. Adafactor (Shazeer & Stern, 2018) reduces memory usage by storing only the row and column sums of the moving average of squared gradients. It uses their normalized outer product to approximate the full elementwise second moment estimate. MALT shares a similar idea but uses the row and column sums for a different purpose. It constructs diagonal preconditioners on the left and right of the momentum matrix before orthogonalization, rather than using them to approximate the full elementwise second moment estimate.

Relation to Shampoo. Given the gradient $\boldsymbol{G}_t \in \mathbb{R}^{m\times n}$, Shampoo (Gupta et al., 2018) maintains dense accumulators for its rows and columns and defines the corresponding preconditioners as

$$\begin{aligned}
\boldsymbol{L}_t &= \mu_2 \boldsymbol{L}_{t-1} + (1-\mu_2)\,\boldsymbol{G}_t \boldsymbol{G}_t^\top \\
\boldsymbol{R}_t &= \mu_2 \boldsymbol{R}_{t-1} + (1-\mu_2)\,\boldsymbol{G}_t^\top \boldsymbol{G}_t \\
\boldsymbol{P}_t &= (\boldsymbol{L}_t + \epsilon \boldsymbol{I}_m)^{-1/8}, \quad \boldsymbol{Q}_t = (\boldsymbol{R}_t + \epsilon \boldsymbol{I}_n)^{-1/8} \\
\boldsymbol{W}_{t+1} &= \boldsymbol{W}_t - \eta_t \boldsymbol{P}_t\,(\boldsymbol{P}_t \boldsymbol{G}_t \boldsymbol{Q}_t)\,\boldsymbol{Q}_t.
\end{aligned} \tag{17}$$

Thus, Shampoo essentially firstly takes the gradient descent step in preconditioned space under full dense matrix preconditioners $\boldsymbol{P}_t, \boldsymbol{Q}_t$ and then maps the update direction back to the original space. The preconditioners in (12) retain this left and right structure but replace the dense matrices of Shampoo with diagonal matrices. The inverse eighth power in (12) is adopted from the corresponding Shampoo factors in (17).

Relation to SOAP. SOAP (Vyas et al., 2025) combines Shampoo preconditioning with Adam in the eigenbasis of the Shampoo preconditioners. Let

$$\boldsymbol{L}_t = \boldsymbol{U}_t \boldsymbol{\Lambda}_t \boldsymbol{U}_t^\top, \qquad \boldsymbol{R}_t = \boldsymbol{V}_t \boldsymbol{\Gamma}_t \boldsymbol{V}_t^\top, \qquad \widehat{\boldsymbol{G}}_t = \boldsymbol{U}_t^\top \boldsymbol{G}_t \boldsymbol{V}_t, \tag{18}$$

where $\boldsymbol{U}_t \in \mathbb{O}^{m\times m}$ and $\boldsymbol{V}_t \in \mathbb{O}^{n\times n}$ contain the eigenvectors of $\boldsymbol{L}_t$ and $\boldsymbol{R}_t$. SOAP applies Adam to the rotated gradient $\widehat{\boldsymbol{G}}_t$ and maps the resulting direction back to the original parameter space:

$$\begin{aligned} \widehat{\boldsymbol{m}}_t &= \beta_1 \widehat{\boldsymbol{m}}_{t-1} + (1-\beta_1)\widehat{\boldsymbol{G}}_t, \\ \widehat{\boldsymbol{v}}_t &= \beta_2 \widehat{\boldsymbol{v}}_{t-1} + (1-\beta_2)\widehat{\boldsymbol{G}}_t \odot \widehat{\boldsymbol{G}}_t, \\ \widehat{\boldsymbol{\Delta}}_t &= \frac{\widehat{\boldsymbol{m}}_t}{\sqrt{\widehat{\boldsymbol{v}}_t}+\epsilon}, \qquad \boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \eta_t \boldsymbol{U}_t \widehat{\boldsymbol{\Delta}}_t \boldsymbol{V}_t^\top. \end{aligned} \tag{19}$$

Hence, SOAP can be viewed as applying Adam in the eigenspace induced by the Shampoo preconditioners.

One may try to combine SOAP with Muon by applying orthogonalization in the same eigenspace:

$$\widehat{\boldsymbol{G}}_t = \boldsymbol{U}_t^\top \boldsymbol{G}_t \boldsymbol{V}_t, \qquad \widehat{\boldsymbol{O}}_t = \mathrm{Orth}(\widehat{\boldsymbol{G}}_t), \qquad \boldsymbol{\Delta}_t = \boldsymbol{U}_t \widehat{\boldsymbol{O}}_t \boldsymbol{V}_t^\top. \tag{20}$$

This construction, however, gives exactly the same direction as applying Muon in the original parameter space. The polar factor is equivariant under orthogonal transformations:

$$\mathrm{Orth}\left(\boldsymbol{U}_t^\top \boldsymbol{G}_t \boldsymbol{V}_t\right) = \boldsymbol{U}_t^\top \mathrm{Orth}(\boldsymbol{G}_t)\boldsymbol{V}_t. \tag{21}$$

It follows that

$$\begin{aligned} \boldsymbol{\Delta}_t &= \boldsymbol{U}_t \,\mathrm{Orth}\left(\boldsymbol{U}_t^\top \boldsymbol{G}_t \boldsymbol{V}_t\right) \boldsymbol{V}_t^\top \\ &= \mathrm{Orth}(\boldsymbol{G}_t). \end{aligned} \tag{22}$$

Thus, rotating the gradient into the Shampoo eigenbasis before orthogonalization does not change the Muon direction. SOAP behaves differently because Adam is not invariant under orthogonal transformation: its elementwise second moment depends on the basis in which it is computed.

Relation to NorMuon. NorMuon (Li et al., 2025) integrated Muon with neuron-wise adaptive learning rates, which maintains a row-wise second-moment estimate

$$(\boldsymbol{v}_t)_i = \beta_2 (\boldsymbol{v}_{t-1})_i + (1-\beta_2)\frac{1}{n}\|(\boldsymbol{O}_t)_{i,:}\|_2^2, \qquad i \in [m]$$

that $\boldsymbol{O}_t = \mathrm{Orth}(\boldsymbol{M}_t)$ is the Muon direction from momentum $\boldsymbol{M}_t$. Letting $V_t = v_t \mathbf{1}_n^\top$, the orthogonalized direction is normalized row-wise as

$$\hat{\boldsymbol{O}}_t = \boldsymbol{O}_t \oslash \left(\sqrt{\boldsymbol{V}_t} + \epsilon\right),$$

followed by a global rescaling of its magnitude before updating the parameter. Thus, NorMuon primarily addresses the imbalance of update magnitudes across neurons *after* orthogonalization. (Li et al., 2025)

In contrast, MALT constructs row- and column-wise diagonal preconditioners from the stochastic gradient and uses them as precondition. Hence, MALT changes the geometry in which the momentum is orthogonalized and subsequently maps the orthogonalized direction back to the original parameter space, with the goal of mitigating curvature anisotropy. NorMuon and MALT therefore address different but complementary sources of optimization inefficiency: neuron-wise update-scale imbalance and curvature anisotropy, respectively.

