

SWAN: SGD WITH NORMALIZATION AND WHITENING ENABLES STATELESS LLM TRAINING

Chao Ma*
Microsoft Research

Wenbo Gong*
Microsoft Research

Meyer Scetbon*
Microsoft Research

Edward Meeds
Microsoft Research

* These authors contributed equally to this work.

ABSTRACT

Adaptive optimizers such as Adam (Kingma & Ba, 2015) have been central to the success of large language models. However, they often require maintaining optimizer states throughout training, which can result in memory requirements several times greater than the model footprint. This overhead imposes constraints on scalability and computational efficiency. Stochastic Gradient Descent (SGD), in contrast, is a *stateless* optimizer, as it does not track state variables during training. Consequently, it achieves optimal memory efficiency. However, its capability in LLM training is limited (Zhao et al., 2024b). In this work, we show that pre-processing SGD using normalization and whitening in a stateless manner can achieve the same performance as the Adam optimizer for LLM training, while maintaining the same memory footprint of SGD. Specifically, we show that normalization stabilizes gradient distributions, and whitening counteracts the local curvature of the loss landscape. This results in SWAN (SGD with Whitening And Normalization), a stochastic optimizer that eliminates the need to store any optimizer states. Empirically, SWAN achieves $\approx 50\%$ reduction on total end-to-end memory compared to Adam. In language modeling tasks, SWAN demonstrates comparable or even better performance than Adam: when pre-training the LLaMA model with 350M and 1.3B parameters, SWAN achieves a 2 \times speedup by reaching the same evaluation perplexity using half as many tokens.

1 INTRODUCTION

Adaptive optimizers, such as Adam and its variants (Kingma & Ba, 2015; Loshchilov & Hutter, 2019; Shazeer & Stern, 2018; Pagliardini et al., 2024; Liu et al., 2023; Zhao et al., 2024a), have been central to the success of training large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023b; Dubey et al., 2024; Bi et al., 2024; Bai et al., 2023; Zhang et al., 2022). However, most adaptive optimizers for LLMs are *stateful*, meaning they require tracking and maintaining internal states. While achieving remarkable empirical success, these states introduce significant memory overhead. For instance, Adam (Kingma & Ba, 2015) – the de facto optimizer for LLM training – involves the tracking of exponential moving averages (EMAs), effectively doubling memory requirements. AdEMAMix (Pagliardini et al., 2024), an extension of Adam that achieves significant convergence speed boost, requires storing even more states, tripling the memory requirements. This overhead can be significant especially in distributed settings, where the optimizer states could consume a significant amount of the GPU memory (Dubey et al., 2024; Korthikanti et al., 2023). On the other hand, while stochastic gradient descent (SGD) is optimal in terms of memory efficiency (i.e., it is *stateless*), their capability to train LLMs is limited (Zhao et al., 2024b; Zhang et al., 2020; Kunstner et al., 2023; 2024). Therefore, a natural question arises:

Can LLMs be trained efficiently, without storing any optimizer states?

There is a growing body of research that has contributed to answering this question positively by developing novel optimizers that reduce the memory requirements associated with tracking internal

state variables during the training of LLMs while achieving similar or even speedup boost performance compared to Adam. For instance, some methods rely solely on tracking the first moment of gradients (Xu et al., 2024a; Jordan et al., 2024), while others introduce an additional one-dimensional tracking variable on top of first moments (Zhang et al., 2024b; Zhao et al., 2024c). Alternatively, approaches focusing exclusively on pre-conditioner tracking have also been proposed (Pooladzandi & Li, 2024; Li, 2017). Another line of work focuses on using low-rank approximations to store the first and second moments, thereby reducing the memory cost associated with tracking optimizer states (Lialin et al., 2023; Hao et al., 2024; Zhao et al., 2024a). Finally, the concurrent work of (Zhu et al., 2024) proposed an approximate gradient scaling method, that tracks the channel-wise or tensor-wise surrogate learning rates, further improving the memory efficiency.

In this work, we address this question by proposing to simply pre-process the instantaneous stochastic gradient in a stateless manner. The result is SWAN (SGD with Whitening And Normalization), a novel stochastic optimizer that eliminates *all* internal optimizer states and empirically achieves comparable or even better performance compared to Adam on several LLM pre-training tasks.

Our optimizer consists of combining two well-known operators to pre-process the raw gradients: GradNorm and GradWhitening. GradNorm applies a row-wise normalization on the gradient matrix, while GradWhitening orthogonalizes the normalized gradient matrix. We show that these operators aims at *stabilizing* the stochasticity of gradient distributions during training, and *neutralizing* the local geometry of the loss landscape, respectively. When applied together, both operators enables SWAN to rely solely on the statistics of the current gradient matrices. This approach eliminates the need to track state variables, thereby matching the memory footprint of SGD. In addition to memory savings, SWAN also demonstrates significant computational efficiency: our empirical evaluations on pre-training LLaMA (Touvron et al., 2023a) models on the C4 dataset with multiple model sizes show consistently the same or better performance than Adam and other low-rank optimizers. Remarkably, at the 350M and 1.3B scale, our method achieves up to 2X faster convergence in terms of tokens seen compared to Adam. Our contributions are summarized below:

- **A practical, stateless, adaptive optimizer.** SWAN (Algorithm 1), is a novel optimizer based on pre-processing instantaneous stochastic gradients with two stateless operators—GradNorm and GradWhitening (Figure 2). They perform gradient stabilization and loss landscape whitening, respectively, using information solely from the current gradient. SWAN offers:
 1. Memory efficiency: SWAN only requires the memory footprint of SGD (w/o momentum), that is: $\approx 50\%$ reduction on total memory, and $\approx 100\%$ reduction on optimizer states compared to Adam. This can be further reduced with LOMO technique (Lv et al., 2023) (Figure 1 (c)).
 2. Sample efficiency: through experiments (Section 6) on LLM pretraining tasks, we demonstrate that SWAN not only reduces memory overhead but also achieves similar or even better performance compared to Adam. Notably, SWAN achieves convergence speed-ups of over $2\times$ in terms of the number of tokens used for both 350M and 1.3B models (Figure 1).
 3. Computational efficiency: we propose a efficient scheme to accelerate the computation of SWAN (Section 4.2), achieving a similar throughput to Adam without the need for distributed computation (Section 6.2) required by recent optimizers such as shampoo Gupta et al. (2018).
 4. Robustness to hyperparameters: the hyperparameters of SWAN are lazily tuned via heuristics; and they are shared across all model sizes. Even when compared to tuned-baselines obtained with learning rate sweeps, SWAN still delivers strong performances on LLM pretraining. This is a promising indication of applicability to real-world problems.
- **Theoretical consistency with LLM dynamnics.** We show that (1) GradNorm can stabilize the heterogeneous covariance of LLM gradients, leveraging the redundancies in LLM gradient flows (Theorem 1 in Appendix C); and (2) GradWhitening can be derived as a non-diagonal second-order update under a specific structural assumption of the Hessian (Section 5.3). Additionally, we highlight that in the quadratic case, GradWhitening leads to convergence rates that are robust to the condition number of the local curvature (Theorem 2, Appendix C.1).

2 RELATED WORKS

Towards Stateless LLM Training. Adaptive optimizers generally rely on tracking internal state variables to perform weight updates, which can substantially increase memory consumption when

Table 1: The summary of memory consumption breakdown of optimizers, and the corresponding learning rate expressiveness. We assume all optimizers are applied on a 2D tensor of m by n ($m < n$). The optimizer state is defined as all intermediate variables maintained over time; while expressiveness is defined as the number of adaptive learning rates that a optimizer can model per tensor.

	Adam	SGD	SGD-Sal	Apollo (rank r)	Apollo (mini)	GaLore (rank r)	SWAN
Optimizer States \downarrow	$2mn$	0	mn	$2nr + 2 + (mr)^1$	$2n + 2 + (m)$	$2nr + mr$	0
Model weights \downarrow	mn	mn	mn	mn	mn	mn	mn
Expressiveness \uparrow	mn	1	blockwise	n	1	mn	mn

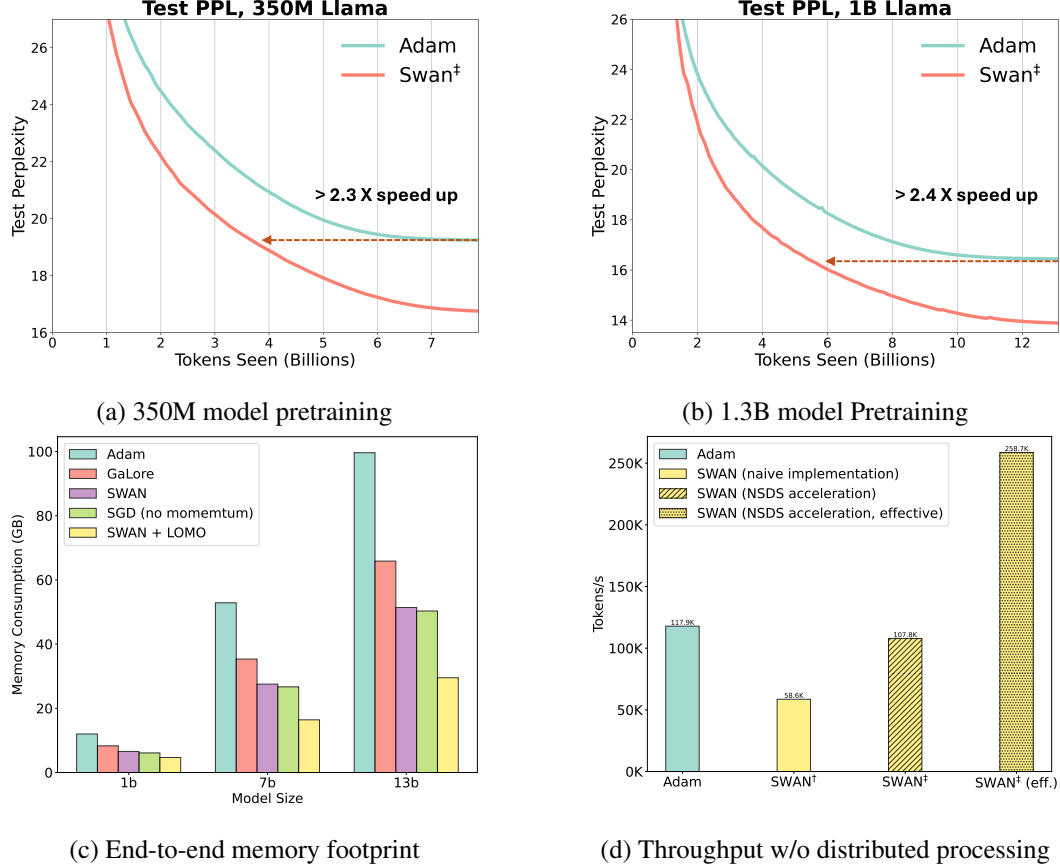


Figure 1: **SWAN Performance on LLM Pretraining.** (a) and (b): On both 350M and 1.3B LLaMa architectures, SWAN achieves over 2X speed-up compared to Adam in terms of tokens seen. (c): Memory footprint. We measure end-to-end memory usage during full-model training with a batch size of 1 sequence. SWAN achieves nearly 100% reduction in optimizer states memory, and 50% total memory reduction (up to 70% when combined with per-layer training technique Zhao et al. (2024a); Lv et al. (2023)), and (d): Training throughput of a 1.3B model without distributed gradient pre-processing. With our acceleration scheme (denoted by SWAN[‡]), we closely match the throughput of Adam without the need for distributed computation (Shi et al., 2023). The rightmost bar shows the effective throughput, adjusted for the token efficiency of the optimizer relative to Adam.

training large models. Several recent works have successfully managed to reduce the memory requirements associated with storing additional state variables for training LLMs. Muon (Jordan et al., 2024; Bernstein & Newhouse, 2024b), a newly proposed optimizer, has demonstrated strong acceleration and memory saving for LLM training by simplifying shampoo-like optimizers (Gupta et al., 2018; Anil et al., 2021; Shi et al., 2023; Wang et al., 2024; Vyas et al., 2024; Peirson et al., 2022;

¹The factor mr comes from the storage of random projection tensors. It can be reduced by only storing the random seeds; however, this might cause challenges in a distributed setting.

(Lin et al., 2024) and requiring only the tracking of a first-moment estimate. SGD-Sal (Xu et al., 2024b) only stores the first moment with a learning rate estimated at the beginning of training. Sign-based methods such as (Chen et al., 2023) have also demonstrated success on training transformer-based models by only tracking first moments. There are also several works that aim to enhance the memory efficiency of Adam by reducing the memory cost associated with second moments. Adam-mini (Zhang et al., 2024b) significantly reduces memory usage by storing only scalar values for each parameter block, while Adalayer (Zhao et al., 2024c) retains only the scalar average of the second moment for each layer. Alternatively, PSGD (Pooladzandi & Li, 2024; Li, 2017) focuses on exclusively tracking a pre-conditioner, eliminating the need to track a first moment estimate. Finally, (Zhu et al., 2024) proposes an optimizer based on approximate gradient scaling to train LLMs, which requires tracking channel-wise or tensor-wise approximation to element-wise adaptive learning rates. However, all the aforementioned optimizers still require the storage of state variables. In contrast, SWAN completely eliminates the need to store internal states for both the first and second moments by employing a combination of GradNorm and GradWhitening steps, which is discussed next.

Pre-processing Gradients. Gradient pre-processing is a common technique to improve optimizer performance. Various pre-processing procedures have been proposed in the literature, such as signed gradient (Bernstein et al., 2018; Crawshaw et al., 2022; Chen et al., 2023; Kunstner et al., 2023), gradient clipping (Zhang et al., 2020), normalization (Zhang et al., 2020; You et al., 2019), and whitening (Yang & Laaksonen, 2008; Kingma & Ba, 2015; Hwang, 2024; Jordan et al., 2024; Bernstein & Newhouse, 2024c;a; Carlson et al., 2015). In this work, we focus on normalization and whitening. We apply normalization row-wise on gradient matrices, together with gradient whitening under a specific structural assumption of the Fisher Information (Hwang, 2024; Martens et al., 2018), recovering the orthogonalization step used in (Jordan et al., 2024; Tuddenham et al., 2022). Our key result is that *composing* normalization and whitening on stochastic gradients can be sufficient to enable the efficient training of LLMs in a stateless manner. In our setting, we show empirically that both removing any one of these two pre-processing steps from SWAN results in significant performance degradation (Section 6.4). Compared to Lamb (You et al., 2019), our normalization operation is applied on raw gradients row-wise; while LAMB applied layer-wise global normalization on Adam states instead. Compared to Muon (Jordan et al., 2024), SWAN removes first-moment tracking and instead uses row-wise normalization, making the optimizer fully statless. Moreover, our proposed efficient heuristic scheme for computing square root inverse requires $\mathcal{O}(m^2)$ instead of $\mathcal{O}(m^3)$ of the standard NS scheme (Song et al., 2022; Li et al., 2018; Huang et al., 2019).

Low-rank methods. Low-rank optimization techniques have been explored in the context of large language model (LLM) training as a means to reduce memory consumption. These methods focus on applying low-rank approximations to model weights, gradients, and/or optimizer state variables. A seminal work in this domain is LORA (Hu et al., 2021) to fine-tune pre-trained models using additional low-rank weight matrices at each layer, thereby significantly reducing memory usage to update the weights. More recently, methods such as ReLoRA (Lialin et al., 2023), FLORA (Hao et al., 2024), and Galore (Zhao et al., 2024a) have advanced low-rank optimization techniques for memory-efficient LLM pre-training. These approaches leverage low-rank gradient projections to enable full-rank learning, thereby achieving memory savings without compromising model capacity. Notably, they have achieved substantial reductions in the memory consumption of optimizer states, with only minimal impact on model performance. While these approaches effectively reduce the memory footprint of LLM training, they still necessitate storing internal states, resulting in higher memory consumption compared to SWAN.

3 PRELIMINARIES

Adam (Kingma & Ba, 2015) is the current standard choice for adaptive optimizers across a multitude of machine learning training tasks, including LLM pre-training. Adam is an example of a *stateful* optimizer, which means it accumulates and stores internal states throughout training. It combines the advantages of two earlier methods: AdaGrad (Duchi et al., 2011), which adapts learning rates based on the historical gradients’ magnitudes, and RMSProp (Tieleman, 2012), which mitigates the aggressive decrease in learning rates by using a decaying average of squared gradients.

Consider a loss function $\mathcal{L}_{\mathbf{W}} : \mathcal{X} \rightarrow \mathbb{R}$, parameterized by weight matrices $\mathbf{W} \in \mathbb{R}^{m \times n}$, and denote $\mathbf{x}^{(t)}$ a mini-batch of inputs provided at the t -th training step that is sampled from data distribution $p_{\text{data}}(\mathbf{x})$. Let $\mathbf{G}^{(t)}$ be the stochastic gradient of $\mathcal{L}_{\mathbf{W}}$ (i.e., a random variable induced by sampling $\mathbf{x}^{(t)}$). Then, Adam can be broken down into the following steps:

$$\begin{aligned} \mathbf{G}^{(t)} &= \nabla_{\mathbf{W}} \mathcal{L}_{\mathbf{W}}(\mathbf{x}^{(t)}), \quad \mathbf{x}^{(t)} \sim p_{\text{data}}(\mathbf{x}) && \text{(stochastic gradient)} \\ \mathbf{m}^{(t)} &= \beta_1 \mathbf{m}^{(t-1)} + (1 - \beta_1) \mathbf{G}^{(t)}, && \text{(EMA first moment)} \\ \boldsymbol{\nu}^{(t)} &= \beta_2 \boldsymbol{\nu}^{(t-1)} + (1 - \beta_2) \mathbf{G}^{(t)^2}, && \text{(EMA second moment)} \\ \mathbf{W}^{(t+1)} &= \mathbf{W}^{(t)} - \eta \left(\frac{\hat{\mathbf{m}}^{(t)}}{\sqrt{\hat{\boldsymbol{\nu}}^{(t)} + \epsilon}} \right) && \text{(weight update)} \end{aligned}$$

where $\mathbf{m}^{(t)}$ and $\boldsymbol{\nu}^{(t)}$ are EMAs of the first and second moments of the gradients; and η is a global step size. Intuitively, Adam estimates the signal-to-noise (SNR) ratios and use it to adjust learning rates element-wise. Tracking and storing these two EMA estimates triple the total memory consumption required to train a LLM model. For example, for LLaMA 405B model, storing model weights requires 810 GB of memory, while the Adam optimizer states requires an additional 1.6TB of memory.

Desired properties of Adam. There is a rich literature on understanding adaptive methods' inner workings and unreasonable effectiveness. Notably, the key desired properties of Adam can be categorized as *gradient whitening*, *gradient smoothing*, and *gradient invariance*.

- *Gradient whitening*: it is known that the inverse second moment $\frac{1}{\sqrt{\hat{\boldsymbol{\nu}}^{(t)} + \epsilon}}$ of Adam performs gradient whitening by approximating the square root inverse of the diagonal of Fisher information matrix (Kingma & Ba, 2015; Hwang, 2024). This step biases the optimization trajectories towards well-conditioned regions (Jiang et al., 2024) and provides a better approximation to the geodesic flow when compared with the natural gradient update (Yang & Laaksonen, 2008).
- *Gradient smoothing*: the EMA operations in Adam naturally reduce the influence of mini-batch noise (Cutkosky & Mehta, 2020; Crawshaw et al., 2022);
- *Gradient invariance*: recent work suggest the performance gap between SGD and Adam might lie in Adam's *sign-descent*-like nature (Bernstein et al., 2018; Crawshaw et al., 2022; Chen et al., 2023). For example, Adam is robust to the rescaling of gradient diagonals (Kingma & Ba, 2015); and is invariant to any sign-preserving scalings (under $\beta_1 = \beta_2 = 0$) (Bernstein et al., 2018).

