Revealing Modular Gradient Noise Imbalance in LLMs: Calibrating Adam via Signal-to-Noise Ratio

Ziqing Wen[1] , **Zhouyang Liu**[1] , **Jiahuan Wang**[1] , **Ping Luo**[1] , **Li Shen**[1] , **Dongsheng Li**[1] and **Tao Sun**[1*]
[1]College of Computer Science and Technology,
National University of Defense Technology, Changsha, Hunan, China
{zqwen, liuzhouyang20,luoping,wangjiahuan,lishen,dsli}@nudt.edu.cn, suntao.saltfish@outlook.com

Abstract

The impressive performance of large language models (LLMs) arises from their massive scale and heterogeneous module composition. However, this structural heterogeneity introduces additional optimization challenges. While adaptive optimizers such as Adam(W) provide per-parameter adaptivity, they do not explicitly account for module-level gradient heterogeneity, resulting in slower convergence, suboptimal performance, or training instability. Existing approaches typically rely on manually tuned module-specific learning rates or specific optimization strategies, which are computationally costly and difficult to generalize across tasks or models. To establish a more principled approach, we first analyze the noise-damping behavior of Adam in high-noise modules and introduce **Module-wise Learning Rate Scaling via SNR (`MoLS`)**. `MoLS` estimates module-level SNRs to scale Adam updates, allowing automated module-wise learning rate allocation without manual tuning. Empirical results through multiple LLM training benchmarks demonstrate that `MoLS` improves convergence speed and generalization, achieving performance comparable to carefully tuned module-specific learning rates, while remaining compatible with memory-efficient training algorithms.

1 Introduction

Large Language Models (LLMs) are built upon functionally distinct modules, typically consisting of stacked Transformer blocks [Vaswani *et al.*, 2017; Radford *et al.*, 2019; Touvron *et al.*, 2023]. This modularity allows for the efficient organization of specialized functional components, such as self-attention and feed-forward networks. By leveraging these functionally diverse modules, LLMs gain the ability to handle complex natural language processing tasks. However, the sheer depth and structural diversity of these modules introduce significant optimization complexities [Zhang *et al.*, 2024]. To handle this complexity, adaptive optimization methods, particularly Adam(W) [Kingma, 2014; Loshchilov and Hutter, 2017], are widely used in practice.

Despite its effectiveness, Adam's element-wise adaptivity does not explicitly account for the structural heterogeneity across modules. Some modules exhibit consistently larger or more variable gradients, leading to scale mismatches that Adam's adaptivity cannot fully correct, resulting in slow convergence or unstable training [Li *et al.*, 2025]. Recent studies indicate that different modules in LLMs exhibit markedly different gradient and optimization properties [Liu *et al.*, 2020; Ormaniec *et al.*, 2024; Kunstner *et al.*, 2024; Zhang *et al.*, 2025]. Building on this heterogeneity, Zeng *et al.* 2022 and Wang *et al.* 2025a have suggested the use of module-specific learning rates in Adam to better accommodate these differences and enhance convergence stability.

While these studies recognize the need for module-wise learning rate configuration even with Adam, they provide no principled or quantitative method to determine the specific scaling ratios. As a result, these ratios are treated as hyperparameters, requiring exhaustive grid searches—an approach that is computationally expensive, infeasible for large-scale models, and theoretically ungrounded. Consequently, current methods rely on fixed heuristics that ignore module-specific gradient statistical properties, leading to configurations that are difficult to generalize across tasks or models. In essence, while existing literature recognizes the need for module-wise learning rate scaling, determining optimal magnitudes remains an open, heuristic-driven challenge.

Motivated by these limitations and the strong correlation between Adam optimizer updates and gradient signal-to-noise ratio (SNR) [Orvieto and Gower, 2025], we analyze the heterogeneity of LLM architectures from the SNR perspective. Our analysis demonstrates that different modules experience systematically imbalanced effective signal strengths, offering a principled explanation for a noise-damping deficit of Adam in heterogeneous architectures that has not been systematically studied. This insight suggests that scaling learning rates according to each module's SNR could naturally rebalance updates, motivating our approach: **Module-wise Learning Rate Scaling via SNR (`MoLS`)**. `MoLS` estimates module-level SNRs during a brief calibration phase and applies the resulting scaling factors to module-wise learning rates. By doing so, `MoLS` provides an automated, lightweight