Relation to Mousse and FISMO. Independent works Mousse (Zhang et al., 2026d) and FISMO (Xu et al., 2026) are the most closely related curvature-aware Muon variants. They share the same goal as MALT: introducing preconditioning to address curvature anisotropy issue of vanilla Muon. There are three main key differences between these two works and ours.

1. FISMO and Mousse use denser representations of curvature. FISMO describes the local geometry using a Fisher metric with a Kronecker structure, whereas Mousse uses dense Shampoo factors to construct a preconditioned space. For a matrix parameter in $\mathbb{R}^{m\times n}$, both methods maintain matrices on the left and right, requiring $\mathcal{O}(m^2+n^2)$ additional optimizer state.
2. These dense factors also require additionally expensive matrix computations. FISMO computes matrix inverses or inverse roots to obtain its left and right preconditioners, while Mousse periodically performs eigendecompositions of the Shampoo factors.

3. Mousse updates the eigenspaces of its preconditioner periodically rather than at every step. Between two consecutive updates, the same preconditioner are reused. The preconditioner may therefore respond with a delay when the local curvature changes rapidly during training, which can result in unstable training as found in existing work (Frans et al., 2026).

In contrast, MALT uses diagonal preconditioners on the left and right and updates them at every step. This design may capture less curvature information than dense preconditioning, but requires substantially less memory and computation. MALT therefore remains close to Muon in cost while still adapting the update to curvature.

B Proof of Theorem 1

Based on L-smoothness property in Assumption 2, there is

$$
\begin{aligned}
\mathbb{E}\left[f\left(\boldsymbol{W}_t\right)\right] &\leq \mathbb{E}\left[f\left(\boldsymbol{W}_{t-1}\right)\right] + \mathbb{E}\left[\left\langle \nabla f\left(\boldsymbol{W}_{t-1}\right), \boldsymbol{W}_t - \boldsymbol{W}_{t-1}\right\rangle\right] + \frac{L}{2}\mathbb{E}\left[\left\|\boldsymbol{W}_t - \boldsymbol{W}_{t-1}\right\|_F^2\right] \\
&\leq \mathbb{E}\left[f\left(\boldsymbol{W}_{t-1}\right)\right] - \mathbb{E}\left[\frac{\sqrt{n}\eta}{\left\|\boldsymbol{D}_t\right\|_F + \epsilon}\left\langle \nabla f\left(\boldsymbol{W}_{t-1}\right), \boldsymbol{L}_t\boldsymbol{O}_t\boldsymbol{R}_t\right\rangle\right] + \frac{L}{2}n\eta^2 \\
&= \mathbb{E}\left[f\left(\boldsymbol{W}_{t-1}\right)\right] - \mathbb{E}\left[\frac{\sqrt{n}\eta}{\left\|\boldsymbol{L}_t\boldsymbol{O}_t\boldsymbol{R}_t\right\|_F + \epsilon}\left\langle \boldsymbol{L}_t\left(\nabla f\left(\boldsymbol{W}_{t-1}\right) - \boldsymbol{M}_t\right)\boldsymbol{R}_t, \boldsymbol{O}_t\right\rangle\right] \\
&\quad - \mathbb{E}\left[\frac{\sqrt{n}\eta}{\left\|\boldsymbol{L}_t\boldsymbol{O}_t\boldsymbol{R}_t\right\|_F + \epsilon}\left\langle \boldsymbol{L}_t\boldsymbol{M}_t\boldsymbol{R}_t, \mathrm{Orth}\left(\boldsymbol{L}_t\boldsymbol{M}_t\boldsymbol{R}_t\right)\right\rangle + \frac{L}{2}n\eta^2\right] \\
&\overset{(i)}{\leq} \mathbb{E}\left[f\left(\boldsymbol{W}_{t-1}\right)\right] + \mathbb{E}\left[\frac{n\eta}{\left\|\boldsymbol{L}_t\boldsymbol{O}_t\boldsymbol{R}_t\right\|_F + \epsilon}\left\|\boldsymbol{L}_t\left(\nabla f\left(\boldsymbol{W}_{t-1}\right) - \boldsymbol{M}_t\right)\boldsymbol{R}_t\right\|_F\right] + \frac{L}{2}n\eta^2 \\
&\quad - \mathbb{E}\left[\frac{\sqrt{n}\eta}{\left\|\boldsymbol{L}_t\boldsymbol{O}_t\boldsymbol{R}_t\right\|_F + \epsilon}\left\|\boldsymbol{L}_t\boldsymbol{M}_t\boldsymbol{R}_t\right\|_*\right] \\
&\overset{(ii)}{\leq} \mathbb{E}\left[f\left(\boldsymbol{W}_{t-1}\right)\right] + \mathbb{E}\left[\frac{n\eta}{\left\|\boldsymbol{L}_t\boldsymbol{O}_t\boldsymbol{R}_t\right\|_F + \epsilon}\left\|\boldsymbol{L}_t\left(\nabla f\left(\boldsymbol{W}_{t-1}\right) - \boldsymbol{M}_t\right)\boldsymbol{R}_t\right\|_F\right] + \frac{L}{2}n\eta^2 \\
&\quad - \mathbb{E}\left[\frac{\sqrt{n}\eta}{\left\|\boldsymbol{L}_t\boldsymbol{O}_t\boldsymbol{R}_t\right\|_F + \epsilon}\left(\left\|\boldsymbol{L}_t\nabla f\left(\boldsymbol{W}_{t-1}\right)\boldsymbol{R}_t\right\|_* - \left\|\boldsymbol{L}_t\left(\nabla f\left(\boldsymbol{W}_{t-1}\right) - \boldsymbol{M}_t\right)\boldsymbol{R}_t\right\|_*\right)\right] \\
&\overset{(iii)}{\leq} \mathbb{E}\left[f\left(\boldsymbol{W}_{t-1}\right)\right] + 2n\epsilon^{-1}\eta\mathbb{E}\left[\left\|\boldsymbol{L}_t\left(\nabla f\left(\boldsymbol{W}_{t-1}\right) - \boldsymbol{M}_t\right)\boldsymbol{R}_t\right\|_F\right] + \frac{L}{2}n\eta^2 \\
&\quad - \mathbb{E}\left[\frac{\sqrt{n}\eta}{\left\|\boldsymbol{L}_t\boldsymbol{O}_t\boldsymbol{R}_t\right\|_F + \epsilon}\left\|\boldsymbol{L}_t\nabla f\left(\boldsymbol{W}_{t-1}\right)\boldsymbol{R}_t\right\|_F\right] \\
&\overset{(iv)}{\leq} \mathbb{E}\left[f\left(\boldsymbol{W}_{t-1}\right)\right] + 2n\epsilon^{-5/4}\eta\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right) - \boldsymbol{M}_t\right\|_F\right] + \frac{L}{2}n\eta^2 \\
&\quad - \frac{\sqrt{n}\eta}{\epsilon^{-1/4}\sqrt{n} + \epsilon}\mathbb{E}\left[\left\|\boldsymbol{L}_t\nabla f\left(\boldsymbol{W}_{t-1}\right)\boldsymbol{R}_t\right\|_F\right] \\
&\leq \mathbb{E}\left[f\left(\boldsymbol{W}_{t-1}\right)\right] + 2n\epsilon^{-5/4}\eta\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right) - \boldsymbol{M}_t\right\|_F\right] \\
&\quad - \frac{\epsilon^{1/4}}{2}\eta\mathbb{E}\left[\left\|\boldsymbol{L}_t\nabla f\left(\boldsymbol{W}_{t-1}\right)\boldsymbol{R}_t\right\|_F\right] + \frac{L}{2}n\eta^2,
\end{aligned}
\tag{23}
$$

where (i) and (ii) are due to Cauchy-Schwarz and triangle inequalities, respectively. (iii) is because norm inequality $\|\boldsymbol{X}\|_F \leq \|\boldsymbol{X}\|_* \leq \sqrt{\mathrm{rank}\left(\boldsymbol{X}\right)}\|\boldsymbol{X}\|_F$. (iv) and last inequality are due to $\|\boldsymbol{L}_t\| \leq \epsilon^{-1/8}, \|\boldsymbol{R}_t\| \leq \epsilon^{-1/8}, \epsilon \in (0,1)$.