For a more comprehensive discussion of these properties, please refer to appendix A.

Adam as SGD pre-processing. Adam can be viewed as history-dependent pre-processing of the gradients ($\{\mathbf{G}^{(0)}, \mathbf{G}^{(1)}, \dots, \mathbf{G}^{(t)}\} \rightarrow \frac{\hat{\mathbf{m}}^{(t)}}{\sqrt{\hat{\boldsymbol{\nu}}^{(t)} + \epsilon}}$) to achieve the desired properties described above. A key observation is that all of these properties are achieved through element-wise operations, where each element of the gradient matrix is independently pre-processed and re-scaled. This approach does not take into account the interactions and structures between different variables, and we hypothesize that this is the reason why additional history information is necessary to bridge this gap, ultimately leading to the requirement for EMA states. Thus, we believe that designing stateless adaptive optimizers is possible if we can achieve similar properties by applying matrix-level operations that pre-process the instantaneous stochastic gradients of SGD ($\mathbf{G}^{(t)} \rightarrow \hat{\mathbf{G}}^{(t)}$).

4 THE SWAN OPTIMIZER: PREPROCESSING SGD WITH NORMALIZATION AND WHITENING

As discussed in Section 3, we believe the key to designing stateless, adaptive, and effective optimizers lies in the incorporation of *matrix-level* operations that exploit rich information contained in the gradient matrix. To this end, we compose two well-known matrix operators, namely normalization and whitening. When applied in tandem, they achieve similar desirable properties of adaptive optimizers, without the need to store historical gradient moments. The result is SWAN (SGD with Whitening And Normalization), a new stateless optimizer which we describe next.

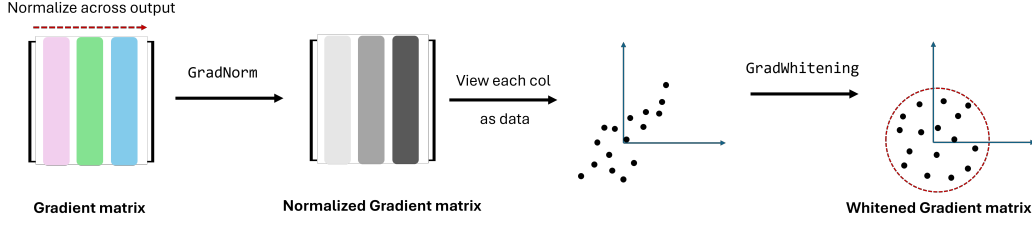


Figure 2: Illustration of GradNorm and GradWhitening operators. In GradNorm operator, we perform standardization across the output dimensions (columns), using statistics computed row-wise. In GradWhitening operator (illustration adapted from Huang et al. (2019)), we treat each column of the gradient matrix \mathbf{G} as a separate data sample. Then, GradWhitening can be seen as stretching/squeezing the data such that the covariance matrix is the identity across all eigen directions.

Algorithm 1 SWAN Optimizer

Input: weight matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$ with $m \leq n$. Step size η . Number of GradWhitening iteration k (default = 10).
 Initialize step $t \leftarrow 0$
repeat
 $\mathbf{G}^{(t)} \in \mathbb{R}^{m \times n} \leftarrow \nabla_{\mathbf{W}^{(t)}} \mathcal{L}^{(t)}(\mathbf{W}^{(t)})$
 $\tilde{\mathbf{G}}^{(t)} \leftarrow \text{GradNorm}(\mathbf{G}^{(t)})$ (Eq (1))
 $\Delta \mathbf{W}^{(t)} \leftarrow \text{GradWhitening}(\tilde{\mathbf{G}}^{(t)}, k)$
 (optional) $\Delta \mathbf{W}^{(t)} \leftarrow \frac{\sqrt{mn} \Delta \mathbf{W}^{(t)}}{\|\Delta \mathbf{W}^{(t)}\|}$
 $\mathbf{W}^{(t)} \leftarrow \mathbf{W}^{(t-1)} - \eta \Delta \mathbf{W}^{(t-1)}$
 $t \leftarrow t + 1$
until convergence criteria met
return $\mathbf{W}^{(t)}$

Algorithm 2 GradWhitening Operator

Input: $\mathbf{G}^{m \times n}$ with $m \leq n$. Number of iterations K . Step size β . Boolean `diag` indicating if use diagonal substitution scheme.
 Initialize $\mathbf{Y} \leftarrow \mathbf{G}\mathbf{G}^\top$, $\mathbf{Z} \leftarrow \mathbf{I}$
for $i = 1, \dots, K$ **do**
 if `diag` **then**
 $\mathbf{Y} \leftarrow \beta \mathbf{Y} \text{Diag}(\mathbf{3I} - \mathbf{Z} \text{Diag}(\mathbf{Y}))$,
 $\mathbf{Z} \leftarrow \beta (\mathbf{3I} - \text{Diag}(\mathbf{Z}) \mathbf{Y}) \text{Diag}(\mathbf{Z})$
 else
 $\mathbf{Y} \leftarrow \beta \mathbf{Y} (\mathbf{3I} - \mathbf{Z} \mathbf{Y})$,
 $\mathbf{Z} \leftarrow \beta (\mathbf{3I} - \mathbf{Z} \mathbf{Y}) \mathbf{Z}$
 end if
end for
return \mathbf{ZG}

Figure 3: SWAN Optimizer. When `diag` is set to True, we cover the fast variant denoted by SWAN[‡].

4.1 SWAN UPDATE RULES

In SWAN (Algorithm 1), the raw SGD gradient \mathbf{G}^t is processed by the operations below²:

$$\begin{cases} \tilde{\mathbf{G}}^{(t)} \leftarrow \text{GradNorm}(\mathbf{G}^{(t)}) \\ \Delta \mathbf{W}^{(t)} \leftarrow \text{GradWhitening}(\tilde{\mathbf{G}}^{(t)}) \end{cases} \quad (\text{SWAN})$$

The weight is then updated by $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \Delta \mathbf{W}^{(t)}$. The GradNorm operator (Equation (1)) denotes the normalization of the gradient matrix row-wise (Figure 2); and the GradWhitening operator denotes the whitening of the gradients (Figure 2). Both operators have been extensively applied in different contexts of optimization and/or architecture. Related applications include the processing of neural network forward pass activations (in the form of RMS layer norm (Zhang & Sennrich, 2019) and decorrelated batch norm (Huang et al., 2018; Song et al., 2022)), as well as the processing of backward gradients (Tuddenham et al., 2022; Jackson, 2023; Jordan et al., 2024).

On the GradNorm Step. Consider the gradient matrix $\mathbf{G} \in \mathbb{R}^{m \times n}$ with rows and columns corresponding to input and output dimensions, respectively, of some block of model parameters. Let $1 \leq i \leq m$ represent the input indices and $1 \leq j \leq n$ represent the output indices. Instead of performing element-wise EMA to stabilize and normalize the noisy gradients, as done in Adam, we

²Here \mathbf{G} is assumed to be the gradient matrix for some parameter block in the model (e.g. a linear layer); enabling us to take advantage of the matrix structure to achieve our smoothing and whitening goals.

propose to standardize across the output dimensions at each time step t :

$$\text{GradNorm}(\mathbf{G}) := \frac{\mathbf{G}}{s\mathbf{1}_n^\top} \quad (1)$$

where $s := \sqrt{\frac{1}{n} \sum_{j=1}^n (\mathbf{G}_{:,j})^2}$ is m -dimensional the root mean square across dimension (m -dimensional column vector); $\mathbf{1}_n$ is a n -dimensional column vector of ones.

GradNorm is the forward pass operator RMS-Norm (Zhang & Sennrich, 2019) applied on backward gradients. GradNorm allows the optimizer to be invariant under matrix-wise and row-wise rescaling. In Section 5 we show that this is the key to stabilizing the LLM gradient distribution. Meanwhile, compared to the sign operation ($\text{sign}(\mathbf{G})$) and matrix-wise normalization ($\frac{\mathbf{G}}{\|\mathbf{G}\|}$) alternatives (Zhang et al., 2020; You et al., 2019), GradNorm preserves richer information of gradient scaling while offering invariance properties.

On the GradWhitening Step. As discussed in Section 3, Adam relies on a second moment estimate to perform *element-wise* gradient whitening. More formally, its second moment estimate a diagonal approximation to the Fisher information matrix (FIM): (Kingma & Ba, 2015; Hwang, 2024):

$$\mathbb{E}(\text{Vec}(\mathbf{G})\text{Vec}(\mathbf{G})^\top) \approx \text{Diag}[\text{Vec}[\mathbb{E}(\mathbf{G}^{\odot 2})]]$$

where $\text{Vec}(\cdot)$ denotes the vectorized operation, $\text{Diag}(\cdot)$ denotes the operation that produces a diagonal matrix given a vector, and \odot denotes the Hadamard product. Hence, whitening using the inverse square root of this diagonal FIM is equivalent to element-wise rescaling with $\frac{1}{\sqrt{\mathbb{E}(\mathbf{G}^{\odot 2})}}$. Here, we consider the following non-diagonal approximation:

$$\mathbb{E}(\text{Vec}(\mathbf{G})\text{Vec}(\mathbf{G})^\top) \approx \mathbf{I}_{n \times n} \otimes \mathbf{G}\mathbf{G}^\top,$$

where \otimes denotes Kronecker product. This leads to the following whitening step:

$$\text{GradWhitening}(\mathbf{G}) := (\mathbf{G}\mathbf{G}^\top)^{-1/2} \mathbf{G} \quad (2)$$

where the exponent $-1/2$ stands for matrix inverse square root. $(\mathbf{G}\mathbf{G}^\top)^{-1/2} \mathbf{G}$ is simply the orthogonalization of \mathbf{G} , i.e., the closest orthogonal matrix to \mathbf{G} (w.r.t the Frobenius norm). The derivation of Equation (2) as structured FIM/Hessian approximation is discussed in Section 5.3. Apart from the FIM/Hessian approximation view, Equation (2) can also be interpreted as *minimizing the condition number of \mathbf{G}* (defined as the ratio between the largest and smallest singular values). It can be shown that after whitening, its condition number achieves the minimum value which is 1.

Similar to GradNorm, GradWhitening (Equation (2)) as a matrix operation has been widely used as a forward-pass operator in the form of decorrelated batch normalization (Huang et al., 2018; Li et al., 2018); and it has also shown great success in processing backward gradients (Tuddenham et al., 2022; Jordan et al., 2024; Gupta et al., 2018). As shown in Figure 2, by treating each column of \mathbf{G} as i.i.d. vector-valued data samples $\mathbf{G} = \{\mathbf{g}_1, \dots, \mathbf{g}_j, \dots, \mathbf{g}_n\}$, GradWhitening can be seen as effectively stretching/squeezing this data matrix along the eigenvectors to whiten its covariance matrix. This essentially forces the gradients to *traverse all eigen-directions at the same rate*.

4.2 PRACTICAL CONSIDERATIONS: FAST AND STABLE IMPLEMENTATION

Fast GradWhitening Computing GradWhitening exactly can be expensive, as it involves solving the matrix square-root inverse. One option is to directly apply the Newton-Schulz variant of decorrelated batch normalization (Song et al., 2022; Li et al., 2018; Huang et al., 2019), which allows a more GPU-friendly estimation. This is given by (Song et al., 2022; Li et al., 2018):

$$\begin{cases} \mathbf{Y}_{k+1} = \frac{1}{2} \mathbf{Y}_k (3\mathbf{I} - \mathbf{Z}_k \mathbf{Y}_k) \\ \mathbf{Z}_{k+1} = \frac{1}{2} (3\mathbf{I} - \mathbf{Z}_k \mathbf{Y}_k) \mathbf{Z}_k \end{cases} \quad (3)$$

where $\mathbf{Y}_0 = \mathbf{G}\mathbf{G}^\top$, $\mathbf{Z}_0 = \mathbf{I}$. At convergence, $\text{GradWhitening}(\mathbf{G}) = \mathbf{Z}\mathbf{G}$ (Algorithm 2). See Appendix J for implementation details. Note that the same N-S procedure has been discussed in (Mei et al., 2023) and (Jackson, 2023) for preconditioned optimizers. Another N-S procedure was been introduced in (Bernstein & Newhouse, 2024b), and optimized in (Jordan et al., 2024).

However, estimating $(\mathbf{G}\mathbf{G}^\top)^{-1/2}$ with NS requires $\mathcal{O}(m^3)$ (assuming $m < n$) complexity, making its scalability for larger models, especially under distributed settings unclear [Jordan et al. \(2024\)](#). Here, we propose a heuristic scheme that has $\mathcal{O}(m^2)$ complexity to estimate square-root inverse (comparable to the $\mathcal{O}(mn)$ complexity of Adam second moment):

$$\begin{cases} \mathbf{Y}_{k+1} = \frac{1}{2}\mathbf{Y}_k \text{Diag}(3\mathbf{I} - \mathbf{Z}_k \text{Diag}(\mathbf{Y}_k)) \\ \mathbf{Z}_{k+1} = \frac{1}{2}(3\mathbf{I} - \text{Diag}(\mathbf{Z}_k)\mathbf{Y}_k) \text{Diag}(\mathbf{Z}_k) \end{cases} \quad (4)$$

where under a slight abuse of notation, $\text{Diag}(\cdot)$ returns a diagonal matrix that has the same diagonals as the input matrix. Intuitively, whenever we encounter matrix multiplication in NS iterations, we replace one of the matrices with its diagonal approximation. The particular choice of where to apply $\text{Diag}(\cdot)$ is determined by performance on synthetic datasets. We refer to this as the *NS with diagonal substitution* (NSDS) scheme. In Appendix B, we empirically demonstrate that NSDS performs well in minimizing the matrix condition number of gradients. As shown in Figure 7, on the gradient distribution induced by Llama model training, the performance of NSDS (when combined with GradNorm) is comparable to or better than the standard NS scheme in reducing the condition number.

We found that NSDS does have the side effect of slowing down the early stage convergence. However, it offers stronger tail convergence in return. For LLM pretraining NSDS helps SWAN achieve similar or better final validation loss performance than the original NS scheme (Section 6.1), enabling Adam-level training throughput without the help of distributed computation (Section 6.2).

Rescaling for robustness. The operator GradWhitening maps the normalized gradient $\tilde{\mathbf{G}}^{(t)}$ onto the closest orthogonal matrix, and as such might drastically change its effective learning rate. In practice, we propose the following re-scaling before updating the weights:

$$\Delta \mathbf{W}^{(t)} \leftarrow \frac{\|\tilde{\mathbf{G}}^{(t)}\| \Delta \mathbf{W}^{(t)}}{\|\Delta \mathbf{W}^{(t)}\|} = \frac{\sqrt{mn} \Delta \mathbf{W}^{(t)}}{\|\Delta \mathbf{W}^{(t)}\|} \quad (5)$$

This helps rescale the norm of the whitened gradient back to the norm of $\tilde{\mathbf{G}}^{(t)}$, that is $\|\tilde{\mathbf{G}}^{(t)}\| = \sqrt{mn}$. Notice that this is exactly the norm of signed gradient descent (= Adam without EMAs) ([Bernstein et al., 2018](#)). This relationship allows us to directly adapt the learning rates from Adam without tuning specifically for SWAN. This can be thought as a variation of LR grafting ([Agarwal et al.](#)).

Finally, the practical SWAN algorithm (GradNorm + NSDS-GradWhitening + rescaling) is presented in Algorithm 1 (under `diag = True`). We denote it as **SWAN[‡]**.

4.3 SAVING ANALYSIS: BALANCING MEMORY EFFICIENCY AND EXPRESSIVENESS

Table 1 shows the theoretical memory savings of SWAN. When applied on a 2D tensor of m by n ($m > n$), SWAN requires storing 0 optimizer states (i.e., intermediate variables maintained over time), compared with other state-of-the-art memory efficient optimizers. Notably, this is achieved without significantly compromising its capabilities to model the adaptive learning rates. SWAN can model *element-wise* adaptive learning rates, while the concurrent work (SGD-sal and Apollo) needs to resort to block-wise or even tensor-wise approximations. In our experiments in Section 6.1, we show that these approximations indeed has a negative impact on LLM training performance.

5 ANALYSIS: A LLM LEARNING DYNAMICS PERSPECTIVE

As a new stateless adaptive optimizer, the complete theoretical properties of SWAN is an open question which we leave for future work. However, as a first analysis, we consider SWAN from a *learning dynamics* perspective, specifically the dynamics of an LLM based upon a simplified transformer block. It is this analysis that led to the design choices for SWAN.

5.1 SETUP

We consider the simplified transformer block (STB) architecture proposed in ([Tian et al., 2023](#)).

Definition 1 (Simplified Transformer Block (STB)). *Given the input activation $\mathbf{x} \in \mathbb{R}^{M_C \times 1}$, query token index q , context embedding matrix $\mathbf{U}_C \in \mathbb{R}^{d \times M_C}$, and the query embedding $\mathbf{u}_q \in \mathbb{R}^{d \times 1}$,*

the STB computes the output $\mathbf{h} \in \mathbb{R}^{n \times 1}$ as $\mathbf{h} = \phi(\mathbf{W}^\top (\mathbf{U}_C (\exp(\mathbf{z}_q) \odot \mathbf{x}) + \mathbf{u}_q))$, where M_C is the context length, the attention logits $\mathbf{z}_q \in \mathbb{R}^{M_C \times 1}$ are given by $z_{ql} = \mathbf{u}_q^\top \mathbf{W}_Q^\top \mathbf{W}_K \mathbf{u}_l$, with $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d \times d}$ being weight matrices for the queries and keys, respectively, $\mathbf{W} \in \mathbb{R}^{d \times n}$ is the weight matrix for the feedforward network, and ϕ is a nonlinearity function such as the ReLU.

Given a STB, we consider a loss function $\mathcal{L}_{\mathbf{W}, \mathbf{z}}(\mathbf{x}^{(t)})$, where $\mathbf{x}^{(t)}$ is a mini-batch of inputs provided at the t -th training step sampled from data distribution $p_{\text{data}}(\mathbf{x})$. Standard mini-batch learning dynamics is then given by

$$\dot{\mathbf{W}}^{(t)} = \frac{\partial \mathcal{L}_{\mathbf{W}, \mathbf{z}_q}(\mathbf{x}^{(t)})}{\partial \mathbf{W}}, \quad \dot{\mathbf{z}}_q^{(t)} = \frac{\partial \mathcal{L}_{\mathbf{W}, \mathbf{z}_q}(\mathbf{x}^{(t)})}{\partial \mathbf{z}_q}.$$

In this case, both $\dot{\mathbf{W}}^{(t)}$ and $\dot{\mathbf{z}}_q$ are viewed as random variables induced by random mini-batch $\mathbf{x}^{(t)}$. For example, for each row i , $\dot{\mathbf{W}}^{(t)}[i, :]$ can be re-written as $\dot{\mathbf{W}}^{(t)}[i, :] = \mathbb{E}[\dot{\mathbf{W}}^{(t)}[i, :]] + \boldsymbol{\varepsilon}_{\mathbf{W}}^{(t)}[i, :]$, where $\boldsymbol{\varepsilon}^{(t)}[i, :]$ is zero mean random variable with covariance $\text{Cov}[\dot{\mathbf{W}}^{(t)}[i, :]]$.