*Corresponding author.

solution to reduce signal-to-noise imbalance across modules, reducing manual tuning and improving both convergence and generalization. Our main contributions are listed as follows:

- **SNR Analysis:** We study module-level gradient statistics from the perspective of SNR and analyze how they relate to learning-rate scaling. Compared to prior works that rely on Hessian-based or sharpness-related measures, our analysis reveals systematic imbalances of SNR across LLM modules and offers an interpretable explanation for the noise-dominated behavior observed in high-noise components.
- **SNR-Based Scaling Strategy:** Based on the SNR analysis, we introduce a learning-rate scaling strategy based on module-wise SNR estimates. SNRs are computed during a short warm-up phase, after which learning rates are gradually adjusted toward their target values. This design avoids loss instability caused by abrupt learning rate scaling, introduces minimal additional computational overhead, and is fully compatible with standard LLM training paradigms.
- **Evaluation:** We evaluate the effectiveness of our method on a range of LLM benchmarks, where MoLS effectively improves convergence speed and generalization performance. Moreover, MoLS achieves performance comparable to manually fine-tuned module-wise learning rates, and is compatible with other memory-efficient methods such as Adam-mini [Zhang *et al.*, 2025] and LoRA [Hu *et al.*, 2022].

Overall, MoLS provides a statistically grounded and efficient approach to module-wise learning-rate scaling in LLM optimization.

2 Related Works

Efficient LLM Optimizers. Adaptive optimizers such as Adam(W) [Kingma, 2014; Loshchilov and Hutter, 2017] are commonly used in LLMs training. Existing works primarily focus on improving the performance and memory efficiency of Adam, including Nesterov-style momentum [Dozat, 2016], confidence-guided updates [You *et al.*, 2020], parameter-noise–normalized SGD [Xu *et al.*, 2024], matrix-based preconditioners [Jordan *et al.*, 2024; Vyas *et al.*, 2024], powered gradients [Wang *et al.*, 2025b], clustering-based learning rates [Li *et al.*, 2025], block-shared normalizers [Zhang *et al.*, 2025], and low-rank projection methods [Zhao *et al.*, 2024; Zhu *et al.*, 2024]. While effective, these approaches either ignore the heterogeneity among LLM modules or depend on complex optimization strategies.

Heterogeneity of LLM modules. LLMs contain multiple functional modules that exhibit pronounced heterogeneity in their optimization characteristics. Zeng *et al.* 2022 identified a significant inconsistency in the gradient scales between the Embedding and other modules. Zhang *et al.* 2024 analyzed the block-diagonal structure of the Hessian across LLM modules, revealing substantial variation in curvature statistics. Focusing on self-attention, Ormaniec *et al.* 2024 examined Hessian blocks corresponding to the Query/Key and Value/Output projections and identified clear differences between these submodules. Building on this, Wang *et al.* 2025a analysed the module-wise sharpness in LLMs and argued that heterogeneous modules may benefit from distinct learning rates to accommodate their optimization landscapes. However, assigning module-wise learning rates makes hyperparameter tuning considerably more difficult.

Positioning of MoLS. MoLS is complementary to existing adaptive optimizers. Instead of altering Adam's update rule or introducing computationally expensive approximations, MoLS addresses a comparatively underexplored challenge: the imbalance in effective signal strength across model modules arising from stochastic gradient noise. By estimating module-wise signal-to-noise ratios, MoLS offers a principled and lightweight mechanism for calibrating learning rates at the module level. From this perspective, MoLS serves as an automated and statistically grounded alternative to manual module-wise learning rate tuning.

3 Motivation

In this section, we first introduce the update rules of Adam [Kingma, 2014] and analyze the updates of Adam from a Signal-to-Noise Ratio (SNR) perspective. We subsequently introduce our **Module-wise LR Scaling via SNR (MoLS)**.