Now denote $s_{t,\tau} := \frac{1-\mu_2}{1-\mu_2^t}\mu_2^{t-\tau}$ we come to lower bound the term $\mathbb{E}\left[\left\|\boldsymbol{L}_t\nabla f\left(\boldsymbol{W}_{t-1}\right)\boldsymbol{R}_t\right\|_F\right]$ as

$$
\begin{aligned}
&\mathbb{E}\left[\left\|\boldsymbol{L}_t\nabla f\left(\boldsymbol{W}_{t-1}\right)\boldsymbol{R}_t\right\|_F\right] \geq \mathbb{E}\left[\sigma_{\min}\left(\boldsymbol{L}_t\right)\sigma_{\min}\left(\boldsymbol{R}_t\right)\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F\right] \\
&\geq \mathbb{E}\left[\frac{1}{\left(\left\|\boldsymbol{l}_t\right\|_1 + \epsilon\right)^{\frac{1}{8}}}\frac{1}{\left(\left\|\boldsymbol{r}_t\right\|_1 + \epsilon\right)^{\frac{1}{8}}}\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F\right]
\end{aligned}
$$

$$
\begin{aligned}
&\overset{(a)}{\geq} \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\mathbb{E}\left[\left(\left\|\boldsymbol{l}_t\right\|_1+\epsilon\right)^{\frac{1}{4}}\right]} \\
&\overset{(b)}{\geq} \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\mathbb{E}\left[\left\|\boldsymbol{l}_t\right\|_1^{\frac{1}{4}}\right]+\epsilon^{\frac{1}{4}}} \\
&= \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\left(1-\mu_2^t\right)^{\frac{1}{4}}\mathbb{E}\left[\sqrt[4]{\sum_{\tau=1}^t s_{t,\tau}\left\|\boldsymbol{G}_\tau\right\|_F^2}\right]+\epsilon^{\frac{1}{4}}} \\
&\overset{(c)}{\geq} \frac{\left(1-\mu_2^t\right)^{-\frac{1}{4}}\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\mathbb{E}\left[\sqrt{\sqrt{\sum_{\tau=1}^t s_{t,\tau}\left\|\boldsymbol{G}_\tau-\nabla f\left(\boldsymbol{W}_{\tau-1}\right)\right\|_F^2}+\sqrt{\sum_{\tau=1}^t s_{t,\tau}\left\|\nabla f\left(\boldsymbol{W}_{\tau-1}\right)\right\|_F^2}}\right]+\epsilon^{\frac{1}{4}}} \\
&\overset{(d)}{\geq} \frac{\left(1-\mu_2^t\right)^{-\frac{1}{4}}\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\left(\mathbb{E}\left[\sqrt[4]{\sum_{\tau=1}^t s_{t,\tau}\left\|\boldsymbol{G}_\tau-\nabla f\left(\boldsymbol{W}_{\tau-1}\right)\right\|_F^2}\right]+\mathbb{E}\left[\sqrt[4]{\sum_{\tau=1}^t s_{t,\tau}\left\|\nabla f\left(\boldsymbol{W}_{\tau-1}\right)\right\|_F^2}\right]\right)+\epsilon^{\frac{1}{4}}} \\
&\overset{(e)}{\geq} \frac{\left(1-\mu_2^t\right)^{-\frac{1}{4}}\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\left(\mathbb{E}^{\frac{1}{4}}\left[\sum_{\tau=1}^t s_{t,\tau}\left\|\boldsymbol{G}_\tau-\nabla f\left(\boldsymbol{W}_{\tau-1}\right)\right\|_F^2\right]+\mathbb{E}^{\frac{1}{4}}\left[\sum_{\tau=1}^t s_{t,\tau}\left\|\nabla f\left(\boldsymbol{W}_{\tau-1}\right)\right\|_F^2\right]\right)+\epsilon^{\frac{1}{4}}} \\
&\overset{(f)}{\geq} \frac{\left(1-\mu_2^t\right)^{-\frac{1}{4}}\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\frac{\sqrt{\sigma}\left(1-\mu_2^t\right)^{\frac{1}{4}}+(B\epsilon)^{\frac{1}{4}}}{B^{\frac{1}{4}}\left(1-\mu_2^t\right)^{\frac{1}{4}}}+\mathbb{E}\left[\sqrt{\sqrt{\sum_{\tau=1}^t s_{t,\tau}\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^2}+\sqrt{\sum_{\tau=1}^t s_{t,\tau}\left\|\nabla f\left(\boldsymbol{W}_{\tau-1}\right)-\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^2}}\right]} \\
&\overset{(g)}{\geq} \frac{\left(1-\mu_2^t\right)^{-\frac{1}{4}}\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\frac{\sqrt{\sigma}}{B^{\frac{1}{4}}}+\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]+\mathbb{E}^{\frac{1}{4}}\left[\sum_{\tau=1}^t s_{t,\tau}\left(t-\tau\right)^2L^2\eta^2n\right]+\frac{\epsilon^{\frac{1}{4}}}{\left(1-\mu_2^t\right)^{\frac{1}{4}}}} \\
&\overset{(h)}{\geq} \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\frac{\sqrt{\sigma}}{B^{\frac{1}{4}}}+\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]+\left(1-\mu_2^t\right)^{\frac{1}{4}}\cdot\sqrt{\eta L}n^{\frac{1}{4}}\left(\frac{1-\mu_2}{1-\mu_2^t}\right)^{\frac{1}{4}}\left(\frac{2}{(1-\mu_2)^3}\right)^{\frac{1}{4}}+\epsilon^{\frac{1}{4}}} \\
&\geq \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]+1+2n^{\frac{1}{4}}\sqrt{\frac{\eta L}{1-\mu_2}}+\frac{\sqrt{\sigma}}{B^{\frac{1}{4}}}}, \\
&\geq \frac{1}{2}\min\left\{\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right], \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{1+2n^{\frac{1}{4}}\sqrt{\frac{\eta L}{1-\mu_2}}+\frac{\sqrt{\sigma}}{B^{\frac{1}{4}}}}\right\},
\end{aligned}
\tag{24}
$$