5.2 GradNorm: STABILIZING GRADIENT DISTRIBUTIONS OF STB

Below we show that, based on the dynamics of the STB, GradNorm stabilizes $\boldsymbol{\varepsilon}_{\mathbf{W}}^{(t)}$.

Theorem 1 (GradNorm stabilizes gradient distributions across time for the STB). *Consider the STB (Definition 1). Assuming we inherit the assumptions in Theorem 1 of Tian et al. (2023), as described in Appendix C. Then consider $\mathbf{U}_C^\top \mathbf{W}$, the composition of the MLP project-up matrix and the embedding matrix as a whole. Then, its standardized stochastic gradients $\tilde{\mathbf{G}}_{\mathbf{U}_C^\top \mathbf{W}}^{(t)} := \text{GradNorm}(\frac{\partial \mathcal{L}_{\mathbf{W}, \mathbf{z}}(\mathbf{x}^{(t)})}{\partial \mathbf{U}_C^\top \mathbf{W}})$ satisfy:*

$$\text{Cov}[\tilde{\mathbf{G}}_{\mathbf{U}_C^\top \mathbf{W}}^{(t_1)}[i, :]] = \text{Cov}[\tilde{\mathbf{G}}_{\mathbf{U}_C^\top \mathbf{W}}^{(t_2)}[i, :]] \quad \text{for all } t_1, t_2, \text{ and } i.$$

In other words, the covariance structure of $\tilde{\mathbf{G}}$ is identical across all time steps t , achieving distributional stability across time. The same relationship also holds for the gradient of attention score $\tilde{\mathbf{G}}_{\mathbf{z}_q}^{(t)} := \text{GradNorm}(\frac{\partial \mathcal{L}_{\mathbf{W}, \mathbf{z}_q}(\mathbf{x}^{(t)})}{\partial \mathbf{z}_q})$.

Theorem 1 suggests that GradNorm implicitly aligns with the dynamics of transformer architectures and removes the time-heterogeneity in gradient covariance structures.

5.3 GradWhitening: AN EFFICIENT NON-DIAGONAL SECOND-ORDER UPDATE

Here, we show that GradWhitening is equivalent to a non-diagonal second-order method under a specific Kronecker factorization assumption of the Hessian/FIM. The assumption is as below:

Assumption 1 (Assumption of GradWhitening). *At time t , the local Hessian \mathbf{H} of the loss has shared block-diagonal structure, such that $\mathbf{H} = \mathbf{I}_{n \times n} \otimes \tilde{\mathbf{H}}$, where $\tilde{\mathbf{H}} \in \mathbb{R}^{m \times m}$.*

Approximating Hessian with a Kronecker factorization is not new and has been extensively studied in the literature (Martens & Grosse, 2015; George et al., 2018; Gao et al., 2023; 2021; Koroko et al., 2022; Eschenhagen et al., 2024; Gupta et al., 2018). Here, this specific structure is useful in our context as 1) it enables estimation of Hessian/FIM without temporal EMAs, nor mini-batch statistics, and 2) it aligns with the statistical property of STB, as shown later in Proposition 1.

By leveraging assumption 1, we can now effectively estimate \mathbf{H} by only using one single gradient matrix sample $\mathbf{G} := [\mathbf{g}_1, \dots, \mathbf{g}_n] \in \mathbb{R}^{m \times n}$. Recall that the Fisher information formulation of Hessian is defined as $\mathbf{H} = \mathbb{E}[\text{Vec}(\mathbf{G})\text{Vec}(\mathbf{G})^\top]$ where $\text{Vec}(\cdot)$ denotes the vectorized operation. Under assumption 1, we can estimate $\mathbf{H} = \mathbf{I}_{n \times n} \otimes \tilde{\mathbf{H}}$ by computing the following simple estimate $\frac{1}{n} \sum_{i=1}^n \mathbf{g}_i \mathbf{g}_i^\top = \mathbf{G} \mathbf{G}^\top$, which approximate the current $\tilde{\mathbf{H}}$. Hence, GradWhitening can be seen as applying a second order update under our structural assumption:

$$\text{GradWhitening}(\mathbf{G}) = (\mathbf{G} \mathbf{G}^\top)^{-\frac{1}{2}} \mathbf{G} = \tilde{\mathbf{H}}^{-\frac{1}{2}} \mathbf{G}$$

In the following Proposition, we show that the assumption 1 of GradWhitening aligns with the equilibrium Hessian structure in the STB regime.

Proposition 1 (Shared structures in the block-diagonal of Hessians at transformer equilibrium). Consider a STB (1), trained with full-batch gradient descent. Next, assume we inherit all the assumptions from Theorem 1 of Tian et al. (2023). Then, as $t \rightarrow \infty$, we have the following shared Hessian structure along the diagonal blocks:

$$\frac{\mathbf{H}_{sk,s'k}}{\sum_{s,s'} \mathbf{H}_{sk,s'k}} \rightarrow \frac{\mathbf{H}_{sk',s'k'}}{\sum_{s,s'} \mathbf{H}_{sk',s'k'}}, \quad \forall 1 \leq s, s' \leq d, 1 \leq k, k' \leq n \quad (6)$$

Where $\mathbf{H}(\mathbf{W})_{sk,s'k'} = \frac{\partial \mathcal{L}}{\partial w_{sk} \partial w_{s'k'}}$.

Proposition 1 shows that, under a simplified setting of the transformer, the Hessian will also converge to an equilibrium solution where the $M_C \times M_C$ blocks over the diagonal direction of Hessian shares an identical structure, which supports the assumption of GradWhitening. This result is verified in our numerical experiment (Appendix, Figure 8). Finally, Theorem 2 presented in Appendix will also show how GradWhitening helps to make the convergence rate of SGD more robust to the condition number of local curvatures, and outperforms both SGD and Adam in the ill-conditioned regime.

5.4 VERIFYING THE THEORETICAL INSIGHTS

We empirically validate the theoretical benefits of GradNorm and GradWhitening. As detailed in Appendix I, we demonstrate that GradNorm effectively stabilizes stochastic gradient distributions during SGD training of a scaled-down LLaMA model on the C4 dataset, evidenced by significantly reduced KL divergence fluctuations compared to standard training. Additionally, GradWhitening significantly enhances optimization performance across high-dimensional and ill-conditioned classic optimization functions, consistently outperforming both SGD and Adam. These findings confirm that GradNorm promotes gradient stability and GradWhitening addresses local curvature challenges, enabling faster and more reliable performance.

6 EXPERIMENTS

In this section, we report empirical results for SWAN. All experiments run on NVIDIA A100 GPUs.

6.1 MEMORY-EFFICIENT PRE-TRAINING OF LLAMA MODELS

Setup We evaluate SWAN on memory-efficient Llama (Touvron et al., 2023a) pre-training tasks, using the standardized settings of (Zhao et al., 2024a), which has been adopted by many recent works (Zhao et al., 2024a; Chen et al., 2024; Zhang et al., 2024c; Huang et al.; Zhu et al., 2024). We consider models with 60M, 130M, 350M, and 1.3B parameters, all trained on the C4 dataset Raffel et al. (2020) using an effective batch size of 130K tokens. Following the setup of Zhao et al. (2024a), SWAN is applied to all linear modules in both attention and MLP blocks. Training uses BF16 by default unless specified, see Appendix J. The other evaluation settings follows Zhao et al. (2024a).

Table 2: Comparison with Adam and its memory-efficient low-rank variants on pre-training various sizes of LLaMA models on C4 dataset. Validation perplexity is reported, along with a memory estimate of the total of parameters and optimizer states based on BF16 format. Baseline results are directly taken from the official numbers reported in [Zhao et al. \(2024a\)](#); [Zhu et al. \(2024\)](#), as they shares exactly the same setup. Note that for Adam we report both the official results from [Zhao et al. \(2024a\)](#) and our reproduced result.

	60M	130M	350M	1.3 B
Adam	33.02 (0.32G)	24.44 (0.75G)	19.24 (2.05G)	16.44 (7.48G)
Adam (cited)	34.06 (0.32G)	25.08 (0.75G)	18.80 (2.05G)	15.56 (7.48G)
SWAN [‡]	30.59 (0.23G)	22.61 (0.43G)	16.63 (0.93G)	13.56 (2.98G)
SWAN [†]	30.00 (0.23G)	22.83 (0.43G)	17.14 (0.93G)	14.42 (2.98G)
SWAN-0	32.28 (0.23G)	24.13 (0.43G)	18.22 (0.93G)	15.13 (2.98G)
Momentum+GradWhitening	31.6 (0.27G)	24.59 (0.59G)	19.30 (1.49G)	16.08 (5.23G)
Galore	34.88 (0.26G)	25.36 (0.57G)	18.95 (1.29G)	15.64 (4.43G)
Apollo-mini	31.93 (0.23G)	23.53 (0.43G)	17.18 (0.93G)	14.17 (2.98G)
Apollo	31.55 (0.26G)	22.94 (0.57G)	16.85 (1.29G)	14.20 (4.43G)
Low-Rank	78.18 (0.26G)	45.51 (0.57G)	37.41 (1.29G)	142.53 (4.43G)
LoRA	34.99 (0.36G)	33.92 (0.80G)	25.58 (1.76G)	19.21 (6.17G)
ReLoRA	37.04 (0.36G)	29.37 (0.80G)	29.08 (1.76G)	18.33 (6.17G)
SWAN [‡] speed up vs Adam	1.52 X	1.6 X	> 2.3 X	> 2.4 X
r of low-rank methods	128	256	256	512
Training Steps	10K	20K	60K	100K

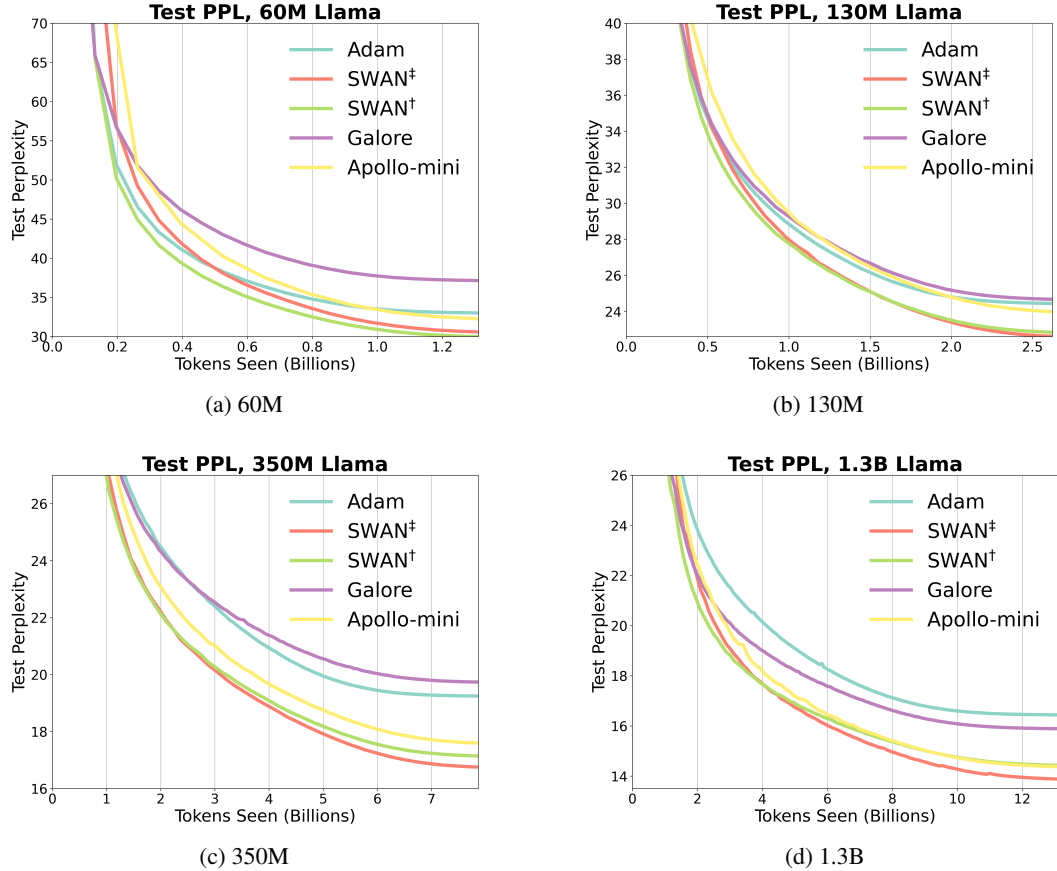


Figure 4: Comparison of convergence rate of different methods on LLM pretraining tasks. The training curves of Adam, Galore and Apollo-mini are reproduced according to the opensource code of [Zhao et al. \(2024a\)](#); [Zhu et al. \(2024\)](#). We further compare to their official results in Table 2.

Table 3: Raw and effective throughput analysis, under different model parallelization (MP) settings.

Method	Raw / eff. throughput (1B)
Adam	117872 / 117872 (tokens/s)
SWAN [†] w/o MP	58600 / 117200 (tokens/s)
SWAN [‡] w/o MP	107808 / 258739 (tokens/s)
SWAN [†] w/ MP	114160 / 228320 (tokens/s)
SWAN [‡] w/ MP	115872 / 278092 (tokens/s)

SWAN optimizer We consider three variants: **SWAN-0**, which aims to show-case the robustness and effectiveness of our method out-of-the-box, with almost no tuning. It uses naive NS-iteration for whitening, disabled learning rate warmup, and use similar learning rates optimized for Adam. **SWAN[†]**, the vanilla version of our method, in which we enabled learning rate warmup, and allowed the use of optimized learning rates that largely differ from Adam. Finally, **SWAN[‡]**, the strongest and most efficient variant that employs the proposed NSDS scheme for fast whitening (section 4.2). For all SWAN variants, we adopt a *lazy-tuning approach* (hyperparameters are set without extensive search) to reduce the possibility of unfair performance distortion. Notably, for SWAN[‡], we share the same hyperparameters across all model sizes. Full details can be found in Appendix J.

Baselines We focus on comparing SWAN with **Adam** (Kingma & Ba, 2015) and other memory-efficient optimizers. All baseline results are directly quoted from corresponding papers as they all share the same standard setup. The baselines include **Adam** (Kingma & Ba, 2015), which is a standard choice for training large models; and **Galore** Zhao et al. (2024a), a memory-efficient Adam variant with low-rank gradient projections. We also consider **Low-Rank** (Kamalakara et al., 2022), a low-rank factorization approach ($\mathbf{W} = \mathbf{BA}$); and **LoRA** Hu et al. (2021), which applies the LoRA method for pre-training as in Zhao et al. (2024a). Additionally, we include **ReLoRA** Lialin et al. (2023), a full-rank extension of LoRA with parameter merging, and **Momentum+GradWhitening**, which applies Newton-Schulz GradWhitening on top of momentum instead of GradNorm; this is equivalent to **Muon** Jordan et al. (2024) with Nesterov acceleration turned off. Finally, we compare with **Apollo-mini** and **Apollo** (Zhu et al., 2024). Full details can be found in Appendix J.

Fair Comparison We follow Kaddour et al. (2024) by allocating the same total training budget (in terms of tokens) for all methods. By the end of training, all methods decay to the same target learning rate, which is 10% of the initial learning rates. This approach eliminates an often overlooked evaluation bias, where optimizers with earlier learning rate cool-downs tend to have an unfair advantage due to artificially fast convergence caused by learning rate decay Kaddour et al. (2024).

Results As shown in Table 2 and Figure 1, all SWAN variants achieve strong performance compared with baselines, requiring the lowest memory consumptions comparable to SGD. Across all models, SWAN-0 surpasses the performance of the Adam and Momentum-GradWhitening (muon-like) baseline in terms of validation perplexities. SWAN[†] further delivers a comparable performance to Apollo series; and finally, SWAN[‡] with NSDS scheme delivers the strongest, SOTA-level performance under this setup. Notably, on the 350M and 1.3B models, all SWAN variants reaches at least $2\times$ speedup (in steps or tokens) relative to Adam (Figure 1 and Figure 5 (a)). Finally, we observe that in general, SWAN[‡] has a slower convergence speed in the early stage than SWAN[†]. However, it offers strong tail convergence in return and surpasses SWAN[†] in terms of final performance.

6.2 MEMORY EFFICIENCY AND THROUGHPUT ANALYSIS

Memory Footprint We compare SWAN, Adam, and Galore on a single A100 GPU. Unlike Zhao et al. (2024a); Zhu et al. (2024), which report layer-wise training memory usage, we measure total end-to-end memory consumption under full-model training using a batch-size of 1 for 1.3B, 7B, and 13B models. As shown in Figure 1 (c), SWAN’s memory usage is on par with SGD, providing nearly a 50% reduction in total memory. If we further incorporate the per-layer training technique of (Zhao et al., 2024a; Lv et al., 2023), SWAN further achieves $\approx 70\%$ total memory reduction on top of $\approx 100\%$ reduction on optimizer states. This underlines the benefit of the stateless design.

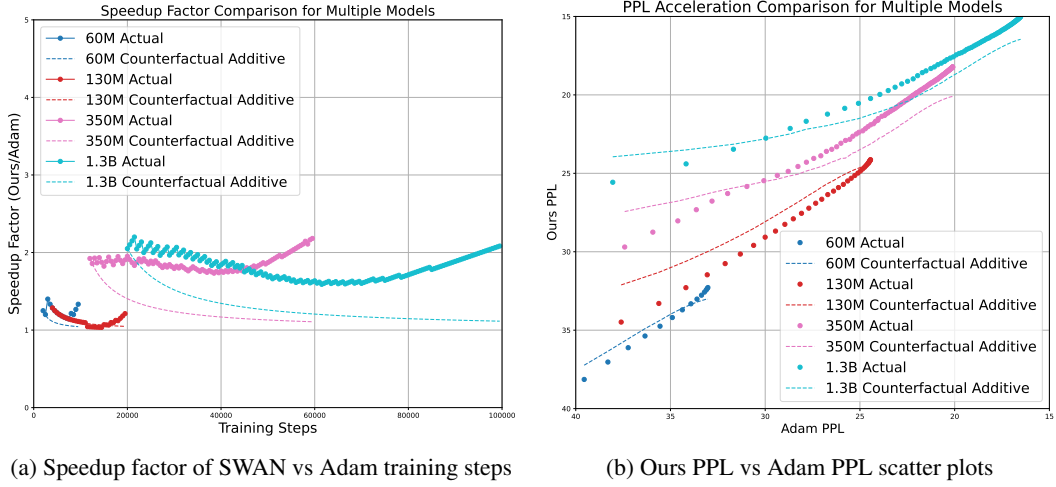


Figure 5: Comparative analysis of SWAN and Adam optimizers: speedup ratios and perplexity metrics across various model sizes. **(a)** shows how SWAN reduces the number of training steps needed to achieve the same evaluation perplexity as Adam for models ranging from 60M to 1.3B parameters. A speedup ratio greater than one indicates that SWAN reaches target PPL values faster than Adam. **(b)** presents a direct comparison of perplexity scores between SWAN and Adam. In both plots, we also provide counterfactual additive curves (dashed lines) modeling baselines corresponding to constant step advantages. Together, these plots highlight the nature of SWAN’s speedup over Adam across different model scales.