3.1 The Noise-Damping Deficit in Adam

Adam(W) [Kingma, 2014; Loshchilov and Hutter, 2017] has become the standard optimizer for training LLMs. Adam maintains the first and second moments for each parameter element in weight $\mathbf{w}$ as follows

$$\begin{aligned} \mathbf{m}^{t+1} &= \beta_1 \cdot \mathbf{m}^t + (1-\beta_1) \cdot \mathbf{g}^t, \\ \mathbf{v}^{t+1} &= \beta_2 \cdot \mathbf{v}^t + (1-\beta_2) \cdot \mathbf{g}^{\odot 2}, \end{aligned} \tag{1}$$

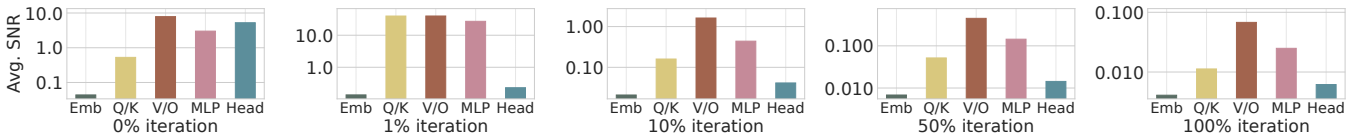
where $\beta_1, \beta_2 \in [0, 1)$ represent the decay rates, and $\odot$ denotes the Hadamard multiplication. Adam updates weight by

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t \cdot \frac{\mathbf{m}^t}{\sqrt{\mathbf{v}^t} + \epsilon}, \tag{2}$$

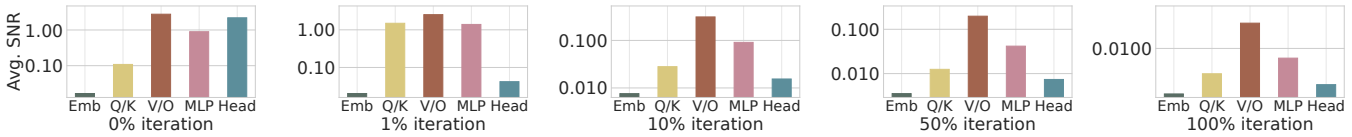
where $\eta_t > 0$ denotes the learning rate at time step t, and ϵ is a small constant for numerical stability. The success of the Adam optimizer lies in its ability to normalize gradient scales, allowing for consistent progress across different dimensions of the loss landscape. Ideally, the first-moment estimate $\mathbf{m}^t$ serves as a proxy for the mean of the gradients, $\mathbb{E}[\mathbf{g}]$, while the second-moment estimate $\mathbf{v}^t$ approximates the local curvature, corresponding to $\mathbb{E}[\mathbf{g}^{\odot 2}]$ [Hwang, 2024; Orvieto and Gower, 2025]. However, this interpretation assumes that the second moment predominantly captures the curvature of the loss surface. In the context of stochastic optimization, $\mathbb{E}[\mathbf{g}^{\odot 2}]$ can be decomposed into the squared signal and the gradient variance as follows

$$\mathbb{E}[\mathbf{g}^{\odot 2}] = \boldsymbol{\mu}^{\odot 2} + \boldsymbol{\sigma}^{\odot 2}, \tag{3}$$

where $\boldsymbol{\mu} = \mathbb{E}[\mathbf{g}]$ denotes the true gradient signal and $\boldsymbol{\sigma}^2 = \mathrm{Var}(\mathbf{g})$ represents the stochastic noise.



(a) Pre-training LLaMA-60M on C4.



(b) Pre-training LLaMA-130M on C4.

Figure 1: **SNR variation across modules during training.** The SNR variation trends across models of different sizes are similar, with a decrease throughout the training. Throughout the training, the SNR of the VO module remains the highest, and the relative SNR relationships between modules stabilize after the first 1% of iterations. Both the Emb and Head experience considerable noise in the early stages of training.

To quantify the impact of noise on optimization, we consider the model parameters as a collection of matrices or tensors $\{\mathbf{w}_m\}$, where each module consists of a set of parameters. For a given module $\mathbf{w}_m$, we define the Effective Signal Step ($\mathcal{D}_m$) as a measure of the module's optimization efficiency, which reflects the average optimization efficiency across all parameters in the module. Specifically, for any parameter i in module m, the effective signal step $\mathcal{D}_i$ at time step t can be defined as

$$\begin{aligned}\mathcal{D}_{m,i} &= \mathbb{E}\left[\frac{\mathbf{m}^t_{m,i}}{\sqrt{\mathbf{v}^t_{m,i}}+\epsilon}\right] \approx \frac{\boldsymbol{\mu}_{m,i}}{\sqrt{\boldsymbol{\mu}^2_{m,i}+\boldsymbol{\sigma}^2_{m,i}}} \\ &= \frac{1}{\sqrt{1+\boldsymbol{\sigma}^2_{m,i}/\boldsymbol{\mu}^2_{m,i}}} = \frac{1}{\sqrt{1+1/S_{m,i}}},\end{aligned} \tag{4}$$