where (a) is because $\left\|\boldsymbol{l}_t\right\|_1=\left\|\boldsymbol{r}_t\right\|_1$ and Cauchy–Schwarz inequality $\mathbb{E}^2\left[XY\right]\leq\mathbb{E}\left[X^2\right]\mathbb{E}\left[Y^2\right]$. (b) is due to $(a+b)^p\leq a^p+b^p, \forall a,b\geq 0, p\in(0,1)$. (c) is based on $\sum_{\tau=1}^t\frac{1-\mu_2}{1-\mu_2^t}\mu_2^{t-\tau}$ and Minkowski inequality. (d) is due to $\sqrt{a+b}\leq\sqrt{a}+\sqrt{b}$. (e) is because Jensen inequality $\mathbb{E}\left[X^{\frac{1}{4}}\right]\leq\mathbb{E}^{\frac{1}{4}}\left[X\right], \forall X\geq 0$. (f) uses the Assumption 3 and Minkowski inequality again. (g) uses the Assumption 2, 11th line of Algorithm 1, $\sum_{\tau=1}^t\frac{1-\mu_2}{1-\mu_2^t}\mu_2^{t-\tau}$ and $\sqrt{a+b}\leq\sqrt{a}+\sqrt{b}$ again. (h) is due to $\sum_{\tau=1}^\infty\mu_2^\tau\tau^2\leq\frac{2}{(1-\mu_2)^3}$. The last two inequalities are due to $\epsilon\in(0,1)$ and inequality $\frac{x^2}{ax+b}\geq\frac{x^2}{2\max\{ax,b\}}=\frac{1}{2}\min\left\{\frac{x}{a},\frac{x^2}{b}\right\}, \forall a,b>0, \forall x\geq 0$.

Substituting (24) into (23) has

$$\mathbb{E}\left[f\left(\boldsymbol{W}_t\right)\right] \leq \mathbb{E}\left[f\left(\boldsymbol{W}_{t-1}\right)\right] + 2n\epsilon^{-5/4}\eta\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right) - \boldsymbol{M}_t\right\|_F\right] + \frac{L}{2}n\eta^2 - \frac{\epsilon^{1/4}\eta}{4}\min\left\{\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right], \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{1 + 2n^{\frac{1}{4}}\sqrt{\frac{\eta L}{1-\mu_2}} + \frac{\sqrt{\sigma}}{B^{\frac{1}{4}}}}\right\}. \tag{25}$$

Further substituting the result in Lemma 1 into (25) and summing over $t = 1, \ldots, T$ yield

$$\begin{aligned}
&\frac{\epsilon^{1/4}\eta}{4}\sum_{t=1}^{T}\min\left\{\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right], \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{1 + 2n^{\frac{1}{4}}\sqrt{\frac{\eta L}{1-\mu_2}} + \frac{\sqrt{\sigma}}{B^{\frac{1}{4}}}}\right\} \\
&\leq \Delta_{0,1} + 2n\epsilon^{-5/4}\eta\sum_{t=1}^{T}\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right) - \boldsymbol{M}_t\right\|_F\right] + \frac{L}{2}n\eta^2 T \\
&\leq \Delta_{0,1} + 2n\epsilon^{-\frac{5}{4}}\eta\sum_{t=1}^{T}\left(\sqrt{\frac{1-\mu_1}{1+\mu_1}}\frac{\sigma}{\sqrt{B}} + \mu_1^t\left\|\nabla f\left(\boldsymbol{W}_0\right)\right\|_F + \frac{\mu_1 L\eta\sqrt{n}}{1-\mu_1}\right) + \frac{L}{2}n\eta^2 T \\
&\leq \Delta_{0,1} + 2n\epsilon^{-5/4}\eta T\sqrt{\frac{1-\mu_1}{1+\mu_1}}\frac{\sigma}{\sqrt{B}} + \frac{2n\epsilon^{-5/4}\eta}{1-\mu_1}\left\|\nabla f\left(\boldsymbol{W}_0\right)\right\|_F \\
&\quad + \frac{2\epsilon^{-5/4}\mu_1 L n^{\frac{3}{2}}\eta^2 T}{1-\mu_1} + \frac{L}{2}n\eta^2 T.
\end{aligned} \tag{26}$$

Thus, we can obtain

$$\begin{aligned}
\min_{t=1,\cdots,T}\min\left\{\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right], \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{1 + 2n^{\frac{1}{4}}\sqrt{\frac{\eta L}{1-\mu_2}} + \frac{\sqrt{\sigma}}{B^{\frac{1}{4}}}}\right\} &\leq \frac{4\epsilon^{-1/4}\Delta_{0,1}}{T\eta} + 2\epsilon^{-1/4}Ln\eta \\
&\quad + 8\epsilon^{-3/2}n\sqrt{\frac{1-\mu_1}{1+\mu_1}}\frac{\sigma}{\sqrt{B}} + \frac{8\epsilon^{-3/2}n\Delta_{0,2}}{\left(1-\mu_1\right)T} + \frac{8\epsilon^{-3/2}\mu_1 L n^{3/2}\eta}{1-\mu_1},
\end{aligned} \tag{27}$$

where $\Delta_{0,1} := f\left(\boldsymbol{W}_0\right) - f^\star, \Delta_{0,2} := \left\|\nabla f\left(\boldsymbol{W}_0\right)\right\|_F$.

For the right hand side of the above result, there is

$$\begin{aligned}
\frac{4\epsilon^{-1/4}\Delta_{0,1}}{T\eta} + 2\epsilon^{-1/4}Ln\eta + \frac{8\epsilon^{-3/2}\mu_1 L n^{3/2}\eta}{1-\mu_1} &\leq \frac{4\epsilon^{-1/4}\Delta_{0,1}}{T\eta} + \frac{10\epsilon^{-3/2}Ln^{3/2}\eta}{1-\mu_1} \\
&= 4\epsilon^{-\frac{7}{8}}\sqrt{\frac{10n^{\frac{3}{2}}L\Delta_{0,1}}{T\left(1-\mu_1\right)}},
\end{aligned} \tag{28}$$

where the equality is due to the step size condition $\eta = \frac{\epsilon^{\frac{5}{8}}}{n^{\frac{3}{4}}}\sqrt{\frac{2\Delta_{0,1}(1-\mu_1)}{5LT}}$. Then (27) becomes

$$\begin{aligned}
&\min_{t=1,\cdots,T}\min\left\{\mathbb{E}\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right], \frac{\mathbb{E}^2\left[\left\|\nabla f\left(\boldsymbol{W}_{t-1}\right)\right\|_F^{\frac{1}{2}}\right]}{1 + 2n^{\frac{1}{4}}\sqrt{\frac{\eta L}{1-\mu_2}} + \frac{\sqrt{\sigma}}{B^{\frac{1}{4}}}}\right\} \\
&\leq 8\epsilon^{-\frac{3}{2}}n\left(\frac{\sigma\sqrt{1-\mu_1}}{\sqrt{B}} + \frac{\Delta_{0,2}}{\left(1-\mu_1\right)T}\right) + \epsilon^{-\frac{7}{8}}\sqrt{\frac{10n^{\frac{3}{2}}L\Delta_{0,1}}{T\left(1-\mu_1\right)}}
\end{aligned} \tag{29}$$