Effective Throughput We assess throughput when training a 1.3B LLaMa model on 8 A100 GPUs with a batch size of 130K. All gradient processing are done in BF16. We use two metrics: *raw throughput*: number of tokens processed per second. *Effective throughput*: raw throughput adjusted by SWAN’s token efficiency relative to Adam. These metrics evaluate the impact of different GradWhitening schemes on training speed, and also account for the fact that some optimizers make more effective use of training tokens. As shown in Table 3, SWAN[†] with naive NS GradWhitening requires model parallelization (where NS of different tensors are distributed to different GPUs) to achieve competitive throughput. Without model parallelization, the throughput of SWAN[†] is 50 % lower than Adam. With the proposed NSDS scheme, SWAN[‡] achieves a raw throughput comparable to Adam, *without* any need for model parallelization. Consequently, SWAN[‡] exhibits a $2 \times$ higher effective throughput than Adam.

6.3 IS THE SPEED-UP MULTIPLICATIVE OR ADDITIVE?

A key question regarding speedup factors is whether the improvement over Adam is *multiplicative* or *additive*. A multiplicative speedup implies that the optimizer’s relative advantage remains proportionally consistent over time, while an additive speedup suggests a less desired constant step advantage. To investigate this, we take the SWAN-0 setup as an example, and address this question using two plots, the *speed-up ratio comparison* and the *perplexity comparison* (Figure 5), across model sizes.

Speedup Ratio Definition We define the *speedup ratio* $R(P)$ for a given perplexity (PPL) threshold P as the ratio of the number of training steps Adam requires to reach a specific evaluation perplexity (PPL) to the number of required steps for SWAN:

$$R(P) = \frac{S_{\text{Adam}}(P)}{S_{\text{SWAN}}(P)}, \quad (7)$$

where $S_{\text{Adam}}(P)/S_{\text{SWAN}}(P)$ are the training steps required by Adam/SWAN to reach perplexity P .

Counterfactual Additive Curve Estimation To test whether the speedup is additive, we compute a *counterfactual additive curve* by assuming SWAN gains a fixed step advantage Δ over Adam in

early training (approximately the first 10%–20% of total steps):

$$\Delta = \frac{1}{N} \sum_{i=1}^N (S_{\text{Adam}}(P_i) - S_{\text{SWAN}}(P_i)), \quad (8)$$

where N is the number of PPL thresholds considered. We then use Δ to define the counterfactual additive speedup ratio:

$$R_{\text{additive}}(P) = \frac{S_{\text{Adam}}(P)}{S_{\text{Adam}}(P) - \Delta}, \quad (9)$$

and the counterfactual additive perplexity estimate:

$$\text{PPL}_{\text{additive}}(S) = \text{PPL}_{\text{Adam}}(S + \Delta). \quad (10)$$

This represents the expected perplexity of SWAN if it only outpaces Adam by fixed Δ steps.

Results Figure 5 compares SWAN’s actual performance against the counterfactual additive curves. If SWAN’s actual curves exceed these additive estimates, it indicates a tendency towards a multiplicative speedup, instead of the additive advantage. We summarize the observations across model sizes as: 1) for smaller models (60M and 130M), the actual speedup trajectories align closely with the additive baseline, indicating a primarily additive speedup; 2) for larger models (350M and 1.3B), the actual curves rise noticeably above the additive estimates, suggesting a multiplicative speedup. This indicates that SWAN yields increasing efficiency gains as model size grows.

6.4 ABLATION STUDIES

We take SWAN-0 from the SWAN series as an example and conduct the following ablation studies. All SWAN optimizer mentioned in this section specifically refers to the SWAN-0 setting.

Effect of GradNorm and GradWhitening on Performance We consider six ablation settings: (1) SWAN(full), (2) SWAN (GradNorm only), (3) SWAN (GradWhitening only), (4) Adam (full), (5) Adam (momentum only), and (6) Adam (second moment only). As shown in Figure 6 (a), both GradNorm and GradWhitening contribute to SWAN’s final performance. Removing either results in performance degradation. Similarly, Adam also requires all moments for optimal performance.

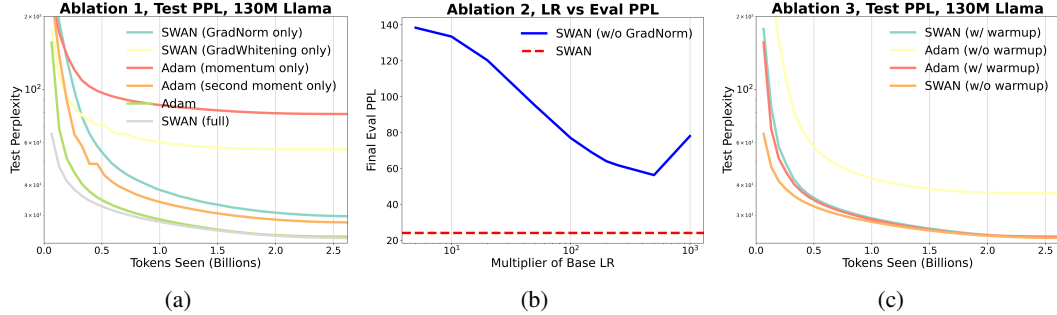


Figure 6: Ablation studies on 130M model. (a) Ablation on the contribution of each components in SWAN and Adam. (b) Ablation on removing GradNorm and compensate with larger learning rates. (c) Ablation on the effect of learning rate warm-ups.

Does SWAN Succeed Only by Increasing Effective Learning Rates though GradNorm? To answer this question, we remove the GradNorm operator from SWAN and run a learning rate sweep. Starting with the default learning rate of full SWAN, we apply multipliers from 1 to 10^3 . In Figure 6 (b), the blue line shows the final validation perplexity for SWAN without GradNorm at different learning rates multipliers, while the dashed red line represents the performance of full SWAN at the default learning rate. Although raising the learning rate can improve the performance of SWAN without GradNorm, the gap to full SWAN remains large. This indicates that GradNorm’s gradient noise stabilization is essential and cannot be replaced simply by increasing the learning rate.

How does warm-up affect the performance? Section 6.1 shows that SWAN can train with no warm-up phase even under a relatively large learning rate (0.001). Here, we compare Adam and SWAN with and without warm-ups. As seen in Figure 6 (c), SWAN without the warm-up phase gives better performance, and it still outperforms Adam under Adam’s own warm-up schedule. On the other hand, Adam’s performance decreases drastically without a proper warm-up. These findings suggest that SWAN is more robust to warm-up schedule and can train effectively with or without it.

7 OPEN QUESTIONS AND EXTENSIONS

In Section 5 we analyzed the design choice of GradNorm and GradWhitening from the perspective of LLM dynamics. However, the *compositional effect* of these two operators is still an open question, which is not discussed in this work. Specifically, in Appendix B we empirically show that when comparing with standard, standalone NS processing of the gradients, both GradWhitening and (NSDS)-GradWhitening of SWAN introduces significant changes, rotating the update by a relatively large magnitude. This indicates that additional efforts are needed to further explain the novel dynamics of SWAN, as well as the question of “what makes a good preprocessing chain for gradients”.

After the v1 version of this paper on arXiv, we posted a follow-up work of SWAN (Scetbon et al., 2025), in which we have shown that SWAN is a special case of a general framework called multi-normalized gradient descent (MNGD). MNGD aims at normalizing gradients according to multiple norms, generalizing the steepest descent viewpoint of Bernstein & Newhouse (2024b) that recasts popular first-order optimizers as normalization of gradients under a single norm.

8 CONCLUSION

We introduced SWAN, a stateless optimizer for LLM training that combines two well-known operators applied to raw gradients, GradNorm and GradWhitening, to stabilize the stochasticity of gradient distribution and neutralizing the local geometry of loss landscape, respectively. Through theoretical analysis and empirical evidence, we showed that SWAN reduces memory usage while achieving on-par or even better performance compared to Adam on LLaMA pre-training tasks. Notably, SWAN achieves $2\times$ speedups compared to Adam in terms of tokens processed when training 350M- and 1.3B-parameter models. These findings serve as a proof-of-concept that the stateless approach has the potential to serve as a practical and efficient alternative to other optimizers that require tracking internal states. Future work may explore other design choices for stateless optimizers and further expand SWAN’s applicability to other complex training regimes beyond standard LLM pre-training.

REFERENCES

- Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Learning rate grafting: Transferability of optimizer tuning.
- Rohan Anil, Vineet Gupta, Tomer Koren, Kevin Regan, and Yoram Singer. Scalable second order optimization for deep learning, 2021. URL <https://arxiv.org/abs/2002.09018>.
- Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016. URL <https://api.semanticscholar.org/CorpusID:8236317>.
- Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
- Jeremy Bernstein and Laker Newhouse. Modular duality in deep learning, 2024a. URL <https://arxiv.org/abs/2410.21265>.
- Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024b.
- Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology, 2024c. URL <https://arxiv.org/abs/2409.20325>.
- Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.
- DeepSeek-AI Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wen-Hui Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, Aixin Liu, Bo Liu (Benjamin Liu), Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Jun-Mei Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Min Tang, Bing-Li Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Yu Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yi Xiong, Hanwei Xu, Ronald X Xu, Yanhong Xu, Dejian Yang, Yu mei You, Shuiping Yu, Xin yuan Yu, Bo Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghu Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. Deepseek llm: Scaling open-source language models with longtermism. *ArXiv*, abs/2401.02954, 2024. URL <https://api.semanticscholar.org/CorpusID:266818336>.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020.
- David E Carlson, Edo Collins, Ya-Ping Hsieh, Lawrence Carin, and Volkan Cevher. Preconditioned spectral descent for deep learning. *Advances in neural information processing systems*, 28, 2015.
- Xi Chen, Kaituo Feng, Changsheng Li, Xunhao Lai, Xiangyu Yue, Ye Yuan, and Guoren Wang. Fira: Can we achieve full-rank training of llms under low-rank constraint? *arXiv preprint arXiv:2410.01623*, 2024.
- Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *NeurIPS*, 2023.
- Michael Crawshaw, Mingrui Liu, Francesco Orabona, Wei Zhang, and Zhenxun Zhuang. Robustness to unbounded smoothness of generalized signsgd. *Advances in neural information processing systems*, 35:9955–9968, 2022.

- Ashok Cutkosky and Harsh Mehta. Momentum improves normalized sgd. In *International conference on machine learning*, pp. 2260–2268. PMLR, 2020.
- André Belotto Da Silva and Maxime Gazeau. A general system of differential equations to model first-order adaptive algorithms. *Journal of Machine Learning Research*, 21(129):1–42, 2020.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, and Kevin Stone. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.
- John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011. URL <https://api.semanticscholar.org/CorpusID:538820>.
- Runa Eschenhagen, Alexander Immer, Richard Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. *Advances in Neural Information Processing Systems*, 36, 2024.
- Kaixin Gao, Xiaolei Liu, Zhenghai Huang, Min Wang, Zidong Wang, Dachuan Xu, and Fan Yu. A trace-restricted kronecker-factored approximation to natural gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7519–7527, 2021.
- Kaixin Gao, Zheng-Hai Huang, Xiaolei Liu, Min Wang, Shuangling Wang, Zidong Wang, Dachuan Xu, and Fan Yu. Eigenvalue-corrected natural gradient based on a new approximation. *Asia-Pacific Journal of Operational Research*, 40(01):2340005, 2023.
- Thomas George, César Laurent, Xavier Bouthillier, Nicolas Ballas, and Pascal Vincent. Fast approximate natural gradient descent in a kronecker factored eigenbasis. *Advances in Neural Information Processing Systems*, 31, 2018.
- Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization, 2018. URL <https://arxiv.org/abs/1802.09568>.
- Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-rank adapters are secretly gradient compressors. *ArXiv*, abs/2402.03293, 2024. URL <https://api.semanticscholar.org/CorpusID:267412117>.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 791–800, 2018.
- Lei Huang, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Iterative normalization: Beyond standardization towards efficient whitening. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4874–4883, 2019.

- Weihao Huang, Zhenyu Zhang, Yushun Zhang, Zhi-Quan Luo, Ruoyu Sun, and Zhangyang Wang. Galore-mini: Low rank gradient learning with fewer learning rates. In *NeurIPS 2024 Workshop on Fine-Tuning in Modern Machine Learning: Principles and Scalability*.
- Dongseong Hwang. Fadam: Adam is a natural gradient optimizer using diagonal empirical fisher information. *arXiv preprint arXiv:2405.12807*, 2024.
- Jacob Jackson. An isometric stochastic optimizer. *arXiv preprint arXiv:2307.12979*, 2023.
- Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.
- Kaiqi Jiang, Dhruv Malik, and Yuanzhi Li. How does adaptive optimization impact local neural network geometry? *Advances in Neural Information Processing Systems*, 36, 2024.
- Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cecista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL <https://kellerjordan.github.io/posts/muon/>.
- Jean Kaddour, Oscar Key, Piotr Nawrot, Pasquale Minervini, and Matt J Kusner. No train no gain: Revisiting efficient training algorithms for transformer-based language models. *Advances in Neural Information Processing Systems*, 36, 2024.
- Siddhartha Rao Kamalakara, Acyr Locatelli, Bharat Venkitesh, Jimmy Ba, Yarin Gal, and Aidan N Gomez. Exploring low rank training of deep neural networks. *arXiv preprint arXiv:2209.13569*, 2022.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.
- Abdoulaye Koroko, Ani Anciaux-Sedrakian, Ibtihel Ben Gharbia, Valérie Garès, Mounir Haddou, and Quang Huy Tran. Efficient approximations of the fisher matrix in neural networks using kronecker product singular value decomposition. *arXiv preprint arXiv:2201.10285*, 2022.
- Vijay Anand Korthikanti, Jared Casper, Sangkug Lym, Lawrence McAfee, Michael Andersch, Mohammad Shoeybi, and Bryan Catanzaro. Reducing activation recomputation in large transformer models. *Proceedings of Machine Learning and Systems*, 5:341–353, 2023.
- Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt. Noise is not the main factor behind the gap between sgd and adam on transformers, but sign descent might be. *arXiv preprint arXiv:2304.13960*, 2023.
- Frederik Kunstner, Robin Yadav, Alan Milligan, Mark Schmidt, and Alberto Bietti. Heavy-tailed class imbalance and why adam outperforms gradient descent on language models. *arXiv preprint arXiv:2402.19449*, 2024.
- Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 947–955, 2018.
- Xi-Lin Li. Preconditioned stochastic gradient descent. *IEEE transactions on neural networks and learning systems*, 29(5):1454–1466, 2017.
- Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. Relora: High-rank training through low-rank updates. In *International Conference on Learning Representations*, 2023. URL <https://api.semanticscholar.org/CorpusID:259836974>.
- Wu Lin, Felix Dangel, Runa Eschenhagen, Juhan Bae, Richard E Turner, and Alireza Makhzani. Can we remove the square-root in adaptive gradient methods? a second-order perspective. *arXiv preprint arXiv:2402.03496*, 2024.
- Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. *CoRR*, abs/2305.14342, 2023.

- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR (Poster)*. OpenReview.net, 2019.
- Kai Lv, Yuqing Yang, Tengxiao Liu, Qinghui Gao, Qipeng Guo, and Xipeng Qiu. Full parameter fine-tuning for large language models with limited resources. *arXiv preprint arXiv:2306.09782*, 2023.
- James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.
- James Martens, Jimmy Ba, and Matt Johnson. Kronecker-factored curvature approximations for recurrent neural networks. In *International Conference on Learning Representations*, 2018.
- Jonathan Mei, Alexander Moreno, and Luke Walters. Kradagrad: kronecker approximation-domination gradient preconditioned stochastic optimization. In *Uncertainty in Artificial Intelligence*, pp. 1412–1422. PMLR, 2023.
- Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, et al. A theory on adam instability in large-scale machine learning. *arXiv preprint arXiv:2304.09871*, 2023.
- Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.
- Matteo Pagliardini, Pierre Ablin, and David Grangier. The ademamix optimizer: Better, faster, older. *arXiv preprint arXiv:2409.03137*, 2024.
- Abel Peirson, Ehsan Amid, Yatong Chen, Vladimir Feinberg, Manfred K Warmuth, and Rohan Anil. Fishy: Layerwise fisher approximation for higher-order neural network optimization. In *Has it Trained Yet? NeurIPS 2022 Workshop*, 2022.
- Omead Pooladzandi and Xi-Lin Li. Curvature-informed sgd via general purpose lie-group preconditioners. *arXiv preprint arXiv:2402.04553*, 2024.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL <http://jmlr.org/papers/v21/20-074.html>.
- Meyer Scetbon, Chao Ma, Wenbo Gong, and Edward Meeds. Gradient multi-normalization for stateless and scalable llm training, 2025. URL <https://arxiv.org/abs/2502.06742>.
- Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4603–4611. PMLR, 2018.
- Hao-Jun Michael Shi, Tsung-Hsien Lee, Shintaro Iwasaki, Jose Gallego-Posada, Zhijing Li, Kaushik Rangadurai, Dheevatsa Mudigere, and Michael Rabbat. A distributed data-parallel pytorch implementation of the distributed shampoo optimizer for training neural networks at-scale. *arXiv preprint arXiv:2309.06497*, 2023.
- Yue Song, Nicu Sebe, and Wei Wang. Fast differentiable matrix square root and inverse square root. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):7367–7380, 2022.
- Yuandong Tian, Yiping Wang, Zhenyu Zhang, Beidi Chen, and Simon Du. Joma: Demystifying multilayer transformers via joint dynamics of mlp and attention. *arXiv preprint arXiv:2310.00535*, 2023.
- Tijmen Tieleman. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26, 2012.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023b. URL <https://arxiv.org/abs/2307.09288>.

Mark Tuddenham, Adam Prügel-Bennett, and Jonathan Hare. Orthogonalising gradients to speed up neural network optimisation. *arXiv preprint arXiv:2202.07052*, 2022.

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

Sike Wang, Pan Zhou, Jia Li, and Hua Huang. 4-bit shampoo for memory-efficient network training. *arXiv preprint arXiv:2405.18144*, 2024.

Minghao Xu, Lichuan Xiang, Xu Cai, and Hongkai Wen. No more adam: Learning rate scaling at initialization is all you need, 2024a. URL <https://arxiv.org/abs/2412.11768>.

Minghao Xu, Lichuan Xiang, Xu Cai, and Hongkai Wen. No more adam: Learning rate scaling at initialization is all you need. *arXiv preprint arXiv:2412.11768*, 2024b.

Zhirong Yang and Jorma Laaksonen. Principal whitened gradient for information geometry. *Neural Networks*, 21(2-3):232–240, 2008.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models? *Advances in Neural Information Processing Systems*, 33:15383–15393, 2020.

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

Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. Why transformers need adam: A hessian perspective. *arXiv preprint arXiv:2402.16788*, 2024a.

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

Zhenyu Zhang, Ajay Jaiswal, Lu Yin, Shiwei Liu, Jiawei Zhao, Yuandong Tian, and Zhangyang Wang. Q-galore: Quantized galore with int4 projection and layer-adaptive low-rank gradients. *arXiv preprint arXiv:2407.08296*, 2024c.

Jiawei Zhao, Zhenyu (Allen) Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *ArXiv*, abs/2403.03507, 2024a. URL <https://api.semanticscholar.org/CorpusID:268253596>.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models. *arXiv preprint arXiv:2407.07972*, 2024b.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models, 2024c. URL <https://arxiv.org/abs/2407.07972>.

Hanqing Zhu, Zhenyu Zhang, Wenyan Cong, Xi Liu, Sem Park, Vikas Chandra, Bo Long, David Z Pan, Zhangyang Wang, and Jinwon Lee. Apollo: Sgd-like memory, adamw-level performance. *arXiv preprint arXiv:2412.05270*, 2024.