where $S_{m,i} = \frac{\|\boldsymbol{\mu}_{m,i}\|^2}{\|\boldsymbol{\sigma}_{m,i}\|^2}$ represents the SNR for the i-th parameter, with $\boldsymbol{\mu}_{m,i}$ and $\boldsymbol{\sigma}_{m,i}$ representing the expected gradient and variance for parameter i, respectively. The behavior of $\mathcal{D}_{m,i}$ characterizes how Adam adaptively scales updates based on the local signal integrity. To build intuition, we examine two limiting cases:

- High SNR Regime ($S_{m,i} \to \infty$): When the signal dominates the noise, $\mathcal{D}_{m,i}$ approaches 1. In this case, the update term $\mathbf{M}/\sqrt{\mathbf{V}}$ simplifies to $\text{sign}(\boldsymbol{\mu})$, and Adam effectively aligns with Sign Gradient Descent (SignGD) [Bernstein *et al.*, 2018]. This observation is consistent with prior findings that sign-based methods can match [Kunstner *et al.*, 2023] or even outperform [Chen *et al.*, 2023] Adam in certain settings.
- Low SNR Regime ($S_{m,i} \to 0$): Conversely, when noise overwhelms the signal where $\boldsymbol{\sigma}_{m,i} \gg \boldsymbol{\mu}_{m,i}$, $\mathcal{D}_{m,i}$ collapses towards $\sqrt{S_{m,i}}$. Here, the update magnitude is heavily suppressed by the variance in the denominator, leading to an unintended noise-damping effect.

In practice, the SNR of different model components varies substantially over the course of training. In the early stages, update directions tend to be more stable, with stronger signal components and higher effective SNR. Later in training, as gradient magnitudes shrink and stochastic fluctuations become comparable to the signal, the SNR typically decreases. This shift is consistent with a transition from rapid progress to fine-grained convergence.

3.2 Module-wise Rescaling via SNR

To estimate the dynamic changes in SNR across different modules in LLM training, we group parameters by functionality (Embedding, Head, Query/Key, Value/Output, MLP) and estimate the average SNR for each module as follows

$$S_m = \frac{1}{|\mathbf{w}_m|}\sum_{i=1}^{|\mathbf{w}_m|} S_{m,i},$$

where $m \in \{\text{Emb, Head, QK, VO, MLP}\}$ and $|\mathbf{w}_m|$ denotes the number of parameters in $\mathbf{w}_m$. This grouping covers 99% of the trainable parameters in LLMs, and because parameters within the same module share similar properties, the resulting module-level average SNR remains statistically meaningful.

This empirical analysis uncovers a pronounced imbalance in SNR across different Transformer modules (Figure 1, and Appendix). At a global level, the SNR of all modules exhibits a gradual decline over the course of training, indicating that optimization is approaching convergence and that the loss landscape has transitioned into a relatively flat region. Despite this shared trend, substantial disparities emerge across modules, particularly during the early stages of training. The attention components, such as the Q/K and V/O, as well as the MLP blocks consistently operate in a high-SNR regime, enabling gradient updates that are largely sign-consistent and dominated by informative signals. By contrast, the embedding layers and the output head display significantly lower SNR values, remaining below 0.1 even at initialization. As a result, Adam's updates become noise-dominated and tend toward $\sqrt{S_m}$.

This disparity implies that under a global learning rate η_t, the Embedding and Head layers are being "left behind" because their effective signal contribution is severely attenuated compared to the VO baseline. To resolve this structural imbalance, we propose the **Relative Signal-wise Equilibrium** principle: the effective signal contribution of every module

should be rescaled to match the baseline established by the healthy reference module.

Considering that the damping factor in Adam follows an inverse square root relationship with SNR (i.e., $\mathcal{D}_m \propto \sqrt{S_m}$), we define the Relative Rescaling Factor (α_m) as

$$\alpha_m = \frac{1/\sqrt{S_m}}{1/\sqrt{S_{base}}} = \sqrt{\frac{S_{base}}{S_m}}. \tag{5}$$

In this framework, the reference module's scaling factor α_{base} is naturally anchored to 1.0, while noise-heavy modules with $S_m < S_{base}$ receive a square-root-proportional boost to their effective learning rate. We present the pseudo-code in the Appendix.