Now set $1 - \mu_1 = \min\left\{\epsilon^{\frac{5}{8}}\sqrt{\frac{5LB\Delta_{0,1}}{2Tn^{\frac{1}{2}}\sigma^2}}, 1\right\}$. If $\epsilon^{\frac{5}{8}}\sqrt{\frac{5LB\Delta_{0,1}}{2Tn^{\frac{1}{2}}\sigma^2}} \leq 1$, then the right hand side of above upper bound becomes

$$4\epsilon^{-\frac{7}{8}}\sqrt{\frac{10n^{\frac{3}{2}}L\Delta_{0,1}}{T\left(1-\mu_1\right)}} + 8\epsilon^{-\frac{3}{2}}n\frac{\sigma\sqrt{1-\mu_1}}{\sqrt{B}} + \frac{8\epsilon^{-\frac{3}{2}}n\Delta_{0,2}}{\left(1-\mu_1\right)T} \leq$$

$$16\epsilon^{-\frac{19}{16}} n^{\frac{7}{8}} \sqrt[4]{\frac{L\Delta_{0,1}\sigma^2}{TB}} + 8\epsilon^{-\frac{17}{8}} n^{\frac{5}{4}} \sqrt{\frac{\Delta_{0,2}^2\sigma^2}{TLB\Delta_{0,1}}}. \tag{30}$$

If $\epsilon^{\frac{5}{8}} \sqrt{\frac{5LB\Delta_{0,1}}{2Tn^{\frac{1}{2}}\sigma^2}} \geq 1$, (27) becomes

$$\begin{aligned}
&\min_{t=1,\cdots,T} \min \left\{ \mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-1})\|_F^{\frac{1}{2}} \right], \frac{\mathbb{E}^2\left[\|\nabla f(\boldsymbol{W}_{t-1})\|_F^{\frac{1}{2}} \right]}{1 + 2n^{\frac{1}{4}} \sqrt{\frac{\eta L}{1-\mu_2}} + \frac{\sqrt{\sigma}}{B^{\frac{1}{4}}}} \right\} \\
&\leq 16\epsilon^{-\frac{7}{8}} n^{\frac{3}{4}} \sqrt{\frac{L\Delta_{0,1}}{T}} + 8\epsilon^{-\frac{3}{2}} n \frac{\sigma}{\sqrt{B}} + \frac{8\epsilon^{-\frac{3}{2}} n\Delta_{0,2}}{T} \\
&\leq 31\epsilon^{-\frac{7}{8}} n^{\frac{3}{4}} \sqrt{\frac{L\Delta_{0,1}}{T}} + \frac{8\epsilon^{-\frac{3}{2}} n\Delta_{0,2}}{T},
\end{aligned} \tag{31}$$

where the last inequality is due to $\sigma \leq \epsilon^{\frac{5}{8}} \sqrt{\frac{5LB\Delta_{0,1}}{2Tn^{\frac{1}{2}}}}$.

Finally, combining the results in the above two cases can obtain the (16).

Lemma 1. *If the loss function satisfies Assumption 2, then there is*

$$\mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-1}) - \boldsymbol{M}_t\|_F \right] \leq \sqrt{\frac{1-\mu_1}{1+\mu_1}} \frac{\sigma}{\sqrt{B}} + \mu_1^t \|\nabla f(\boldsymbol{W}_0)\|_F + \frac{\mu_1 L\eta\sqrt{n}}{1-\mu_1}. \tag{32}$$

Proof. The proof idea follows that of (Shen et al., 2025) and the difference is that we consider the zero initialization. We construct $\boldsymbol{C}_t = \mu_1 \boldsymbol{C}_{t-1} + (1-\mu_1)\nabla f(\boldsymbol{W}_{t-1}), \forall t \geq 1$ and $\boldsymbol{C}_0 = \boldsymbol{0}_{m\times n}$.

$$\begin{aligned}
\mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-1}) - \boldsymbol{M}_t\|_F \right] &\leq \mathbb{E}\left[\|\boldsymbol{C}_t - \boldsymbol{M}_t\|_F \right] + \mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-1}) - \boldsymbol{C}_t\|_F \right] \\
&= \mathbb{E}\left[\left\| (1-\mu_1) \sum_{\tau=1}^{t} \mu_1^{t-\tau} \left(\nabla f(\boldsymbol{W}_{\tau-1}) - \boldsymbol{G}_\tau \right) \right\|_F \right] \\
&\quad + \mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-1}) - \mu_1 \boldsymbol{C}_{t-1} - (1-\mu_1)\nabla f(\boldsymbol{W}_{t-1})\|_F \right] \\
&\leq \sqrt{\frac{1-\mu_1}{1+\mu_1}} \frac{\sigma}{\sqrt{B}} + \mu_1 \mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-2}) - \boldsymbol{C}_{t-1}\|_F \right] \\
&\quad + \mu_1 \mathbb{E}\left[\|\nabla f(\boldsymbol{W}_{t-1}) - \nabla f(\boldsymbol{W}_{t-2})\|_F \right] \\
&\leq \sqrt{\frac{1-\mu_1}{1+\mu_1}} \frac{\sigma}{\sqrt{B}} + \mu_1^{t-1} \mathbb{E}\left[\|\nabla f(\boldsymbol{W}_0) - \boldsymbol{C}_1\|_F \right] + \frac{\mu_1 L\eta\sqrt{n}}{1-\mu_1} \\
&= \sqrt{\frac{1-\mu_1}{1+\mu_1}} \frac{\sigma}{\sqrt{B}} + \mu_1^t \|\nabla f(\boldsymbol{W}_0)\|_F + \frac{\mu_1 L\eta\sqrt{n}}{1-\mu_1},
\end{aligned} \tag{33}$$

where the second inequality follows the proof of Lemma A.4 of (Shen et al., 2025), and the last equality uses the $\boldsymbol{C}_1 = (1-\mu_1)\nabla f(\boldsymbol{W}_0)$ due to $\boldsymbol{C}_0 = \boldsymbol{0}_{m\times n}$. □

C Simulation Setting of Fig. 1

We use a matrix quadratic regression problem to illustrate the effect of curvature anisotropy on Muon and the benefit of preconditioning. Specifically, we consider

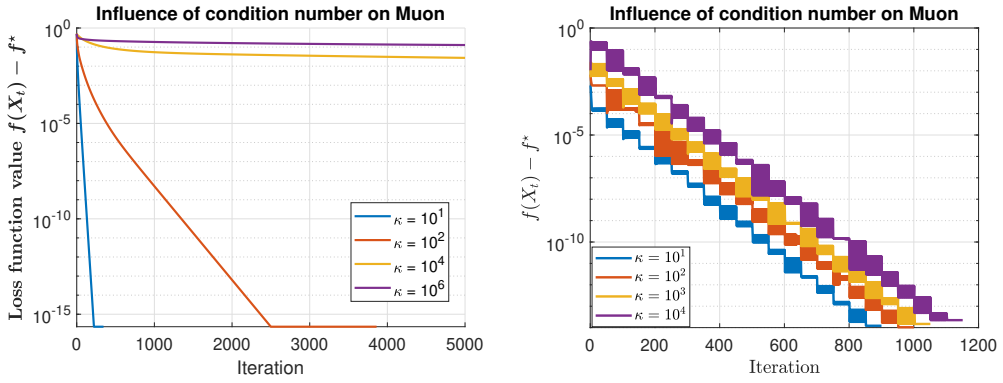
$$\min_{\boldsymbol{X}\in\mathbb{R}^{r\times r}} f(\boldsymbol{G}) := \frac{1}{2} \left\| \boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^\top - \boldsymbol{C} \right\|_F^2, \tag{34}$$