Zhanxing Zhu, Jingfeng Wu, Bing Yu, Lei Wu, and Jinwen Ma. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. *arXiv preprint arXiv:1803.00195*, 2018.

A DESIRED PROPERTIES OF ADAPTIVE OPTIMIZERS

There is a rich literature on understanding adaptive methods’ inner workings and unreasonable effectiveness. Using Adam as an example, we first summarize from the literature below the key desired properties of stateful adaptive optimizers that contribute to their empirical success: *gradient smoothing*, *gradient invariance*, and *gradient whitening*. Then we discuss how these understandings will lead to the design of *stateless* adaptive optimizers.

Gradient Smoothing. Under the stochastic optimization setting, mini-batch sampling introduces heterogeneous distribution shift on the gradient distribution: $\mathbf{G}^{(t)} = \mathbb{E}[\mathbf{G}^{(t)}] + \epsilon^{(t)}$, where $\epsilon^{(t)}$ is time-heterogeneous noise induced by mini-batch sampling. While $\epsilon^{(t)}$ helps SGD escapes local optima (Jastrzēbski et al., 2017; Zhu et al., 2018), the *covariate shift* of $\epsilon^{(t)}$ over time also present challenges to learning as the model needs to adjust and compensate for this shift, especially under the emergence of heavy tailed gradient distributions (Zhang et al., 2020)³. Following this viewpoint, it has been proven that momentum reduces the influence of noises for SGD (Cutkosky & Mehta, 2020; Crawshaw et al., 2022). Therefore we hypothesize that the first moment estimate $\mathbf{m}^{(t)}$ of Adam also effectively stabilizes gradient distribution and reduces effect of $\epsilon^{(t)}$. This smoothing stabilizes the variance caused by noisy stochastic gradients across time.

Gradient Invariance. More recently it has also been identified (Kunstner et al., 2023; 2024) that the major factor contributing to the performance gap between SGD and Adam might lie in Adam’s *Sign-descent*-like nature (Bernstein et al., 2018; Crawshaw et al., 2022; Chen et al., 2023). Intuitively, Adam without bias correction under $\beta_1 = 0$ and $\beta_2 = 0$ is equivalent to signed gradient descent ($\Delta \mathbf{W} = \text{sign}(\mathbf{G})$). Indeed, the performance of Adam can be closely reproduced (Kunstner et al., 2023; Crawshaw et al., 2022) or even surpassed (Chen et al., 2023) by variants of signed descent with momentum. Apart from sign-based methods, evidence on performance boost using gradient clipping/normalization was also discussed in the context of understanding Adam (Zhang et al., 2020). Therefore, we hypothesize that one of the key properties of Adam is that it offers *invariance over certain transformations* on gradients. Particularly, the original Adam is invariant to diagonal rescaling of the gradients (Kingma & Ba, 2015); the signed gradient method is invariant to *any* scaling that preserves the sign of gradients; and the clipped SGD variant is invariant to extreme gradient magnitude spikes.

Gradient Whitening. Finally, we argue that the empirical success of adaptive methods also lies in that they model the curvature by first-order information. This is realized by the second moment estimate $\nu^{(t)}$, which approximates the diagonal of the Fisher information matrix (Kingma & Ba, 2015; Hwang, 2024); helping to counteract local curvatures of the problem. Specifically, Adam computes a trailing estimation of the diagonal coefficients of the Fisher matrix $\mathbf{F} = \mathbb{E}[\mathbf{g}\mathbf{g}^\top]$ by tracking $\hat{\mathbf{F}} = \text{diag}(\mathbf{F}) = \text{diag}[\mathbb{E}[\mathbf{g}^2]]$, where $\mathbf{g} = \text{vec}(\mathbf{G})$ is the vectorized gradient. Interestingly, instead of preconditioning the first moment as $\hat{\mathbf{F}}^{-1}\text{vec}(\mathbf{m})$, Adam uses a whitening-like preconditioned update $\hat{\mathbf{F}}^{-\frac{1}{2}}\text{vec}(\mathbf{m})$, suggesting an *element-wise* approximate whitening of the gradient. It has been shown that such element-wise whitening leads to diagonal approximation to inverse Hessian $\hat{\mathbf{F}}^{-\frac{1}{2}} \approx \text{diag}(\mathbf{H}^{-1})$ (Molybog et al., 2023). Recent empirical studies show that Adam biases optimization trajectories towards regions where the condition number of Hessian is low (Jiang et al., 2024). Therefore, we hypothesize that Adam approximately whitens the gradients element-wise, leading to well-conditioned regions.

³Such shift cannot be removed by forward covariate-shift reduction architectures such Layer Norm, as it is only invariant to global scaling and re-centering, such as $\mathbf{W}^{(t)} = \delta \mathbf{W}^{(t)} + \gamma \mathbf{1}^\top$ for some scalar δ and incoming vector shift γ (Ba et al., 2016).

B ACCELERATION OF NEWTON-SCHULZ ITERATION VIA DIAGONAL SUBSTITUTION

B.1 ALGORITHMS

Computing `GradWhitening` exactly can be expensive, as it involves solving the matrix square-root inverse. One option is to directly apply the Newton-Schulz variant of decorrelated batch normalization (Song et al., 2022; Li et al., 2018; Huang et al., 2019), which allows a more GPU-friendly estimation. This is given by (Song et al., 2022; Li et al., 2018):

$$\begin{cases} \mathbf{Y}_{k+1} = \frac{1}{2}\mathbf{Y}_k(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k) \\ \mathbf{Z}_{k+1} = \frac{1}{2}(3\mathbf{I} - \mathbf{Z}_k\mathbf{Y}_k)\mathbf{Z}_k \end{cases}$$

where $\mathbf{Y}_0 = \mathbf{G}\mathbf{G}^\top$, $\mathbf{Z}_0 = \mathbf{I}$. At convergence, $\text{GradWhitening}(\mathbf{G}) = \mathbf{Z}\mathbf{G}$ (Algorithm 2). However, estimating $(\mathbf{G}\mathbf{G}^\top)^{-1/2}$ with NS requires $\mathcal{O}(m^3)$ (assuming $m < n$) complexity. Here, we propose a heuristic scheme that has $\mathcal{O}(m^2)$ complexity to estimate square-root inverse:

$$\begin{cases} \mathbf{Y}_{k+1} = \frac{1}{2}\mathbf{Y}_k\text{Diag}(3\mathbf{I} - \mathbf{Z}_k\text{Diag}(\mathbf{Y}_k)) \\ \mathbf{Z}_{k+1} = \frac{1}{2}(3\mathbf{I} - \text{Diag}(\mathbf{Z}_k)\mathbf{Y}_k)\text{Diag}(\mathbf{Z}_k) \end{cases}$$

where $\text{Diag}(\cdot)$ returns a diagonal matrix that has the same diagonal elements as the input matrix. Basically, whenever we encounter matrix multiplication in NS iterations, we replace one of them by its diagonal approximation. We refer to this as the *NS with diagonal substitution* (NSDS) scheme.

Note that in the above standard presentation we have fixed that both NS and NSDS uses coefficients = 0.5 for \mathbf{Y} updates and \mathbf{Z} updates, respectively. In practice we may further tune these coefficients to compensate for short number of iterations (usually under 10).

B.2 EXPERIMENT: LLM GRADIENT CONDITION NUMBER REDUCTION

Setup In this synthetic experiment, we assessed the effectiveness of two whitening methods, Newton-Schulz and the proposed Newton-Schulz with diagonal substitution (NSDS) scheme, on gradient matrices obtained from LLM training. As discussed in Section 4, the exact `GradWhitening` operator results in matrices with optimal condition number ($= 1$); therefore, we hereby investigate the matrix condition numbers of the processed gradients obtained from different methods. Specifically, both methods use 5 NS iterations with NS step size optimized. We train a 130M Llama model following the architecture setting of Section 6.1 on randomly generated sequences for 1000 steps, and take the MLP weights of a middle layer (we take the fifth layer without loss of generality) and use different methods to whiten the gradient matrices. We consider three methods: standard NS; NSDS; and SWAN with NSDS (that is, composing `GradNorm` with NSDS-`GradWhitening`). At each training step, the condition number reduction ratio of different method was calculated for both whitening methods (the higher the better). Note that for all methods, the gradients have been pre-normalized by its norm.

Results Results are shown in Figure 7 (a). We notice that NSDS alone (orange curve) is not sufficient to reach a good condition number reduction ratio. However, when combined with `GradNorm` (i.e., SWAN with NSDS, the green curve), its performance started to catch up and even outperform the standard NS method after 500 training steps. This show the effectiveness of the proposed scheme. One potential caveat that we spot is that the condition number produced by SWAN with NSDS is more noisy than the standard NS iteration; which might lead to improvements that will be addressed in future work.

The significance of `GradNorm` and NSDS The results above highlight the importance of `GradNorm`. A potential question is whether `GradNorm`, when followed by `GradWhitening`, is merely a no-op that rescales the initial location of the NS iteration for better convergence. In fact, for all methods considered in Figure 7, the gradients have been pre-normalized by their (global) norm before being fed into each method. This re-scaling applied to all methods provides a negative answer to the question. To clarify further, we estimate the cosine similarities between the processed gradients produced by different pairs of methods. The results, shown in Figure 7 (b), reveal the following: As

the training iterations increase, the cosine similarity score between SWAN-NSDS and NS (denoted as $\cos(\text{SWAN-NSDS}, \text{NS})$) monotonically decreases to small values, indicating a near-orthogonal relationship. Both $\cos(\text{SWAN}, \text{NS})$ and $\cos(\text{SWAN}, \text{SWAN-NSDS})$ decrease over time, suggesting that both GradNorm and NSDS contribute to the orthogonality of $\cos(\text{SWAN-NSDS}, \text{NS})$. This demonstrates that the changes introduced by both GradWhitening and NSDS-GradWhitening are significant, rotating the update by a relatively large magnitude. This might also explain the observation in Section 6.1 that SWAN with NSDS behaves differently from other variants, showing slower early convergence but stronger long-term convergence.

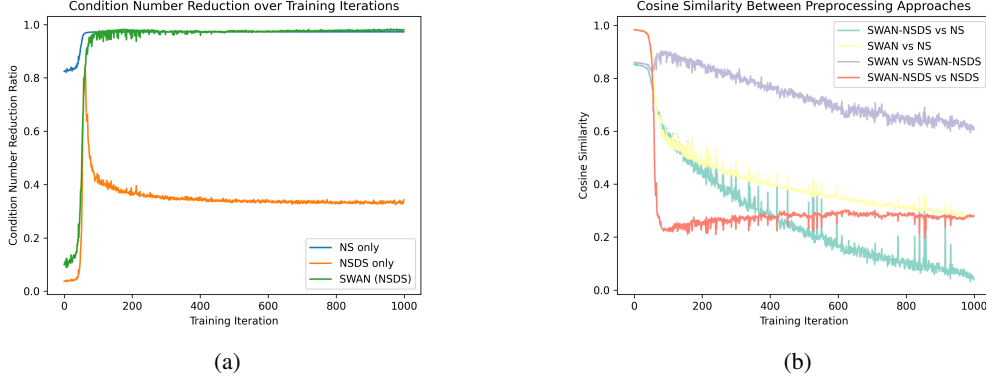


Figure 7: Comparison between different whitening schemes. (a) Performance comparison by condition number reduction ratio (higher the better). (b) Cosine similarities between the whitened matrix produced by different preprocessing schemes, respectively.

B.3 ABLATION: EFFECT OF NSDS ITERATIONS

We examine the impact of the number of iterations of our Newton Schulz with Diagonal Substitution (NSDS) scheme. With SWAN[‡], we compare the test PPL performance on 130M model. Results are shown in Table 4. As we can see, the improvement brought by additional NSDS iteration is marginal; hence in this paper, we only apply 2 iterations of NSDS.

Table 4: Effect of NSDS iterations on test PPL. Results are obtain using FP32 precision.

# NSDS iterations	Test PPL
1	- (LLM loss diverge)
2	22.63
5	22.62
10	22.61

B.4 ABLATION: PRECISION

We compare the performance of different precisions of GradWhitening with NSDS scheme. As shown in Table 5, on 130M model (2 step NSDS iterations) ablation we show that BF16 can be used without major performance degrade compared to FP32.

Table 5: BF16 vs FP 32 NSDS

NSDS Precision	Test PPL on 130M
BF16	22.61
FP32	22.63

C ADDITIONAL ANALYSIS

C.1 ANALYZING THE GRADWHITENING PT. II: ROBUSTNESS AGAINST LOCAL CURVATURE

In this section, we present main results regarding the convergence rate of the GradWhitening method, understand its implications, and compare it with the lower bounds of GD and Adam.

First, for simplicity, we focus on the following quadratic problem:

$$\mathcal{L}(\mathbf{W}) = \frac{1}{2}\text{Tr}(\mathbf{W}^\top \mathbf{H} \mathbf{W}) - \text{Tr}(\mathbf{C}^\top \mathbf{W}), \quad (11)$$

where $\mathbf{W} \in \mathbb{R}^{m \times n}$ is the parameter matrix, $\mathbf{H} \in \mathbb{R}^{m \times m}$ is a positive definite matrix, and $\mathbf{C} \in \mathbb{R}^{m \times n}$ is a constant matrix.

For simplicity and without loss of generality, we assume $\mathbf{C} = 0$. This is because minimizing $\mathcal{L}(\mathbf{W}) = \frac{1}{2}\text{Tr}(\mathbf{W}^\top \mathbf{H} \mathbf{W}) - \text{Tr}(\mathbf{C}^\top \mathbf{W})$ is equivalent to minimizing $\mathcal{L}(\mathbf{W}) = \frac{1}{2}\text{Tr}[(\mathbf{W} - \mathbf{W}^*)^\top \mathbf{H}(\mathbf{W} - \mathbf{W}^*)]$, where $\mathbf{W}^* = \mathbf{H}^{-1}\mathbf{C}$. By defining $\mathbf{Z} = \mathbf{W} - \mathbf{W}^*$, the problem reduces to minimizing $\mathcal{L}(\mathbf{Z}) = \frac{1}{2}\text{Tr}(\mathbf{Z}^\top \mathbf{H} \mathbf{Z})$.

Remark Most results in this note can be easily extended to any loss function that are either i) strongly convex; or ii) has twice differentiable functions and Lipschitz continuous Hessian, by considering their the second order approximation around \mathbf{W}^* .

Next, to understand the effect of GradWhitening, we will examine the gradient flow dynamics induced by GradWhitening. Consider the GradWhitening-modified gradient descent:

$$\Delta \mathbf{W}^{(t)} = -\eta \text{GradWhitening}(\mathbf{G}^{(t)}) \quad (12)$$

its exact convergence rate is given by the result as below:

Theorem 2 (Contraction factor of GradWhitening). *Consider the quadratic loss function Equation (11). Assume the initialization distribution of \mathbf{W}^0 assigns zero probability to any set of zero Lebesgue measure in $\mathbb{R}^{m \times n}$. Let our update rule be:*

$$\mathbf{W}_{\text{whitened}}^{(t+1)} = \mathbf{W}_{\text{whitened}}^{(t)} - \eta \text{GradWhitening}(\mathbf{G}^{(t)})$$

where the learning rate is η . Then, with probability 1, we have:

- The optimal dynamic learning rate to achieve the fastest convergence is given by

$$\eta^{(t)*} = \frac{\|\mathbf{H} \mathbf{W}_{\text{whitened}}^{(t)}\|_1}{\text{Tr}[\mathbf{H}]}. \quad (13)$$

where $\|\mathbf{H} \mathbf{W}_{\text{whitened}}^{(t)}\|_1$ denotes the Schatten p -norm with $p = 1$ (i.e., sum of singular values).

- Under $\eta^{(t)*}$, the contraction factor of loss function at t is given by:

$$\frac{\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t+1)}) - \mathcal{L}^*}{\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t)}) - \mathcal{L}^*} = 1 - \frac{\|\mathbf{H} \mathbf{W}_{\text{whitened}}^{(t)}\|_1^2}{\text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H} \mathbf{W}_{\text{whitened}}^{(t)}] \text{Tr}[\mathbf{H}]} \quad (14)$$

- Furthermore, if we additionally enforce $\mathbf{W}^0 \sim V^{m \times n}(\mathbb{R})$, i.e., initialized as an element in Stiefel manifold. Then we have

$$\frac{\mathcal{L}(\mathbf{W}_{\text{whitened}}^{t=1}) - \mathcal{L}^*}{\mathcal{L}(\mathbf{W}^0) - \mathcal{L}^*} = 0 \quad (15)$$

That is, GradWhitening solves the optimization problem (Equation (11)) with 1 step iteration.

Theorem 2 has the following key implications.

Convergence rate is condition number agnostic Unlike the convergence rates of GD and Adam presented in Zhang et al. (2024a), as well as Theorem 3 and Corollary 1 in Appendix, the optimal convergence rate (14) of GradWhitening no longer explicitly depends on the condition number κ of H . In fact, consider a lower bound $\frac{\|\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}\|_1^2}{\text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]\text{Tr}[\mathbf{H}]} \geq \frac{\text{Tr}[\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]^2}{\text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]\text{Tr}[\mathbf{H}]}$, since trace of H appear both in the nominator and denominator, we expect that to be more robust to ill-conditioned problems. For example, consider the specific initialization $\mathbf{W}_{\text{whitened}}^{(t)} = c\mathbf{I}$, it is straightforward to show that $\frac{\|\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}\|_1^2}{\text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]\text{Tr}[\mathbf{H}]} \geq \frac{\text{Tr}[\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]^2}{\text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]\text{Tr}[\mathbf{H}]} \perp \kappa$, which is completely disentangled from the condition number. Hence $\frac{\|\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}\|_1^2}{\text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]\text{Tr}[\mathbf{H}]}$ would not shrink as $\kappa \rightarrow \infty$. See Proposition 2 for less extreme situations.

Superlinear convergence with Stiefel manifold initialization Theorem 2 suggests that if $\mathbf{W}_{\text{whitened}}^{(t)}$ is initialized in the Stiefel manifold, then GradWhitening reaches superlinear convergence rate (= Newton’s method), while being cheaper. In fact, it is straightforward to verify that GradWhitening reaches optimal solution with 1 step update. This implies GradWhitening is theoretically the optimal optimization algorithm if \mathbf{W} is initialized in the Stiefel manifold.

Estimation and interpretation of optimal learning rate Compared to the optimal dynamic learning rate of gradient descent $G = \frac{G^\top G}{G^\top H G}$, the optimal learning rate $\eta^{(t)*}$ of GradWhitening is much easier to compute. $\frac{\text{Tr}[\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]}{\text{Tr}[\mathbf{H}]}$ can be seen as balancing the average gradient magnitude against the average curvature. A higher trace of gradient ($\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}$) (strong gradients) relative to \mathbf{H} (steep curvature) suggests a larger learning rate, promoting faster updates. Conversely, a higher trace of \mathbf{H} would imply a smaller learning rate to ensure stable convergence in regions with high curvature.

Next, we show that the convergence speed of GradWhitening update is indeed robust to the condition number of local curvature.

Proposition 2 (Robustness of GradWhitening update convergence rate against the condition number of local Hessian). *Consider the quantity:*

$$Q := \frac{\text{Tr}[\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]^2}{\text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]\text{Tr}[\mathbf{H}]}$$

Assume: i), $\mathbf{W}_{\text{whitened}}^{(t)} \neq \mathbf{W}^*$; and ii) the norm of \mathbf{H} is bounded. Then, there exist some finite positive constant c , such that

$$Q > c$$

This holds even if $\kappa \rightarrow +\infty$, where κ is the condition number of \mathbf{H} .