Comparison with Existing Works. Wang *et al.* 2025a studied sharpness heterogeneity across LLM modules and showed that module-wise learning rates can speed up training, but left tuning guidelines unspecified, requiring manual adjustment. Similarly, Li *et al.* 2025 examined intra-layer heterogeneity and proposed clustering-based scaling. Crucially, MoLS eliminates the need for clustering heuristics or hyperparameter tuning, offering a fully automated and principled solution for module-wise learning rate configuration.

4 Methodology

In this section, we present our practical methodology, building directly upon the module-wise SNR analysis in the previous section. Our goal is to correct the structural imbalance in Adam's effective updates across Transformer modules using a lightweight and scalable calibration, without introducing dynamic overhead or instability. We refer to our approach as **Module-wise LR Scaling via SNR (MoLS)**. We design the method with the following considerations:

- **Computation cost:** Dynamic SNR estimation during training is computationally expensive for LLMs.
- **Training stability:** Sudden changes in learning rate or update can destabilize LLM training, causing spikes in the loss. This requires additional transition intervals to smooth the change of the learning rate.
- **Empirical observation:** We observe empirically that after the first 1% of training steps, module-level SNR magnitudes tend to stabilize, which motivates a one-shot estimation during warm-up followed by smooth scaling of module learning rates.

One-shot Module-wise SNR Estimation. We perform a one-shot module-wise SNR estimation using a small set of additional samples during the warm-up phase for each module. The estimated SNRs are then used to compute the relative scaling factors for each module.

Relative Signal-wise Rescaling. Let m_{base} denote a reference module with high SNR (e.g., the module with the largest estimated SNR during warm-up). For module m, the scaling factor is defined in equation 5. This factor is applied multiplicatively to the global learning rate. Once determined, the factors remain fixed for the remainder of training.

Practical Integration. The method does not modify Adam's internal updates and incurs negligible overhead. By distributing the scaling smoothly over the remaining warm-up steps, we avoid abrupt learning rate changes, ensuring stable training for large-scale LLMs.

5 Experiments

In this section, we experimentally validate the effectiveness of our proposed method, focusing on both pre-training and fine-tuning tasks. Detailed descriptions of experimental settings can be found in the Appendix.

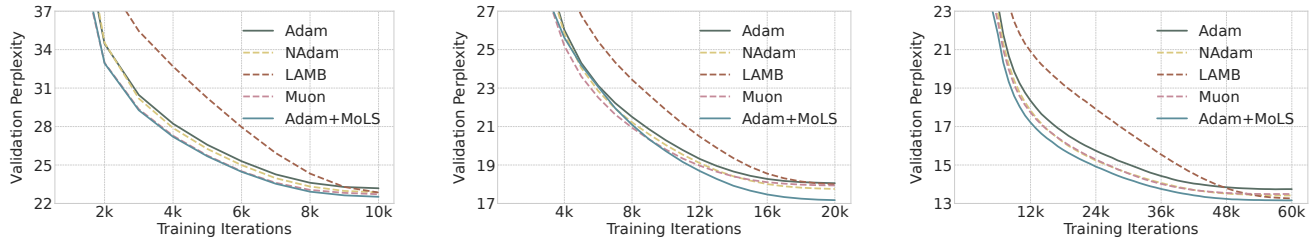
5.1 Pre-Training LLaMA Models

We show that MoLS can be incorporated into the Adam-style optimizer in a plug-and-play manner while incurring negligible computational overhead. Across different pre-training datasets and models up to 7B parameters, MoLS achieves faster convergence and lower perplexity compared to the baselines.

Setup. All experiments use BF16 precision to save GPU memory, and we follow the same LLaMA model settings as those reported by Zhao *et al.* 2024. The default sequence length is set to 256, with a total batch size of 512, and a gradient clipping of 1.0. We apply linear warmup over the first 10% of training steps and cosine decay for the remainder. We evaluate the effectiveness of MoLS under two settings: full-rank pre-training on C4 [Raffel *et al.*, 2020] and OpenWebText [Gokaslan *et al.*, 2019] dataset, where we apply MoLS on Adam. The other setting is memory-efficient pre-training on C4, where MoLS is applied on Adam-mini.