where $n = 50, r = 5, \boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n\times r}$. In vectorized form, the Hessian with respect to $\mathrm{vec}(\boldsymbol{X})$ is $\boldsymbol{H}_X = (\boldsymbol{B}^\top\boldsymbol{B}) \otimes (\boldsymbol{A}^\top\boldsymbol{A})$, and therefore its condition number satisfies $\kappa_X = \mathrm{cond}(\boldsymbol{H}_X) = \mathrm{cond}(\boldsymbol{B})^2 \,\mathrm{cond}(\boldsymbol{A})^2$. To generate controlled-condition problems, we first construct a rank-r matrix $\boldsymbol{C} = \boldsymbol{U}^\star \boldsymbol{S}^\star \boldsymbol{V}^{\star\top}$, where $\boldsymbol{U}_\star$ and $\boldsymbol{V}_\star$ are orthonormal factors and $\boldsymbol{S}^\star$ is a normalized random core matrix. We then set $\boldsymbol{A} = \boldsymbol{U}^\star \boldsymbol{R}_1, \boldsymbol{B} = \boldsymbol{V}^\star \boldsymbol{R}_2$, where $\boldsymbol{R}_1, \boldsymbol{R}_2 \in \mathbb{R}^{r\times r}$ are random matrices with prescribed condition number. We test four condition numbers $\kappa_X \in \{10^1, 10^2, 10^3, 10^4\}$.

For Fig. 1(a), we use the exact polar version of Muon without nuclear-norm scaling. The optimizer used in Fig. 1(b) is the exact-Hessian preconditioned Muon update. For the quadratic objective above, the exact whitening preconditioners are chosen as $\boldsymbol{P} = (\boldsymbol{A}^\top \boldsymbol{A})^{-1/2}, \boldsymbol{Q} = (\boldsymbol{B}^\top \boldsymbol{B})^{-1/2}$. Thus, the update is

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t - \eta_t (\boldsymbol{A}^\top \boldsymbol{A})^{-1/2} \operatorname{Orth}\left((\boldsymbol{A}^\top \boldsymbol{A})^{-1/2} \nabla f(\boldsymbol{X}_t) (\boldsymbol{B}^\top \boldsymbol{B})^{-1/2}\right) (\boldsymbol{B}^\top \boldsymbol{B})^{-1/2}. \quad (35)$$

Both simulations use the same initialization $\boldsymbol{X}_0 = \boldsymbol{0}$, the same maximum number of iterations $T = 5000$, and the same warmup-stable-decay learning rate schedule. The learning rate linearly warms up to $\eta_{\max} = 10^{-3}$ during the first 500 iterations, remains constant for 3500 iterations, and then linearly decays to $\eta_{\min} = 0$ during the last 1000 iterations. For each setting, we report the optimality gap $f(\boldsymbol{X}_t) - f^\star$, where $f^\star$ is computed using the closed-form least-squares solution.



(a) Muon (b) Preconditioned Muon

Figure 5: Additional step size settings for Muon-style updates. (a) Nuclear-norm scaled polar update. (b) Exact polar Muon with geometric step size decay.

To further examine whether the observed sensitivity of Muon is caused by a particular learning rate schedule, we also test two additional learning rate schedules for Muon-style updates on the same matrix quadratic regression problem (34). The first setting uses a nuclear-norm scaled polar update (Lau et al., 2025) as

$$\boldsymbol{G}_{t+1} = \boldsymbol{G}_t - \eta_t \|\nabla f(\boldsymbol{G}_t)\|_* \boldsymbol{O}_t, \quad (36)$$

which uses constant base step size $\eta_t = \eta_0 = \frac{C_\eta}{L_G r}$,, where L_G is the largest eigenvalue of the Hessian with respect to $\operatorname{vec}(\boldsymbol{G})$ and $C_\eta = 2$.

The second setting uses an exact polar Muon direction with a periodically decayed geometric learning rate schedule, where the step size is periodically decayed as $\eta_t = \eta_0 \rho^{\lfloor (t-1)/p \rfloor}$, where p is the decay period and ρ is the decay factor. In our implementation, we set $p = 50$ and $\rho = 0.5$. The initial stepsize is chosen according to the natural scale of the least-squares solution:$\eta_0 = C_\eta \frac{\|\boldsymbol{G}_{\text{lsq}}\|_F}{\sqrt{r}}$, where $C_\eta = 0.5$ and $\boldsymbol{G}_{\text{lsq}}$ is the closed-form least-squares solution.

Fig. 5 further reports Muon-style updates under these two learning rate schedules. In Fig. 5(a), we can observe that the nuclear-norm scaled polar update is still clearly sensitive to the Hessian condition number: as κ increases, the convergence becomes much slower. This observation is consistent with Theorem 3.2 in (Lau et al., 2025), which predicts a slower convergence rate when the Hessian condition number becomes larger. In contrast, Fig. 5(b) shows that Muon with geometric stepsize decay is much less sensitive to κ, as the curves exhibit similar staircase-like decreasing trends across different condition numbers.

These results suggest that the sensitivity of Muon to ill-conditioned curvature of loss (34) depends on the learning rate schedule. We use the WSD schedule in Fig. 1 because WSD is commonly used and studied in modern language-model training. Therefore, the main WSD-based comparison is more aligned with the training protocol used in deep learning and large-scale language-model pretraining.

D Additional Experimental Details and Results

D.1 Experimental Settings

Models and datasets. The GPT-2 model implementation follows NanoGPT (Karpathy, 2022). In order to validate the scalability of our proposed optimizers, we consider three model sizes: GPT-2 Small (124M), GPT-2 Medium (355M), and GPT-2 Large (774M), with detailed configuration listed in Table 4. All experiments are conducted on the OpenWebText dataset (Gokaslan et al., 2019), which contains approximately 9B training tokens and 4.4M validation tokens.

Model	n_{layer}	n_{head}	d_{model}	Params
GPT-2 Small	12	12	768	124M
GPT-2 Medium	24	16	1024	355M
GPT-2 Large	36	20	1280	774M

Table 4: GPT-2 model configurations used in our experiments.

Baselines. For AdamW (Kingma & Ba, 2015), we use the PyTorch implementation. For Muon (Jordan et al., 2024), we adapt the implementation from the MoonshotAI (Liu et al., 2025), using RMS update scaling and five Newton–Schulz iterations for orthogonalization. For SOAP (Vyas et al., 2025), we follow the NanoGPT-SOAP implementation (Vyas, 2024). The Shampoo eigenbasis is initialized using an exact eigendecomposition, while subsequent basis updates are performed every $T = 10$ steps using one power-iteration step followed by QR decomposition. For Mousse (Zhang et al., 2026d), we follow implementation released by the authors. Mousse update Shampoo eigenbasis every $T = 10$ steps using eigendecomposition. One thing differs from the original Mousse implementation is that we adopt RMS update scaling instead of spectral norm scaling.

Training and evaluation protocols. We consider two training settings in our experiments.