Below, we provide comparison between GradWhitening modified gradient descent and Adam. We only consider non-Stiefel initialization for GradWhitening, since with non-Stiefel initialization GradWhitening is optimal according to Theorem 2. Our results below shows that, for poor conditioned problems GradWhitening with a properly chosen single global learning rate always outperforms Adam even with *optimally tuned sub-group learning rates*, in terms of convergence speed.

Proposition 3 (GradWhitening with single lr vs Adam with tuned group lr). *Consider the optimization problem Equation (11). Assume \mathbf{H} is block-diagonal, i.e., $\mathbf{H} = \text{diag}(\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_L)$, where each $\mathbf{H}_l \in \mathbb{R}^{m_l \times m_l}$ is a positive definite matrix for $l = 1, 2, \dots, L$, and $\sum_{l=1}^L m_l = m$. Assuming for GradWhitening we use one global learning rate for all parameters; and for Adam, we use the optimally chosen group learning rate η_l and initial condition w_0 for each block \mathbf{H}_l .*

Assume either if i) certain regularity conditions are met (see proof in Appendix), or ii), if \mathbf{H} is poorly-conditioned (its condition number is large enough). Then: regardless of its initialization, GradWhitening with a properly chosen learning rate will still have a strictly better convergence speed (i.e., smaller contraction factor) across all blocks $l \in [L]$ than Adam ($\beta_1 = 0, \beta_2 = 1$) under optimal group-wise learning rates and initial condition.

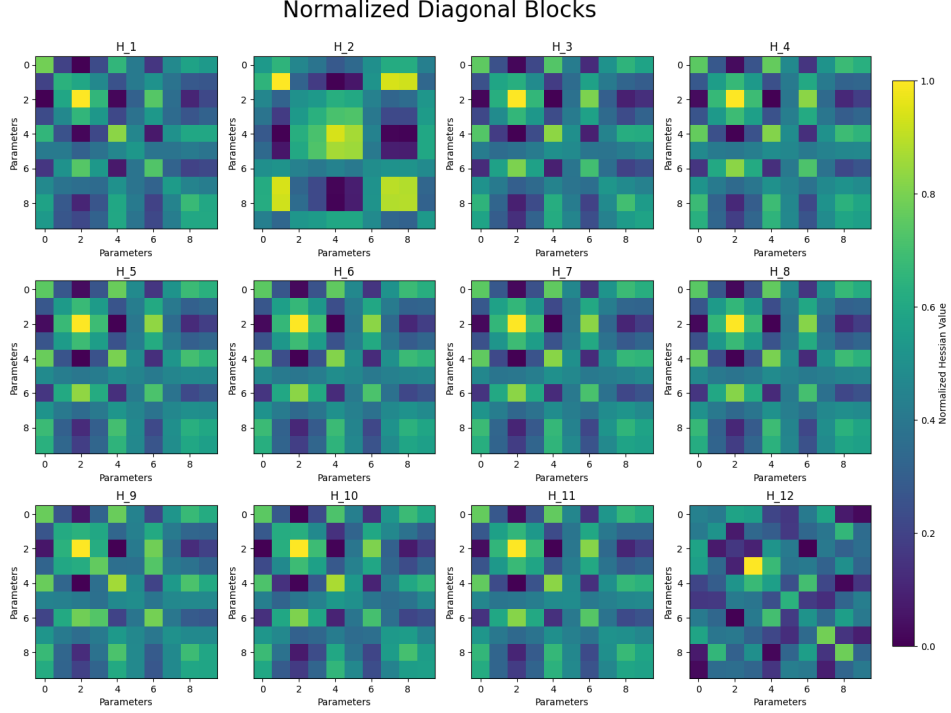


Figure 8: Normalized Hessian Blocks of size $M_C \times M_C$ along the diagonal direction of the Hessian, obtained from numerically solving the STB ODE (with $n = 12$, $M_C = 10$) (1) given by the full-batch dynamics (i.e., removing noise in Equation (16)). During all training steps, we analytically track the evolution of Hessian. As predicted by Proposition 1, we see very similar structures across the diagonal blocks of the Hessian.

Remark As pointed out by Zhang et al. (2024a) and Da Silva & Gazeau (2020), Adam with $\beta_2 < 1$ will have issues with convergence, which will not be completely removed even with lr decay. Therefore, we will not discuss the case of $\beta_2 < 1$ to avoid the complication.

C.2 NUMERICAL VERIFICATION OF PROPOSITION 1

Given a STB, we consider the following standard full-batch learning dynamics (Tian et al., 2023). Define the conditional expectation $\mathbb{E}_{q=m}[\cdot] := \mathbb{E}[\cdot | q = m]$. Consider the dynamics of the weight matrix W and the attention logits z_q , if we train the model with a batch of inputs that always end up with query $q[i] = m$. The weight update for W and z_q are given by the following noisy updates:

$$\dot{\mathbf{W}}^{(t)} = \mathbb{E}_{q=m} \left[\mathbf{f}^{(t)} (\mathbf{G}_h \odot \mathbf{h}'^{(t)})^\top \right], \quad \dot{\mathbf{z}}_m^{(t)} = \mathbb{E}_{q=m} \left[\left(\frac{\partial \mathbf{b}}{\partial \mathbf{z}_m^{(t)}} \right)^\top \mathbf{U}_C^\top \mathbf{g}_f^{(t)} \right], \quad (16)$$

Where $\mathbf{f}^{(t)} = \left(\mathbf{U}_C \left(\exp(\mathbf{z}_q^{(t)}) \odot \mathbf{x} \right) + \mathbf{u}_q \right)$, $(\mathbf{h}^{(t)})' = \phi'((\mathbf{W}^{(t)})^\top \mathbf{f}^{(t)})$ is the derivative of the current activation, $\mathbf{G}_h^{(t)} = \nabla_{\mathbf{h}^{(t)}} \mathcal{L}$ is the gradient of the loss function \mathcal{L} with respect to the hidden activation $\mathbf{h}^{(t)}$, and $\mathbf{g}_{\mathbf{f}^{(t)}}^{(t)} = \sum_k \mathbf{g}_{\mathbf{h}_k^{(t)}}^{(t)} (\mathbf{h}_k^{(t)})' \mathbf{w}_k^{(t)}$ is the sum of the gradients with respect to the attention logits. Here, $\mathbf{w}_k^{(t)}$ is the k -th column of $\mathbf{W}^{(t)}$, $\mathbf{g}_{\mathbf{h}_k^{(t)}}^{(t)}[i]$ be the backpropagated gradient sent to node k at sample i .

Then, we numerically solving the STB ODE with $n = 12$, $M_C = 10$ in Equation (16). During all training steps, we analytically track the evolution of Hessian of rmW . Results are shown in Figure 8. As predicted by Proposition 1, we see very similar structures across the diagonal blocks of the Hessian.

D PROOF OF THEOREM 1

Proof. We first consider the noiseless, full batch dynamics. Define $\mathbf{V} \in \mathbb{R}^{M_C \times n}$ as $\mathbf{V} := \mathbf{U}_C^\top \mathbf{W}$. Then following Theorem 2 in [Tian et al. \(2023\)](#), each column of \mathbf{V} satisfies the following differential equation:

$$\dot{\mathbf{V}}_{[:,j]} = \exp(\mathbf{V}_{[:,j]}^2/2 + C) \odot \mathbb{E}_q[g_{h_j} \mathbf{x}] \quad (17)$$

The corresponding dynamics of attention score is given by:

$$\mathbf{z}_q = \frac{1}{2} \sum_j \mathbf{V}_{[:,j]}^2. \quad (18)$$

Without loss of generality, in this proof we only consider $C = 0$.

Now, following the argument of Lemma B.6 of [Zhao et al. \(2024a\)](#), we reparameterize the dynamics *row-wise*. For this, consider instead

$$\mathbf{V} = \begin{bmatrix} \mathbf{u}_1^\top \\ \mathbf{u}_2^\top \\ \vdots \\ \mathbf{u}_{M_C}^\top \end{bmatrix}$$

Then, equation 17 becomes:

$$\dot{\mathbf{u}}_i = [\exp(\mathbf{u}_i^2) \cdot \mathbf{1}] \boldsymbol{\mu}_i \quad (19)$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^{n \times 1}$ is given by $[\boldsymbol{\mu}_i]_j := \mathbb{E}_q[g_{h_j} x_i]$. Therefore, it is clear that \mathbf{u}_i always move along the direction of $\boldsymbol{\mu}_i$ due to the stationary back-propagated gradient assumption. Hence, $\dot{\mathbf{u}}_i = \alpha_i(t) \boldsymbol{\mu}_i$ for some scalar dynamics $\alpha_i(t)$.

Next, consider the mini-batch version of the dynamics. In this case, the packpropagated gradient term $[\boldsymbol{\mu}_i]_j := \mathbb{E}_q[g_{h_j} x_i]$ is corrupted by some i.i.d. mini-batch noise $\boldsymbol{\xi}$. The noisy row-wise dynamics now becomes:

$$\dot{\mathbf{u}}_i = \alpha_i(t) (\boldsymbol{\mu}_i + \boldsymbol{\xi}_i) \quad (20)$$

Therefore, after row-wise standardization, the new dynamics becomes

$$\begin{aligned} \dot{\hat{\mathbf{u}}}_i &= \frac{\alpha_i(t) (\boldsymbol{\mu}_i + \boldsymbol{\xi}_i)}{\alpha_i(t) (\frac{1}{n} \sum_j (\mu_{ij} + \xi_{ij})^2)} \\ &= \frac{(\boldsymbol{\mu}_i + \boldsymbol{\xi}_i)}{(\frac{1}{n} \sum_j (\mu_{ij} + \xi_{ij})^2)} \end{aligned}$$

Therefore, the normalized noisy gradient $\dot{\hat{\mathbf{u}}}_i$ no longer depend on the time-variant component $\alpha(t)$. Hence, we have proved:

$$\text{Cov}[\tilde{\mathbf{G}}_{\mathbf{U}_C^\top \mathbf{W}}[i, :](^{t_1})] = \text{Cov}[\tilde{\mathbf{G}}_{\mathbf{U}_C^\top \mathbf{W}}[i, :](^{t_2})] \quad \text{for all } t_1, t_2, \text{ and } i.$$

The corresponding result for $\tilde{\mathbf{G}}_{\mathbf{z}_q}^{(t)} := \text{GradNorm}(\frac{\partial \mathcal{L}_{\mathbf{W}, \mathbf{z}_q}(\top \mathbf{x}^{(t)})}{\mathbf{z}_q})$ can be trivially derived due to Equation (18).

□

E PROOF OF THEOREM 2

Proof. We first show that $\nabla \mathcal{L}(\mathbf{W}^{(0)}) = \mathbf{H} \mathbf{W}^{(0)}$ (and hence $\nabla \mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t)})$ with $t \neq \infty$) are non-zero with probability 1 under Assumption of the theorem. Given $\nabla \mathcal{L}(\mathbf{W}^{(0)}) = \mathbf{H} \mathbf{W}^{(0)}$, the set of matrices $\mathbf{W}^{(0)}$ such that $\text{Tr}(\mathbf{H} \mathbf{W}^{(0)}) = 0$ forms a hyperplane in the space of $d \times d$ matrices. Specifically, it is defined by the linear equation: $\text{Tr}(\mathbf{H} \mathbf{W}^{(0)}) = 0$. Since \mathbf{H} is positive definite, at least one entry of \mathbf{H} is non-zero. Thus, the hyperplane $\text{Tr}(\mathbf{H} \mathbf{W}^{(0)}) = 0$ has zero Lebesgue measure

in the space of $d \times d$ matrices. Given that $\mathbf{W}^{(0)}$ is sampled from a continuous distribution, the probability that $\text{Tr}(\mathbf{H}\mathbf{W}^{(0)}) = 0$ is zero. Therefore, $\nabla \mathcal{L}(\mathbf{W}^{(0)}) \neq 0$ (and hence $\nabla \mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t)})$ with $t \neq \infty$) with probability 1.

Next, we define the cost-to-go as:

$$\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}^* = \frac{1}{2} \text{Tr} \left[(\mathbf{W}^{(t)})^\top \mathbf{H} \mathbf{W}^{(t)} \right],$$

and the per-step improvement is (since $\mathcal{L}^* = 0$ under $\mathbf{W} = 0$,):

$$\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}(\mathbf{W}^{(t+1)}) = \frac{1}{2} \text{Tr} \left[(\mathbf{W}^{(t)})^\top \mathbf{H} \mathbf{W}^{(t)} \right] - \frac{1}{2} \text{Tr} \left[(\mathbf{W}^{(t+1)})^\top \mathbf{H} \mathbf{W}^{(t+1)} \right].$$

Substituting the update rule $\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \text{GradWhitening}(\mathbf{G}_{\text{whitened},l}) = \mathbf{W}^{(t)} - \eta \mathbf{U} \mathbf{V}^\top$, we get:

$$\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}(\mathbf{W}^{(t+1)}) = \frac{1}{2} \text{Tr} \left[(\mathbf{W}^{(t)})^\top \mathbf{H} \mathbf{W}^{(t)} \right] - \frac{1}{2} \text{Tr} \left[(\mathbf{W}^{(t)} - \eta \mathbf{U} \mathbf{V}^\top)^\top \mathbf{H} (\mathbf{W}^{(t)} - \eta \mathbf{U} \mathbf{V}^\top) \right].$$

Expanding the right-hand side, we have

$$\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}(\mathbf{W}^{(t+1)}) = \eta \text{Tr} \left[(\mathbf{W}^{(t)})^\top \mathbf{H} \mathbf{U} \mathbf{V}^\top \right] - \frac{\eta^2}{2} \text{Tr} \left[(\mathbf{U} \mathbf{V}^\top)^\top \mathbf{H} (\mathbf{U} \mathbf{V}^\top) \right].$$

Now, noticing that $\mathbf{G} = \mathbf{H} \mathbf{W}^{(t)} = \mathbf{U} \Sigma \mathbf{V}^\top$, we have:

$$\text{Tr} \left[(\mathbf{W}^{(t)})^\top \mathbf{H} \mathbf{U} \mathbf{V}^\top \right] = \text{Tr} \left[(\mathbf{H} \mathbf{W}^{(t)})^\top \mathbf{U} \mathbf{V}^\top \right] = \text{Tr} \left[(\mathbf{U} \Sigma \mathbf{V}^\top)^\top \mathbf{U} \mathbf{V}^\top \right] = \text{Tr} \left[\mathbf{V} \Sigma \mathbf{U}^\top \mathbf{U} \mathbf{V}^\top \right] = \text{Tr} \left[\mathbf{V} \Sigma \mathbf{V}^\top \right].$$

Since \mathbf{V} is orthogonal, $\mathbf{V}^\top \mathbf{V} = \mathbf{I}$, and Σ is diagonal, we obtain:

$$\text{Tr} \left[(\mathbf{W}^{(t)})^\top \mathbf{H} \mathbf{U} \mathbf{V}^\top \right] = \text{Tr}(\Sigma) = \|\mathbf{H} \mathbf{W}_{\text{whitened}}^{(t)}\|_1.$$

Similarly:

$$\text{Tr} \left[(\mathbf{U} \mathbf{V}^\top)^\top \mathbf{H} (\mathbf{U} \mathbf{V}^\top) \right] = \text{Tr} \left[\mathbf{V} \mathbf{U}^\top \mathbf{H} \mathbf{U} \mathbf{V}^\top \right] = \text{Tr} \left[\mathbf{V} \Lambda \mathbf{V}^\top \right] = \text{Tr}(\mathbf{H}),$$

where Λ is the eigenvalue matrix of \mathbf{H} . Given those intermediate results, we have:

$$\begin{aligned} \frac{\mathcal{L}(\mathbf{W}^{(t+1)}) - \mathcal{L}^*}{\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}^*} &= 1 - \frac{\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}(\mathbf{W}^{(t+1)})}{\mathcal{L}(\mathbf{W}^{(t)}) - \mathcal{L}^*} \\ &= 1 - \frac{\eta \|\mathbf{H} \mathbf{W}_{\text{whitened}}^{(t)}\|_1 - \frac{\eta^2}{2} \text{Tr}(\mathbf{H})}{\frac{1}{2} \text{Tr} \left[(\mathbf{W}^{(t)})^\top \mathbf{H} \mathbf{W}^{(t)} \right]}. \end{aligned}$$

Noticing that this is a quadratic function of η and the second order coefficient is positive, it is straightforward to verify via the quadratic formula that the optimal learning rate is given by

$$\eta_t^* = \frac{\|\mathbf{H} \mathbf{W}_{\text{whitened}}^{(t)}\|_1}{\text{Tr}(\mathbf{H})}.$$

Under which the optimal contraction factor is given by

$$\frac{\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t+1)}) - \mathcal{L}^*}{\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t)}) - \mathcal{L}^*} = 1 - \frac{\|\mathbf{H} \mathbf{W}_{\text{whitened}}^{(t)}\|_1^2}{\text{Tr} \left[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H} \mathbf{W}_{\text{whitened}}^{(t)} \right] \text{Tr}(\mathbf{H})}.$$

Finally, if we additionally enforce $\mathbf{W}^{(0)} \sim V^{m \times n}(\mathbb{R})$, i.e., we can parameterize $\mathbf{W}^{(0)} = \mathbf{O}$ where \mathbf{O} is orthogonal, then it is trivial to verify that GradWhitening reaches the optimal solution with a 1-step update. To see this, consider the GradWhitening update:

$$\mathbf{W}_{\text{whitened}}^{(1)} = \mathbf{O} - \eta^* \text{GradWhitening}(\mathbf{H} \mathbf{O}) = \mathbf{O} - \frac{\|\mathbf{H} \mathbf{O}\|_1}{\text{Tr}(\mathbf{H})} \text{GradWhitening}(\mathbf{H} \mathbf{O}),$$

noticing that $\frac{\|\mathbf{H}\mathbf{O}\|_1}{\text{Tr}(\mathbf{H})} = 1$, and $\text{GradWhitening}(\mathbf{H}\mathbf{O}) = \mathcal{P}(\mathbf{Q}\mathbf{A}\mathbf{Q}^\top \mathbf{O}) = \mathbf{Q}\mathbf{Q}^\top \mathbf{O} = \mathbf{O}$. Hence:

$$\mathbf{W}_{\text{whitened}}^{(1)} = \mathbf{O} - \eta^* \text{GradWhitening}(\mathbf{H}\mathbf{O}) = \mathbf{O} - \mathbf{O} = \mathbf{0} = \mathbf{W}^*.$$

Hence, the proof is complete. \square

F PROOF OF PROPOSITION 2

Proof. Since $\mathbf{W}_{\text{whitened}}^{(t)} \neq \mathbf{W}^*$, the square of the trace of the gradient $\text{Tr}[\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]^2$ must exceed some positive constant C_G , that is, $\text{Tr}[\mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]^2 > C_G$.