Baselines. We consider a range of **open-source** LLM pre-training optimizers as baselines, including **Adam** [Kingma, 2014; Loshchilov and Hutter, 2017], which is the standard optimizer used in LLM training. **NAdam** [Dozat, 2016], the Nesterov-accelerated version of Adam. **LAMB** [You *et al.*, 2020], an Adam variant with layer-wise adaptive scaling. **Muon** [Jordan *et al.*, 2024; Liu *et al.*, 2025], an SGD-with-momentum variant incorporating Newton–Schulz iterations. Furthermore, we evaluate a set of memory-efficient optimizers, including **Adam-mini** [Zhang *et al.*, 2025], an Adam variant with block-wise second-moment storage, **GaLore** [Zhao *et al.*, 2024], and **APOLLO** [Zhu *et al.*, 2024]. For a fair comparison, all methods adopt an identical learning rate search strategy, exploring the range $\{1\text{e}{-4}, 2.5\text{e}{-4}, 5\text{e}{-4}, 1\text{e}{-3}, 2.5\text{e}{-3}, 5\text{e}{-3}, 1\text{e}{-2}, 2.5\text{e}{-2}\}$. We provide the full set of hyperparameters in the appendix.

Main Results. We report the final validation perplexity (PPL) of LLaMA models ranging from 60M to 1.3B parameters on the C4 dataset in Table 1 and α_m in the Appendix. We further compare the wall-clock training time of Adam and Adam + MoLS, from which several observations emerge. First, MoLS consistently achieves lower final validation PPL than all evaluated baselines, yielding up to a 1.06 PPL reduction over Adam on the 1.3B model. Second, MoLS introduces negligible computational overhead. As shown in Table 2, the additional cost incurred by SNR estimation accounts for less



(a) LLaMA-60M (b) LLaMA-130M (c) LLaMA-350M

Figure 2: **PPL curves for pre-training LLaMA-60M to 350M on OpenWebText dataset.** MoLS outperforms the finely-tuned baselines while maintaining a higher convergence rate.

Methods	60M	130M	350M	1.3B
Pre-training with Full-Rank Optimizers				
Adam	$29.55_{\pm.02}$	$22.82_{\pm.00}$	$17.25_{\pm.00}$	$14.51_{\pm.00}$
NAdam	$28.95_{\pm.04}$	$22.42_{\pm.02}$	$16.76_{\pm.00}$	$15.06_{\pm.00}$
LAMB	$28.39_{\pm.11}$	$22.30_{\pm.05}$	$16.69_{\pm.05}$	$13.74_{\pm.00}$
Muon	$28.99_{\pm.03}$	$22.73_{\pm.01}$	$17.23_{\pm.01}$	$14.29_{\pm.00}$
Adam+MoLS	**28.48**$_{\pm.10}$	**21.94**$_{\pm.06}$	**16.41**$_{\pm.03}$	**13.45**$_{\pm.00}$
Δ Reduction	**-1.07**	**-0.88**	**-0.84**	**-1.06**
Pre-training with Memory-efficient Optimizers				
Adam-mini	$29.63_{\pm0.5}$	$23.73_{\pm.01}$	$17.83_{\pm.01}$	$15.10_{\pm.00}$
GaLore	$33.24_{\pm.03}$	$25.22_{\pm.02}$	$18.67_{\pm.01}$	$14.90_{\pm.00}$
APOLLO	$29.92_{\pm.15}$	$22.86_{\pm.20}$	$16.66_{\pm.07}$	$14.20_{\pm.00}$
Adam-mini+MoLS	**28.49**$_{\pm.07}$	**21.91**$_{\pm.07}$	**16.86**$_{\pm.03}$	**13.57**$_{\pm.00}$
Δ Reduction	**-1.14**	**-1.82**	**-0.97**	**-1.53**
Training Tokens	1.1B	2.2B	6.4B	13.1B

Table 1: **Final validation PPL (lower is better, 3 independent runs for 60M to 350M models) for pre-training LLaMA models on the C4 dataset. Bold** and **green** types denote the best results and PPL reduction↓ of MoLS (blue background).

than 2% of the total training time, making the overhead effectively negligible. Moreover, since SNR estimation is performed on the CPU, MoLS introduces no additional GPU memory overhead. Third, MoLS delivers approximately 1.2× to 1.4× speedup over Adam with tuned hyperparameters and exhibits faster early-stage convergence compared to LAMB. Finally, MoLS is fully compatible with memory-efficient optimizers such as Adam-mini, reducing the final PPL by 1.53 on the 1.3B model. Together, these results demonstrate the effectiveness, efficiency, and broad applicability of MoLS across model scales and optimization settings.