Standard setting. Unless otherwise specified, we use a global batch size of 480 sequences and a block size of 1024, corresponding to 491,520 tokens per optimizer step. For GPT-2 Small, Medium, and Large, we use mini-batch size and gradient accumulation steps pairs of $(60, 8)$, $(40, 12)$, and $(20, 24)$, respectively, corresponding to the same global batch size of 480 sequences. We use a warmup–stable learning-rate schedule, where the learning rate is linearly warmed up during the first 2K steps.

Large-batch WSD setting. For the experiments in Fig. 3, we additionally consider a larger-batch setting with 960 sequences per optimizer step, corresponding to 983,040 tokens. We train GPT-2 Medium model for 10K steps using a 10%–80%–10% warmup–stable–decay (WSD) schedule, where the learning rate is linearly warmed up during the first 1K steps, kept constant for the following 8K steps, and linearly decayed to zero during the final 1K steps.

Every 500 steps, we estimate the training and validation losses by averaging over 200 randomly sampled evaluation mini-batches from each split.

Hyperparameters. For AdamW, we use $(\beta_1, \beta_2) = (0.9, 0.95)$. For Muon, we set the momentum coefficient to $\beta = 0.95$. For MALT and MALTER, we use $(\mu_1, \mu_2) = (0.95, 0.99)$ and set the damping constant to $\epsilon = 10^{-8}$. For SOAP, its Adam moment coefficients are set to $(\beta_1, \beta_2) = (0.95, 0.95)$, and $\beta_{\text{Shampoo}} = 0.95$. Preconditioners of SOAP are updated every $T = 10$ iterations. For Mousse, momentum coefficient is $\beta = 0.95$, without Nesterov momentum. And we set Mousse $\beta_{\text{Shampoo}} = 0.95$, damping 10^{-5}, and spectral-tempering exponent $\alpha = 0.125$. Mousse preconditioners are updated every $T = 10$ iterations.

We apply weight decay 0.1 to the hidden weight matrices, but not to the embeddings, the tied output head, biases, or LayerNorm parameters. For matrix optimizers, Muon, SOAP, Mousse, MALT, and MALTER, embeddings, normalization parameters, biases, and other one-dimensional parameters are optimized using AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and 0 weight decay.

Learning rate tuning. For the experiments in Fig. 2 and Table 3, we first perform a grid search over peak learning rates, as shown in Table 5. Each optimizer is trained for 10K steps, and we select the peak learning rate that achieves the lowest validation loss during the 10K-step run. The selected learning rates are reported in Table 6. To evaluate longer-horizon performance, we continue training

GPT-2 Small and Large from their corresponding 10K-step checkpoints using the selected learning rates. GPT-2 Small, Medium, and Large are trained for 50K, 10K, and 20K steps in total, respectively.

For the large-batch experiments in Fig. 3, we perform a grid search shown in Table 7 under WSD learning rate schedule, and select the learning rate achieving lowest final validation loss at 10K steps.

Hardware. Pretraining experiments are conducted on cloud GPU clusters equipped with 4 H100 GPUs or 4–8 H20 GPUs, as well as a workstation with 2 PRO6000 GPUs. All wall-clock time and memory measurements are performed on a single H100 GPU.

Model Size	Optimizer	Learning Rate Grid
Small (124M)	AdamW, Muon, MALT	{0.0005, 0.0009, 0.0013, 0.0018, 0.0025, 0.0032}
	SOAP, Mousse	{0.0005, 0.0009, 0.0013, 0.0018, 0.0025, 0.0032}
	MALTER	{0.007, 0.009, 0.012, 0.015, 0.018, 0.025}
Medium (355M)	AdamW, Muon, MALT	{0.0005, 0.0009, 0.0013, 0.0018, 0.0025, 0.0032}
	SOAP, Mousse	{0.0005, 0.0009, 0.0013, 0.0018, 0.0025, 0.0032}
	MALTER	{0.005, 0.006, 0.007, 0.009, 0.013, 0.015}
Large (774M)	AdamW, Muon, MALT	{0.0003, 0.0005, 0.0006, 0.0007, 0.0009, 0.0013}
	MALTER	{0.003, 0.005, 0.006, 0.007, 0.009, 0.013}

Table 5: Full learning rate search grids for experiments in Fig. 2 and Table 3. AdamW, Muon, SOAP, Mousse, and MALT share the same search space, while MALTER is tuned on a distinct grid.

Model Size	Optimal Learning Rate					
	AdamW	**Muon**	**SOAP**	**Mousse**	**MALT**	**MALTER**
Small (124M)	0.0013	0.0013	0.0018	0.0013	0.0013	0.012
Medium (355M)	0.0009	0.0009	0.0009	0.0009	0.0009	0.009
Large (774M)	0.0009	0.0007	−−	−−	0.0007	0.006

Table 6: Optimal learning rates found via 10K step grid search for each model scale and optimizer for experiments in Fig. 2 and Table 3.

Model Size	Optimizer	Learning Rate Grid
Medium (355M)	Muon, Mousse, MALT	{0.0009, 0.0013, 0.0018, 0.0025, 0.0032}
	SOAP	{0.0009, 0.0013, 0.0018, 0.0025, 0.0032, 0.0045}

Table 7: Full learning rate search grids for experiments in Fig. 3 under large-batch WSD setting.

Runtime and Memory Measurement. We measure runtime and optimizer-state memory under the same *Standard Setting* GPT-2 Medium model, batch size, training precision, and hardware configuration for all methods. All optimizer states are stored in FP32. For runtime, we use `time.perf_counter()` with `torch.cuda.synchronize()` before and after each measured region to account for asynchronous CUDA execution. We discard the first 200 training iterations before collecting timing statistics. The training-step time measures the full training iteration, including gradient accumulation, forward and backward passes, gradient clipping, the optimizer update, and gradient clearing, while excluding evaluation and checkpointing. The optimizer-step time measures only the call to `scaler.step(optimizer)`.

Optimizer-state memory is measured by summing the storage of all tensors in `optimizer.state`, where the memory of each tensor is computed as the product of its number of elements and element size. We report this quantity after optimizer states have been initialized.

Ablation Study Settings. All ablation experiments are conducted on GPT-2 Small for 10K steps under the standard training and evaluation setting. For the preconditioner placement ablation, MALT

(Pre-NS only) uses the update $\mathrm{Orth}(\boldsymbol{L}_t\boldsymbol{M}_t\boldsymbol{R}_t)$ without norm grafting. MALT (Post-NS only) uses the raw update $\boldsymbol{L}_t\,\mathrm{Orth}(\boldsymbol{M}_t)\boldsymbol{R}_t$ and applies norm grafting to match the Frobenius norm of $\mathrm{Orth}(\boldsymbol{M}_t)$. All other optimizer hyperparameters are kept the same as MALT. For MALT, MALTER, and the two placement variants, we search the learning rate over $\{5\times10^{-4}, 9\times10^{-4}, 1.3\times10^{-3}, 1.8\times10^{-3}, 2.5\times10^{-3}, 3.2\times10^{-3}\}$. For NAMO (Zhang et al., 2026b), we use the learning-rate grid $\{5\times10^{-3}, 7\times10^{-3}, 9\times10^{-3}, 1.2\times10^{-2}, 1.5\times10^{-2}\}$. For each method, we report the result corresponding to the learning rate that achieves the lowest final validation loss.