On the other hand, because:

1. The quadratic loss term $\text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}]$ is upper-bounded on $\mathbb{R}^{n \times n}$, and
2. $\text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}] \neq 0$ (due to $\mathbf{W}_{\text{whitened}}^{(t)} \neq \mathbf{W}^*$),

we have that there exists a positive number $0 < C_{\mathcal{L}}$ such that

$$0 < \text{Tr}[(\mathbf{W}_{\text{whitened}}^{(t)})^\top \mathbf{H}\mathbf{W}_{\text{whitened}}^{(t)}] < C_{\mathcal{L}}.$$

Finally, since the norm of \mathbf{H} is upper bounded, its trace must also be upper bounded by some constant C_H . Therefore, putting everything together, we have:

$$Q > \frac{C_G^2}{C_{\mathcal{L}} C_H}.$$

This inequality holds even as the condition number $\kappa \rightarrow +\infty$. \square

G PROOF OF PROPOSITION 3

To prove Proposition 3, we first generalize existing work on the convergence rate lower bound (via contraction factor) of gradient descent and Adam (we only consider $\beta_2 = 1$) under the same setting:

Theorem 3 (Contraction factor lower bound for gradient descent, generalized based on Zhang et al. (2024a)). *Consider the optimization problem in Equation (11). Let \mathbf{W}_{GD}^t be the output of GD after t steps. Then, for any step size η , there exists an initial condition such that the following lower bound on the contraction rate holds:*

$$\mathcal{L}(\mathbf{W}_{GD}^{t+1}) - \mathcal{L}^* \geq \left(1 - \frac{2}{\kappa + 1}\right) (\mathcal{L}(\mathbf{W}_{GD}^t) - \mathcal{L}^*),$$

where $\mathcal{L}^* = \mathcal{L}(\mathbf{W}^*)$. Furthermore, under optimal $\eta = \frac{2}{\lambda_1 + \lambda_m}$, the bound becomes tight regardless of the settings of \mathbf{H} , where λ_1 and λ_m are the largest and smallest eigen values of \mathbf{H} , respectively.

Proof. The proposition 1 in Zhang et al. (2024a) has shown that the lower bound holds for diagonal positive definite Hessian \mathbf{H} . To show that the lower bound holds for a general positive definite Hessian \mathbf{H} we will reformulate the problem to align with the setup in diagonal case (Proposition 1 of Zhang et al. (2024a)).

First, for any positive definite Hessian \mathbf{H} , we can perform an eigen decomposition $\mathbf{H} = \mathbf{U}\mathbf{S}\mathbf{U}^\top$, where \mathbf{U} is an orthogonal matrix and \mathbf{S} is a diagonal matrix containing the eigenvalues of \mathbf{H} . Define a change of variables $\mathbf{Z} = \mathbf{U}^\top \mathbf{W}$. Then, the optimization problem becomes

$$\mathcal{L}(\mathbf{Z}) = \frac{1}{2} \text{Tr}(\mathbf{Z}^\top \mathbf{S} \mathbf{Z}),$$

which reduces the problem to the case of a diagonal \mathbf{H} with condition number $\kappa = \frac{\lambda_1}{\lambda_m}$, where λ_1 and λ_m are the largest and smallest eigenvalues of \mathbf{H} , respectively.

Thus, by applying Proposition 1 of [Zhang et al. \(2024a\)](#) to this transformed problem, we conclude that there exists initial point such that the lower bound on the contraction rate

$$\mathcal{L}(\mathbf{W}_{GD}^{(t+1)}) - \mathcal{L}^* \geq \left(1 - \frac{2}{\kappa + 1}\right) \left(\mathcal{L}(\mathbf{W}_{GD}^{(t)}) - \mathcal{L}^*\right)$$

holds for the transformed variables \mathbf{Z} and, equivalently, for the original variables \mathbf{W} since the condition number is preserved under orthogonal transformations.

Therefore, the lower bound for gradient descent applies to any general positive definite Hessian \mathbf{H} provided the condition number κ remains unchanged.

Finally, under the optimal step size $\eta = \frac{2}{\lambda_1 + \lambda_m}$, the bound becomes tight regardless of the settings of \mathbf{H} . This is achieved by selecting η to minimize the contraction factor, aligning with well-known results regarding the optimal convergence rate of gradient descent on quadratic objectives ([Nesterov, 2013](#)).

This completes the proof of Theorem 3. \square

Corollary 1 (Lower bound on Adam ($\beta_2 = 1$)). *Consider the optimization problem in Equation (11). Assume the weight initialization \mathbf{W}^0 assigned zero probability to any set of zero Lebesgue measure in $\mathbb{R}^{m \times n}$. Let $\mathbf{W}_{Adam}^{(t)}$ be the parameter after t iterations of Adam with hyperparameters $\beta_1 = 0$ and $\beta_2 = 1$. Then, for any step size η , the following lower bound on the contraction rate holds:*

$$\mathcal{L}(\mathbf{W}_{Adam}^{(t+1)}) - \mathcal{L}^* \geq \left(1 - \frac{2}{\kappa'(\mathbf{W}^0) + 1}\right) \left(\mathcal{L}(\mathbf{W}_{Adam}^{(t)}) - \mathcal{L}^*\right),$$

where $\kappa'(\mathbf{W}^0)$ is the \mathbf{W}^0 -dependent condition number of the preconditioned Hessian $\text{diag}(|\mathbf{H}\mathbf{W}^0|^{-1})\mathbf{H}$, and $\mathcal{L}^* = \mathcal{L}(\mathbf{W}^*)$.

Proof. The update rule of Adam with $\beta_1 = 0$ and $\beta_2 = 1$ is given by [Zhang et al. \(2024a\)](#):

$$\mathbf{W}_{Adam}^{(t+1)} = \mathbf{W}_{Adam}^{(t)} + \eta \text{diag}(|\mathbf{H}\mathbf{W}^{(0)}|^{-1}) \mathbf{H}\mathbf{W}_{Adam}^{(t)}.$$

This can be interpreted as gradient descent with a preconditioned Hessian matrix $\text{diag}(|\mathbf{H}\mathbf{W}^{(0)}|^{-1})\mathbf{H}$. By applying Theorem 3, we conclude that the contraction rate for Adam under these settings satisfies the lower bound:

$$\mathcal{L}(\mathbf{W}_{Adam}^{(t+1)}) - \mathcal{L}^* \geq \left(1 - \frac{2}{\kappa + 1}\right) \left(\mathcal{L}(\mathbf{W}_{Adam}^{(t)}) - \mathcal{L}^*\right),$$

where κ is the condition number of the Hessian matrix \mathbf{H} .

Therefore, the proof is complete. \square

Next, We extend our Theorem 2 to block-diagonal Hessian case to prepare for discussions on group learning rates when comparing to Adam.

Corollary 2 (Upper Bound Convergence Rate of SWAN). *Consider the same quadratic loss function $\mathcal{L}(\mathbf{W}) = \frac{1}{2} \text{Tr}(\mathbf{W}^\top \mathbf{H}\mathbf{W})$ with \mathbf{H} being block-diagonal. That is, $\mathbf{H} = \text{diag}(\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_L)$, where each $\mathbf{H}_l \in \mathbb{R}^{m_l \times n_l}$ is a positive definite matrix for $l = 1, 2, \dots, L$, and $\sum_{l=1}^L m_l = m$ and $\sum_{l=1}^L n_l = n$. Assume the initialization distribution of $\mathbf{W}^{(0)}$ assigns zero probability to any zero measure set in $\mathbb{R}^{m \times n}$. Let $\mathbf{W}_{whitened}^{(t)}$ be the parameter matrix after t iterations of the SWAN optimizer defined in Theorem 2, with learning rate η . Then, under the conditions that ⁴:*

$$\|\mathbf{H}_l \mathbf{W}_l^{(t)}\|_1^2 - \text{Tr}(\mathbf{H}_l) \cdot \text{Tr}((\mathbf{W}_l^{(t)})^\top \mathbf{H}_l \mathbf{W}_l^{(t)}) \cdot \frac{2\lambda_{l,m_l}}{\lambda_{l,1} + \lambda_{l,m_l}} > 0,$$

⁴Note that according to Proposition 2, this is always achievable when \mathbf{H}_l is poorly-conditioned ($\frac{\lambda_{l,d_l}}{\lambda_{l,1}}$ is small enough). The lower interval above converges to zero, and one can simply pick e.g., $\eta = \min_{l \in [L]} \frac{\text{Tr}(\mathbf{H}_l \mathbf{W}_l^{(t)})}{\text{Tr}(\mathbf{H}_l)}$.

where $\lambda_{l,1}$ and λ_{l,m_l} are the largest and smallest singular value of \mathbf{H}_l , respectively; then there exists a proper learning rate η such that: with probability 1, the loss satisfies:

$$\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t+1)}) - \mathcal{L}^* < \max_{l \in [L]} \left(1 - \frac{2}{\kappa_l + 1} \right) \left(\mathcal{L}(\mathbf{W}_l^{(t)}) - \mathcal{L}^* \right), \quad (21)$$

where κ_l is the condition number of \mathbf{H}_l .

Proof. Applying the arguments in the proof of Theorem 2 to each block l , we have (for simplicity, we will drop the subscript “whitened” when there is no confusion):

$$\begin{aligned} \frac{\mathcal{L}(\mathbf{W}_l^{(t+1)}) - \mathcal{L}^*}{\mathcal{L}(\mathbf{W}_l^{(t)}) - \mathcal{L}^*} &= 1 - \frac{\mathcal{L}(\mathbf{W}_l^{(t)}) - \mathcal{L}(\mathbf{W}_l^{(t+1)})}{\mathcal{L}(\mathbf{W}_l^{(t)}) - \mathcal{L}^*} \\ &= 1 - \frac{\eta \|\mathbf{H}_l \mathbf{W}_l^{(t)}\|_1 - \frac{\eta^2}{2} \text{Tr}(\mathbf{H}_l)}{\frac{1}{2} \text{Tr}[(\mathbf{W}_l^{(t)})^\top \mathbf{H}_l \mathbf{W}_l^{(t)})]}. \end{aligned}$$

It is straightforward to verify via the quadratic formula that if one chooses η satisfying:

$$\begin{aligned} &\frac{\|\mathbf{H}_l \mathbf{W}_l^{(t)}\|_1 - \sqrt{\|\mathbf{H}_l \mathbf{W}_l^{(t)}\|_1^2 - \text{Tr}(\mathbf{H}_l) \cdot \text{Tr}((\mathbf{W}_l^{(t)})^\top \mathbf{H}_l \mathbf{W}_l^{(t)}) \cdot \frac{2\lambda_{l,m_l}}{\lambda_{l,1} + \lambda_{l,m_l}}}}{\text{Tr}(\mathbf{H}_l)} < \eta \\ &< \frac{\|\mathbf{H}_l \mathbf{W}_l^{(t)}\|_1 + \sqrt{\|\mathbf{H}_l \mathbf{W}_l^{(t)}\|_1^2 - \text{Tr}(\mathbf{H}_l) \cdot \text{Tr}((\mathbf{W}_l^{(t)})^\top \mathbf{H}_l \mathbf{W}_l^{(t)}) \cdot \frac{2\lambda_{l,m_l}}{\lambda_{l,1} + \lambda_{l,m_l}}}}{\text{Tr}(\mathbf{H}_l)}, \end{aligned}$$

then we have:

$$\frac{\frac{1}{2} \text{Tr}[(\mathbf{W}_l^{(t)})^\top \mathbf{H}_l \mathbf{W}_l^{(t)})]}{\eta \|\mathbf{H}_l \mathbf{W}_l^{(t)}\|_1 - \frac{\eta^2}{2} \text{Tr}(\mathbf{H}_l)} < \frac{\kappa_l + 1}{2}.$$

Rearranging, we obtain:

$$\frac{\mathcal{L}(\mathbf{W}_l^{(t+1)}) - \mathcal{L}^*}{\mathcal{L}(\mathbf{W}_l^{(t)}) - \mathcal{L}^*} < 1 - \frac{2}{\kappa_l + 1}.$$

Since \mathbf{H} is block-diagonal, the updates for each block l are independent. Summing the loss over all blocks, we obtain:

$$\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t+1)}) - \mathcal{L}^* = \sum_{l=1}^L \left(\mathcal{L}(\mathbf{W}_l^{(t+1)}) - \mathcal{L}^* \right) < \sum_{l=1}^L \left(1 - \frac{2}{\kappa_l + 1} \right) \left(\mathcal{L}(\mathbf{W}_l^{(t)}) - \mathcal{L}^* \right).$$

Taking the maximum contraction factor across all blocks:

$$\begin{aligned} \mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t+1)}) - \mathcal{L}^* &< \max_{l \in [L]} \left(1 - \frac{2}{\kappa_l + 1} \right) \sum_{l=1}^L \left(\mathcal{L}(\mathbf{W}_l^{(t)}) - \mathcal{L}^* \right) \\ &= \max_{l \in [L]} \left(1 - \frac{2}{\kappa_l + 1} \right) \left(\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t)}) - \mathcal{L}^* \right). \end{aligned}$$

Thus, the overall contraction factor for the SWANOptimizer is:

$$\rho_{\text{SWAN}} = \max_{l \in [L]} \left(1 - \frac{2}{\kappa_l + 1} \right).$$

□

Finally, we are ready to prove Proposition 3:

Proposition 2 (GradWhitening with single lr vs Adam with tuned group lr). *Consider the optimization problem Equation (11). Assume \mathbf{H} is block-diagonal, i.e., $\mathbf{H} = \text{diag}(\mathbf{H}_1, \mathbf{H}_2, \dots, \mathbf{H}_L)$, where each $\mathbf{H}_l \in \mathbb{R}^{m_l \times m_l}$ is a positive definite matrix for $l = 1, 2, \dots, L$, and $\sum_{l=1}^L m_l = m$. Assuming for GradWhitening we use one global learning rate for all parameters; and for Adam, we use the optimally chosen group learning rate η_l and initial condition w_0 for each block \mathbf{H}_l . Assume either if i) certain regularity conditions are met (see proof in Appendix), or ii), if \mathbf{H} is poorly-conditioned (its condition number is large enough). Then: regardless of its initialization, GradWhitening with a properly chosen learning rate will still have a strictly better convergence speed (i.e., smaller contraction factor) across all blocks $l \in [L]$ than Adam ($\beta_1 = 0, \beta_2 = 1$) under optimal group-wise learning rates and initial condition.*

Proof. For simplicity, we will drop the subscript “whitened” when there is no confusion. Let $\kappa'_l(\mathbf{W}_l^{(0)})$ denote the $\mathbf{W}_l^{(0)}$ -dependent condition number of the l -th block preconditioned Hessian $\text{diag}(\left| \mathbf{H}_l \mathbf{W}_l^{(0)} \right|^{-1}) \mathbf{H}_l$. Let $\lambda_{l,m_l}(\mathbf{W}_l^{(0)})$ and $\lambda_{l,1}(\mathbf{W}_l^{(0)})$ be the smallest and largest eigenvalues of $\text{diag}(\left| \mathbf{H}_l \mathbf{W}_l^{(0)} \right|^{-1}) \mathbf{H}_l$, respectively. Then,

Case 1: Under the conditions that:

1. **Existence of roots:** $\forall l \in [L]$,

$$\|\mathbf{H}_l \mathbf{W}_l^{(t)}\|_1^2 - \text{Tr}(\mathbf{H}_l) \cdot \text{Tr}((\mathbf{W}_l^{(t)})^\top \mathbf{H}_l \mathbf{W}_l^{(t)}) \cdot \frac{2\lambda_{l,m_l}(\mathbf{W}_l^{(0)})}{\lambda_{l,1}(\mathbf{W}_l^{(0)}) + \lambda_{l,m_l}(\mathbf{W}_l^{(0)})} > 0,$$

and

2. **Overlap condition:**

$$\begin{aligned} & \min_{l \in [L]} \frac{\text{Tr}(\mathbf{H}_l \mathbf{W}_l^{(t)}) + \sqrt{\text{Tr}(\mathbf{H}_l \mathbf{W}_l^{(t)})^2 - \text{Tr}(\mathbf{H}_l) \cdot \text{Tr}((\mathbf{W}_l^{(t)})^\top \mathbf{H}_l \mathbf{W}_l^{(t)}) \cdot \frac{2\lambda_{l,m_l}(\mathbf{W}_l^{(0)})}{\lambda_{l,1}(\mathbf{W}_l^{(0)}) + \lambda_{l,m_l}(\mathbf{W}_l^{(0)})}}}{\text{Tr}(\mathbf{H}_l)} \\ & > \max_{l \in [L]} \frac{\text{Tr}(\mathbf{H}_l \mathbf{W}_l^{(t)}) - \sqrt{\text{Tr}(\mathbf{H}_l \mathbf{W}_l^{(t)})^2 - \text{Tr}(\mathbf{H}_l) \cdot \text{Tr}((\mathbf{W}_l^{(t)})^\top \mathbf{H}_l \mathbf{W}_l^{(t)}) \cdot \frac{2\lambda_{l,m_l}(\mathbf{W}_l^{(0)})}{\lambda_{l,1}(\mathbf{W}_l^{(0)}) + \lambda_{l,m_l}(\mathbf{W}_l^{(0)})}}}{\text{Tr}(\mathbf{H}_l)}. \end{aligned}$$

Then, there exists a global learning rate η , such that for all $l \in [L]$,

$$\frac{\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t+1)})_l - \mathcal{L}_l^*}{\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t)})_l - \mathcal{L}_l^*} < 1 - \frac{2}{\kappa'_l(\mathbf{W}_l^{(0)}) + 1} \leq \frac{\mathcal{L}(\mathbf{W}_{\text{Adam}}^{(t+1)})_l - \mathcal{L}_l^*}{\mathcal{L}(\mathbf{W}_{\text{Adam}}^{(t)})_l - \mathcal{L}_l^*}.$$

Case 2: If \mathbf{H} is poorly-conditioned, i.e., $\frac{\lambda_{l,m_l}(\mathbf{W}_l^{(0)})}{\lambda_{l,1}(\mathbf{W}_l^{(0)})} \rightarrow 0$, then Proposition 2 asserts that the following term

$$\max_{l \in [L]} \frac{\text{Tr}(\mathbf{H}_l \mathbf{W}_l^{(t)}) - \sqrt{\text{Tr}(\mathbf{H}_l \mathbf{W}_l^{(t)})^2 - \text{Tr}(\mathbf{H}_l) \cdot \text{Tr}((\mathbf{W}_l^{(t)})^\top \mathbf{H}_l \mathbf{W}_l^{(t)}) \cdot \frac{2\lambda_{l,m_l}(\mathbf{W}_l^{(0)})}{\lambda_{l,1}(\mathbf{W}_l^{(0)}) + \lambda_{l,m_l}(\mathbf{W}_l^{(0)})}}}{\text{Tr}(\mathbf{H}_l)} \rightarrow 0,$$

and one can simply choose, for example, $\eta = \min_{l \in [L]} \frac{\text{Tr}(\mathbf{H}_l \mathbf{W}_l^{(t)})}{\text{Tr}(\mathbf{H}_l)}$. Under this choice of η , we still have

$$\frac{\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t+1)})_l - \mathcal{L}_l^*}{\mathcal{L}(\mathbf{W}_{\text{whitened}}^{(t)})_l - \mathcal{L}_l^*} < 1 - \frac{2}{\kappa'_l(\mathbf{W}_l^{(0)}) + 1} \leq \frac{\mathcal{L}(\mathbf{W}_{\text{Adam}}^{(t+1)})_l - \mathcal{L}_l^*}{\mathcal{L}(\mathbf{W}_{\text{Adam}}^{(t)})_l - \mathcal{L}_l^*}$$

for all $l \in [L]$. □

H PROOF OF PROPOSITION 1

Proof. First, define $\mathbf{V} \in \mathbb{R}^{M_C \times n}$ as $\mathbf{V} := \mathbf{U}_C^\top \mathbf{W}$, and consider the Hessian with respect to \mathbf{V} instead of \mathbf{W} . Notice that although the loss function \mathcal{L} is unknown, its first-order derivatives are known. Specifically, they are given by:

$$\frac{\partial \mathcal{L}}{\partial v_{lk}} = \dot{v}_{lk} = \mathbb{E}_{q=m} [g_{h_k} x_l] e^{\frac{1}{2} \sum_s v_{ls}^2}.$$

Therefore, the second-order derivatives, i.e., the Hessian matrix $\mathbf{H}(\mathbf{V})$, are:

$$\mathbf{H}(\mathbf{V})_{lk, l'k'} = \frac{\partial^2 \mathcal{L}}{\partial v_{lk} \partial v_{l'k'}} = \frac{\partial \left[\mathbb{E}_{q=m} [g_{h_k} x_l] e^{\frac{1}{2} \sum_s v_{ls}^2} \right]}{\partial v_{l'k'}} = \dot{v}_{lk} v_{l'k'} \delta_{ll'},$$

where $\delta_{ll'}$ is the Kronecker delta, which is 1 if $l = l'$ and 0 otherwise.