D.2 Training Loss Curves for Fig. 2

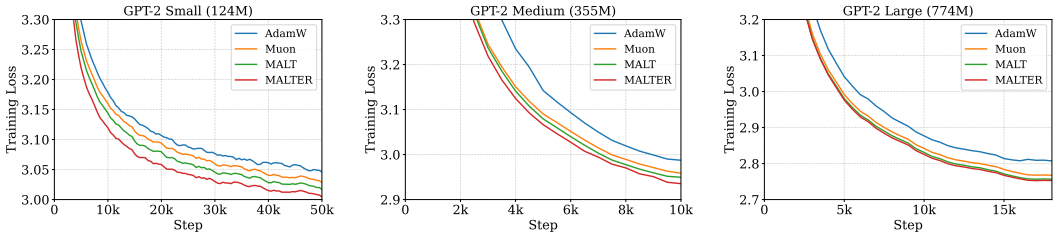


Figure 6: Training losses for GPT-2 Small, Medium, and Large trained for 50K, 10K, and 18K steps with batchsize 0.492M tokens.

D.3 Learning-Rate Sweep Results for Fig. 2 and Table 3

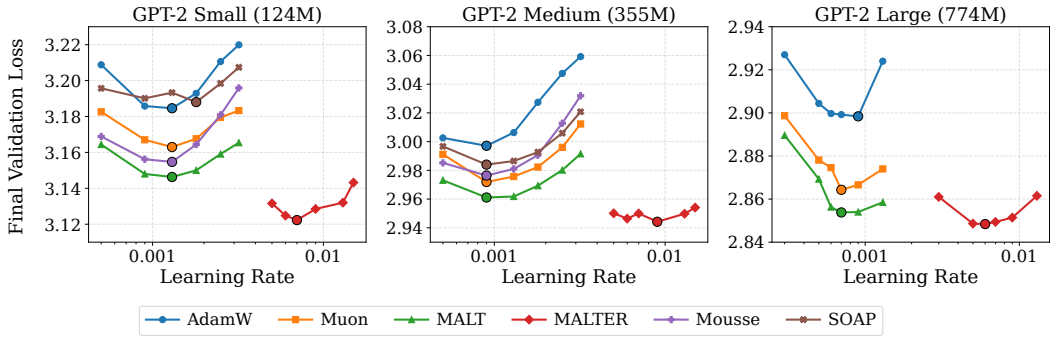


Figure 7: Final validation loss of pretraining GPT-2 Small, Medium and Large for 10K steps under different learning rates. The learning-rate grids are shown in Table 5. The best point for each optimizer is marked by a dot.

D.4 Learning-Rate Sweep Results for Ablation Variants

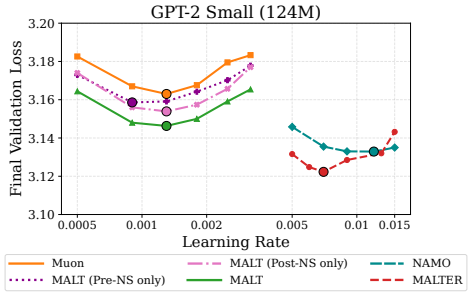


Figure 8: Final validation loss across learning rates for the methods compared in the GPT-2 Small ablation study after 10K training steps. The best point for each optimizer is marked by a dot.

D.5 Analysis of Diagonal Preconditioners

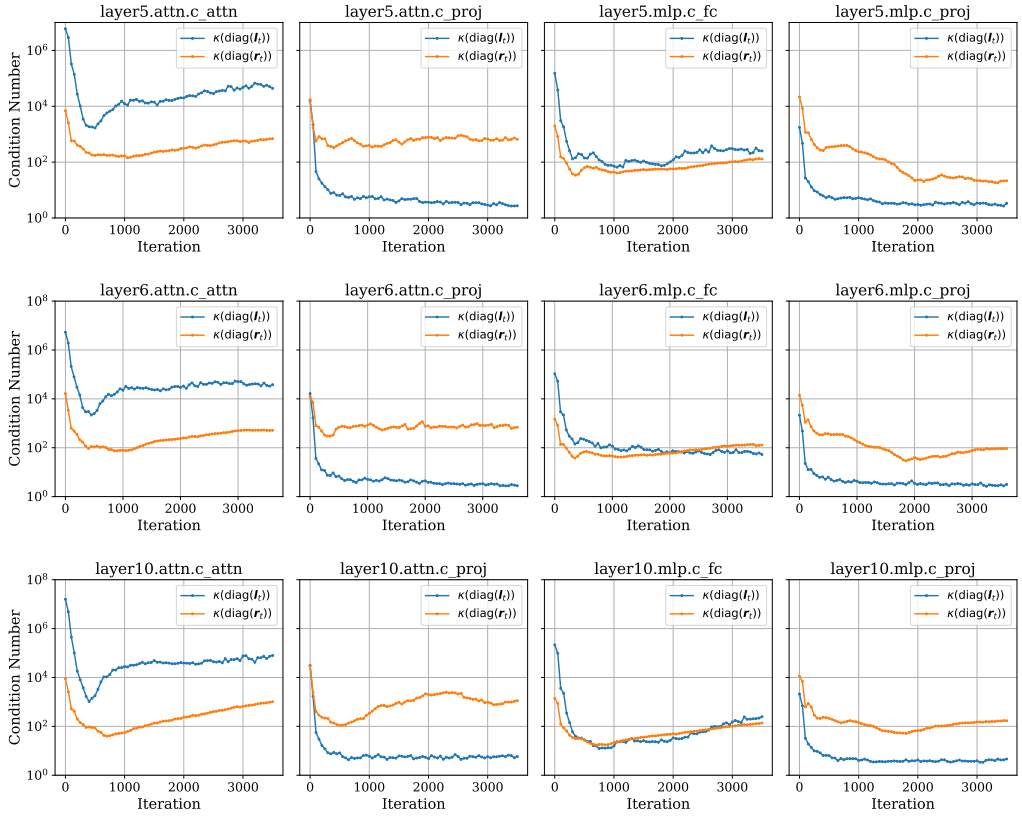


Figure 9: Condition numbers of the diagonal preconditioners in MALT for representative transformer layers. From top to bottom, we show Layer 0, Layer 5, and Layer 10. For each layer, we report the condition number of $\mathrm{diag}(\boldsymbol{l}_t)$ and $\mathrm{diag}(\boldsymbol{r}_t)$ for attention and MLP matrices.

We further analyze the left and right diagonal preconditioners of MALT. To verify that these preconditioners are not trivial scalings, we train a GPT-2 Medium model for 3500 steps using MALT, with a learning rate 0.0009, and collect the $\boldsymbol{l}_t$ and $\boldsymbol{r}_t$ every 50 steps.

In Fig. 9, we visualize the condition numbers of the left and right diagonal preconditioners for representative Transformer blocks. Here, `attn.c_attn` denotes the fused query-key-value projection, `attn.c_proj` denotes the output projection of the self-attention module, `mlp.c_fc` denotes the expansion layer in the feed-forward network, and `mlp.c_proj` denotes the corresponding down-projection layer. It can be seen that the condition numbers of preconditioners are often much larger than one across different layers and modules. This indicates that the diagonal preconditioners are non-trivial and capture substantial row- and column-wise anisotropy of the gradient during training.