Based on Lemma B.6 in [Zhao et al. \(2024a\)](#), as $t \rightarrow \infty$, there exists an index subset $O_l \subset \{1, \dots, M_C\}$ such that:

$$v_{l^*k} \gg v_{lk}, \quad \dot{v}_{l^*k} \gg \dot{v}_{lk}, \quad \forall l^* \in O_l, l \notin O_l, \forall k.$$

Consequently,

$$\mathbf{H}(\mathbf{V})_{l^*k, l^*k'} \gg \mathbf{H}(\mathbf{V})_{lk, l'k'}, \quad \forall l^* = l'^* \in O_l, l, l' \notin O_l, \forall k, k'.$$

After normalization, as $t \rightarrow \infty$, we have

$$\frac{\mathbf{H}(\mathbf{V})_{lk, l'k'}}{\sum_{l, l'} \mathbf{H}(\mathbf{V})_{lk, l'k'}} \xrightarrow{t \rightarrow \infty} \begin{cases} 1, & \text{if } l = l' \in O_l, \\ 0, & \text{otherwise.} \end{cases}$$

Reverting back to the \mathbf{W} space, we have

$$\mathbf{H}(\mathbf{W}) = (\mathbf{I}_K \otimes \mathbf{U}_C) \mathbf{H}(\mathbf{V}) (\mathbf{I}_K \otimes \mathbf{U}_C)^\top,$$

where \otimes denotes the Kronecker product and \mathbf{I}_K is the identity matrix of appropriate dimensions.

Therefore, for all $1 \leq s, s' \leq d$ and $1 \leq k, k' \leq n$, we obtain

$$\frac{\mathbf{H}(\mathbf{W})_{sk, s'k'}}{\sum_{s, s'} \mathbf{H}(\mathbf{W})_{sk, s'k'}} = \frac{\mathbf{H}(\mathbf{W})_{sk', s'k'}}{\sum_{s, s'} \mathbf{H}(\mathbf{W})_{sk', s'k'}} \quad \text{as } t \rightarrow \infty.$$

This holds for all $1 \leq s, s' \leq d$ and $1 \leq k, k' \leq n$. \square

I EMPIRICAL VERIFICATION OF THEORETICAL INSIGHTS FROM SECTION 5 REGARDING GradNorm AND GradWhitening

I.0.1 DOES GradNorm STABILIZE GRADIENT DISTRIBUTIONS OF SGD?

To examine whether GradNorm stabilizes the distribution of the stochastic gradients as suggested by Theorem 1, we conduct controlled experiments using a scaled-down LLaMA-based model (about 10 million parameters) ([Lialin et al., 2023](#)), trained on the C4 dataset. Our goal is to measure how GradNorm affects the distribution of stochastic gradients over multiple training steps. Specifically, we employ a small-scale LLaMA-based model with approximately 10 million parameters ([Lialin et al., 2023](#)). Training is conducted on the C4 dataset ([Raffel et al., 2020](#)).

Baselines We compare:

- **Standard training:** This uses an SGD optimizer with a learning rate of 5×10^{-4} and a linear learning rate scheduler, including a 10% warm-up of total training steps (10,000 steps).
- **GradNorm-processed training:** This applies GradNorm to pre-process the stochastic gradient before the parameter update. All other settings match the standard training baseline.

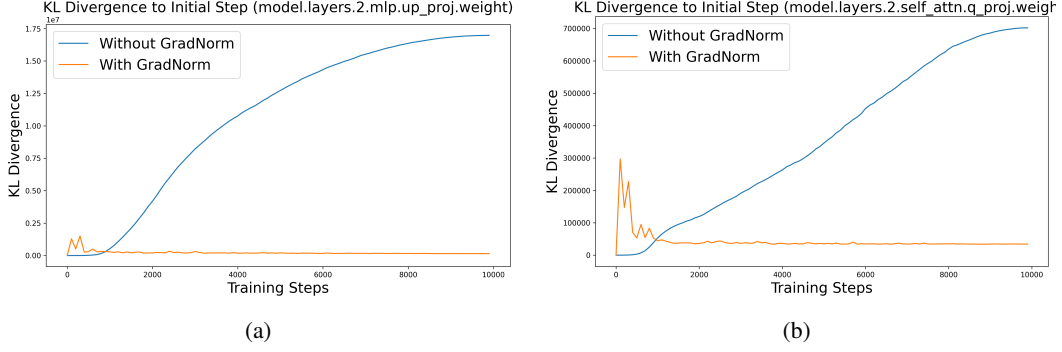


Figure 9: KL divergence comparison of gradient distributions against initial gradient distribution across training Steps. We use the projection weights in attention and MLP modules of the second layer as an example. The plots compare standard training with GradNorm-augmented training. Lower KL divergence values indicate greater stability in gradient distributions.

Methodology We measure gradient statistics in the presence of mini-batch noise. At step $t = 0$, we sample 16 additional mini-batches (batch size 64 each) and compute the mean and standard deviation of the corresponding raw or GradNorm gradients in each batch, and obtain the approximated initial gradient distribution. After each training step of baseline methods, we perform the same procedure and calculate the Kullback-Leibler divergence between the resulting gradient distributions and the initial gradient distributions. This process tracks how the gradient distribution changes over time.

Results Figure 9 shows the KL divergence of gradient distributions for standard and GradNorm-augmented training, relative to the corresponding initial approximated distributions. Apart from early spikes, GradNorm reduces fluctuations in the gradient distribution throughout training.

I.0.2 DOES GRADWHITENING COUNTERACTS LOCAL CURVATURE AND PROVIDE FAST CONVERGENCE ON ILL-CONDITIONED PROBLEMS?

This subsection evaluates the optimization performance of gradient descent when combined with GradWhitening. We use three classic problem settings:

- **High-dimensional quadratic optimization.** A quadratic problem of the form in Equation (11), where $\mathbf{W} \in \mathbb{R}^{50 \times 50}$.
- **Ill-conditioned quadratic optimization.** Same setup as above, but with a deliberately chosen ill-conditioned \mathbf{H} .
- **Non-convex optimization with multiple local optima.** We use the multivariate Rastrigin function:

$$f(\mathbf{W}) = m^2 \mathbf{A} + \frac{1}{2} \text{Tr}[\mathbf{W}^\top \mathbf{W}] - \mathbf{A} \sum_{ij} \cos(2\pi W_{ij}),$$

where \mathbf{W} is an $m \times m$ matrix and $m = 50$. This function has 10^{m^2} possible local optima.

Baselines We compare five methods on all three problems: gradient descent (GD) with the theoretical optimal learning rate in Theorem 3, Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a hand-tuned learning rate, Newton’s method with a tuned learning rate, and two GradWhitening-based variants (with and without orthogonal initialization). This is to verify, under orthogonal initialization, GradWhitening-processed GD behaves similarly to Newton’s method, as discussed in Section 5.3 and Theorem 2. All methods share the same initialization, except for the orthogonal GradWhitening variant, which projects the initial parameters onto an orthogonal matrix.

Results From Figure 10 (a)–(c), we summarize the following outcomes:

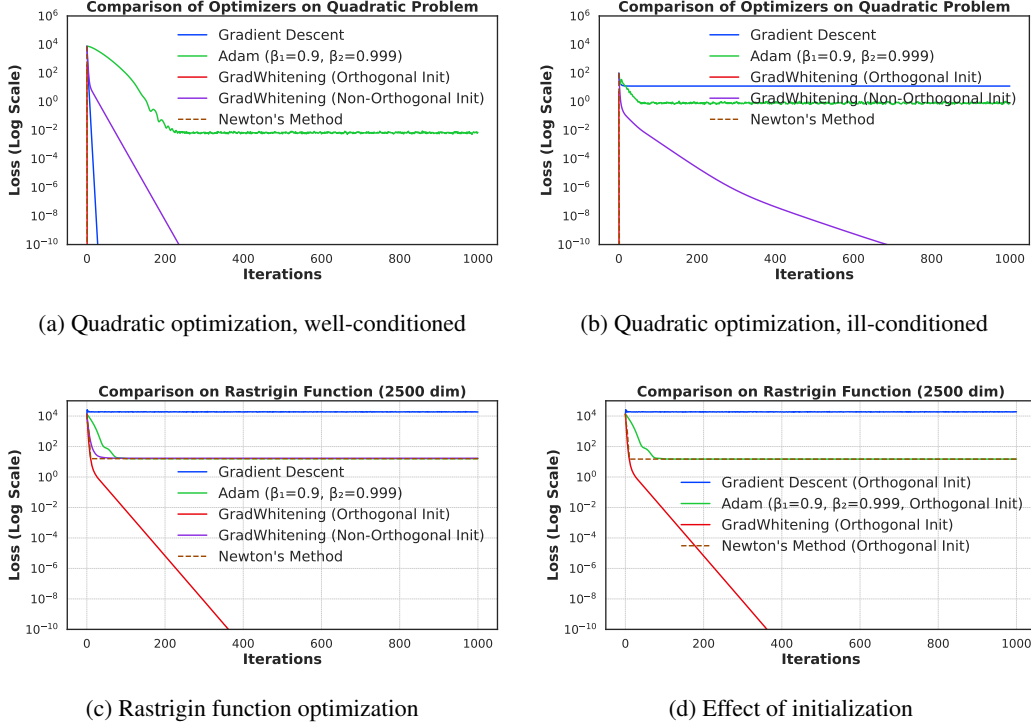


Figure 10: Comparison of convergence rate of different methods on quadratic and non-convex optimization problems. (a): 2500-dimensional quadratic optimization with well-conditioned \mathbf{H} . (b): 2500-dimensional quadratic optimization with ill-conditioned \mathbf{H} . (c): 2500-dimensional Rastrigin function optimization. (d): 2500-dimensional Rastrigin function optimization, but forcing all methods to use the same orthogonal initial location.

- **Quadratic problems (Figure 10 (a) and (b)).** GradWhitening with orthogonal initialization and Newton’s method converge to optimum in one step, aligning with the theoretical predictions in Section 5.3 and Theorem 2.
- **Well- vs. ill-conditioned cases (Figure 10 (a) and (b)).** In the well-conditioned setting (a), standard GD outperforms both Adam and GradWhitening (non-orthogonal initialization). In the ill-conditioned setting (b), GradWhitening (non-orthogonal initialization) outperforms GD by a large margin, while GD experiences slow convergence.
- **Comparison with Adam (Figure 10 (a)–(c)).** In all three settings, GradWhitening with non-orthogonal initialization consistently outperforms Adam, consistent with Proposition 3.
- **Rastrigin function (Figure 10 (c)).** On this non-convex problem, GradWhitening performs comparably to Newton’s method, with or without orthogonal initialization.
- **Effect of initialization (Figure 10 (d))** Furthermore, we force all methods to share the same orthogonalized initialization. As shown by the result, GradWhitening GD with orthogonal initialization still consistently outperforms all baselines, confirming that this initialization is only beneficial to GradWhitening GD among all baselines.

J IMPLEMENTATION DETAILS

General setup We describe the implementation setups for SWAN used in LLM pre-training tasks. To enable a more straightforward and comparable analysis, we simply replicate the setting of Zhao et al. (2024a), under exactly the same model configs and optimizer hyperparameter configs, whenever possible. This includes the same model architecture, tokenizer, batch size, context length, learning rate scheduler, learning rates, subspace scaling, etc.

Table 6: Reproduced results.

	130M	350M	1.3 B
Adam (reproduced)	24.44 (0.75G)	19.24 (2.05G)	16.44 (7.48G)
Apollo-mini (reproduced)	23.97 (0.43G)	17.60 (0.93G)	14.37 (2.98G)
Galore (reproduced)	24.67 (0.57G)	19.74 (1.29G)	15.89 (4.43G)
r of low-rank methods	256	256	512
Training Steps	20K	60K	100K

Precision All baselines uses BF16 for model weights, gradients, and optimizer states storage. For all SWAN variants, we use BF16 for model weights and gradients. For the GradWhitening step of SWAN-0 and SWAN[†] we use FP32 to whiten the BF16 gradients and then convert it back to BF16. We found that this helps to improve training stability and performance. However, for the NSDS scheme of SWAN[‡] we observe that FP32 does not offer performance boost over BF16 (Appendix B.4), therefore we stick with BF16.

Learning rate scheduling we use exactly the same scheduler as in Zhao et al. (2024a), with the exception of SWAN-0, which does not require any learning rate warmup. Therefore, for SWAN-0, we directly start with maximum learning rate, and enter the learning rate decay phase, using the same decay parameters as Zhao et al. (2024a).

Reproducing baseline results Most baseline results are cited from respective papers (Zhao et al., 2024a; Zhu et al., 2024) as we share the exact same setup. We also tried to reproduce their results using the same opensourced code, and generally obtain slightly worse results for Galore, Apollo and Adam for larger models (350M and 1B, see Table 6). Therefore in the main paper we only compare with the official results. For the reproduced results of Adam, we specify the details below as it was not disclosed in Zhao et al. (2024a). We use same learning rate tuning procedure as suggested by Zhao et al. (2024a) (i.e., performing grid search over $\{0.01, 0.005, 0.001, 0.0005, 0.0001\}$). We found that the optimal learning rates for Adam is 0.001. The only exception is that for a model of size 1.3B: as we already know that a larger model requires smaller learning rates, we conduct a learning search for Adam over a smaller but more fine-grained grid of $\{0.001, 0.0007, 0.0005, 0.0003, 0.0001\}$. As a result, the optimal learning rate found for Adam on 1.3B is 0.0007. Finally, one baseline that does not exist in the literature is the Momentum + GradWhitening (Muon-like optimizer without Nesterov acceleration) baseline; and we report our own results. We start from the default learning rates used by Muon Jordan et al. (2024) and tuned them over a grid of 0.01, 0.02, 0.03, 0.04, 0.05.

SWAN Settings and hyperparameters Since SWAN utilizes matrix-level operations on gradients, it can only be applied to 2D parameters. Therefore, in our experiments, we only apply SWAN on all linear projection weights in transformer blocks. Similar to Galore (Zhao et al., 2024a), the rest of the non-linear parameters still uses Adam as the default choice. Therefore, we follow the learning rate setup of Galore, where we fix some global learning rate across all model sizes and all modules. Then, for the linear projection modules where SWAN is applied, we simply apply a scaling factor α on top of the global learning rate. For all SWAN variants, we adopt a *lazy-tuning approach* (hyperparameters are set without extensive search), as detailed below. This helps to reduce the possibility of unfair performance distortion due to excessive tuning.

- **SWAN-0** uses naive NS-iteration for whitening, disabled learning rate warmup, and use similar learning rates optimized for Adam. We fix the global learning rate to be the same as Adam, and fix $\alpha = 1$. The only exception is the 1.3 B case. This is because we observe that the optimal learning rate of Adam under 1.3B becomes smaller than 0.001, hence we also reduce the learning rate on SWAN, where we used $\alpha = 0.3$, resulting an effective learning rate of 0.0003. To summarize the hyperparameter of **SWAN-0** is set to be similar to Adam, without any tuning. This is to demonstrate the robustness of SWAN series optimizers and their capability to work out-of-the-box as a replacement for Adam.
- **SWAN[†]**, is the vanilla version of our method, in which we enabled learning rate warmup, and allowed the use of optimized learning rates that largely differ from Adam. We notice

that SWAN allows larger learning rates than Adam. We use a global learning rate of 0.02, as well as the scaling factor $\alpha = 0.05$. This is selected by simply searching the learning rate over a constraint grid $\{0.01, 0.02, 0.05\}$, and then setting $\alpha = 0.05$ such that the effective learning rate is scaled back to 0.001. There is no guarantee that this heuristic rule is optimal; but we found that this usually does not make a run fail (e.g., with loss divergence).

- Finally, **SWAN[‡]**, the most efficient version of SWAN that employs the proposed NSDS scheme for fast whitening (section 4.2). Similar to **SWAN[†]**, we use the same global learning rate of 0.02, as well as the scaling factor $\alpha = 0.05$ across all model sizes. We suspect with more careful tuning, its performance can be significantly improved; however, this is out of the scope of the paper.

The configurations of GradWhitening is discussed next.

Implementation of GradWhitening For GradWhitening, before N-S iteration, we further normalize its input matrix by its Frobenius norm. This is applied in all our ablation studies as well, regardless of whether the gradient is processed by GradNorm. For our proposed Newton-Schulz with Diagonal Substitution (NSDS) used in **SWAN[‡]**, we only run it for 2 steps across all model sizes. We found that using a step size $\beta \neq 0.5$ in the GradWhitening operator (Algorithm 2) can improve its convergence. We set $\beta = 0.4$. We found that NSDS is generally robust to β as long as it is not too large; and our specific choice of parameters usually already gives satisfactory performance in LLM pretraining. For the naive N-S iteration used in **SWAN-0** and **SWAN[†]**, we run 10 steps which is usually sufficient. We set $\beta = 0.8$. The naive NS is run in FP32 precision, on top of BF16 gradients and weights; while the NSDS is run in BF16.

Computational Overhead. Below, we discuss the computational overhead of Naive Newton-Schulz. In practice, we only run the naive Newton-Schulz for ≤ 10 iterations, which corresponds to ≤ 50 matrix multiplications. These matrix multiplications are in general GPU friendly, hence for the task of training LLMs, the batch size is the more dominant factor for compute. For example QWen 14B (Bai et al., 2023) has 4M batch size vs a model dimension $d_{\text{model}} = 5120$, DeepSeek (Bi et al., 2024) 67B has 6M batch size vs $d_{\text{model}} = 8192$, and LLama 3 (Dubey et al., 2024) 405B has 4-16M batch size vs $d_{\text{model}} = 16384$. To estimate the computational overhead, assuming the N-S iteration involves approximately $50 \times d_{\text{model}}^3$ FLOPs. In contrast, the primary training cost scales with the batch size and is proportional to $\text{batch_size} \times d_{\text{model}}^2$ FLOPs. In those examples, the estimated computational overhead of Newton-Schulz is typically below $\leq 7\%$. A similar estimation has been given in (Jackson, 2023) ($\leq 5\%$), as well as in (Jordan et al., 2024) ($\leq 1\%$).

However, note that whether the above analysis hold for large scale, distributed LLM training is still unclear. This is due to a few factors. First, the N-S iteration is after all a $\mathcal{O}(m^3)$ complexity operation and might need to scale as model size increases. Second, the distributed computation of N-S steps needs extra non-trivial effort, which brings additional infrastructure challenges. This motivates us to propose the NSDS scheme, which enables Adam-level throughput without performance compromise.