Methods	PPL	Training Time (h)
Adam	14.51	54.18
Adam + MoLS	13.45(**+7.30%**)	55.08(**+1.66%**)
Adam-mini	15.10	53.10
Adam-mini + MoLS	13.57(**+10.1%**)	54.01(**+1.71%**)

Table 2: **Gains vs Costs.** Relative **gains**↑ in final PPL and **costs**↓ in total training time on pre-training LLaMA-1.3B.

Scaling up to Pre-Training LLaMA-3B and 7B. We validate MoLS on 8-bit Adam [Dettmers *et al.*, 2021] for pre-training LLaMA-3B and 7B models with gradient checkpointing to save memory. Owing to limited computational resources, we do not include Nadam, LAMB, and Adam-mini in this experiment, and present the validation PPL across training steps in Table 3. Our MoLS reduces the final PPL by 1.12 on LLaMA-3B and 1.69 on 7B. This effectively validates the effectiveness of MoLS on large-scale LLMs.

Methods	30K	60K	90K	120K	150K
Pre-training LLaMA-3B					
8-bit Adam	18.63	15.69	14.36	14.19	-
Muon	18.14	15.40	14.13	14.02	-
GaLore	18.44	15.98	14.90	14.73	-
APOLLO	19.49	15.40	14.09	13.75	-
8-bit Adam + MoLS	17.81	14.51	13.26	**13.07**	-
Pre-training LLaMA-7B					
8-bit Adam	18.73	16.13	14.56	13.40	13.23
Muon	18.16	15.71	13.87	13.23	13.19
GaLore	18.35	15.63	14.33	13.77	13.69
APOLLO	18.49	15.17	13.54	12.80	12.63
8-bit Adam + MoLS	16.22	13.77	12.34	11.70	**11.54**

Table 3: Pre-training LLaMA-3B and LLaMA-7B models on the C4 dataset with gradient checkpointing.

5.2 Efficient Fine-Tuning

In this section, we validate our method on LLM downstream tasks. We show that MoLS remains effective on these downstream tasks and is compatible with LoRA [Hu *et al.*, 2022].

Setup. We employ four open-source pre-trained models in the fine-tuning experiments, including LLaMA-3.2-1B/3B [Grattafiori *et al.*, 2024], Gemma3-1B [Team *et al.*, 2025], and Qwen2.5-7B [Qwen *et al.*, 2024]. The downstream tasks are divided into two categories: SFT on 8 common-sense reasoning tasks, including Winogrande (WG) [Sakaguchi *et al.*, 2021], PIQA [Bisk *et al.*, 2020], SIQA [Sap *et al.*, 2019], Open-BookQA (OBQA) [Mihaylov *et al.*, 2018], HellaSwag (HS) [Zellers *et al.*, 2019], BoolQ [Clark *et al.*, 2019], and ARC (ARC-Easy and ARC-Challenge) [Clark *et al.*, 2018]. The experimental setup follows Zhu *et al.* 2024; PEFT on MMLU [Hendrycks *et al.*, 2020] tasks, the implementation follows Zheng *et al.* 2024.

Baselines. For the common-sense reasoning tasks, we compare against full-parameter baselines used in Section 5.1. For

Method	WG	PIQA	SIQA	OBQA	HS	BoolQ	Arc-E	Arc-C	Avg.
Adam	68.19	76.12	72.36	69.00	69.19	64.34	72.22	55.12	68.07
NAdam	67.91	76.20	70.77	68.21	69.03	63.77	73.31	55.20	68.05
Lamb	68.33	74.81	71.56	69.07	68.21	64.00	71.48	54.92	67.79
Muon	70.21	76.34	71.90	68.89	69.31	64.51	72.15	55.14	68.55
Adam + MoLS	70.47	77.61	72.41	70.33	69.18	64.40	73.73	55.36	**69.18**

Table 4: Evaluation results on Llama-3.2-1B for common-sense reasoning tasks.

PEFT experiments, we include Full-Adam, LoRA [Hu *et al.*, 2022], GaLore, and APOLLO as representative baselines.

Main Results. The SFT results are reported in Table 4, while the MMLU accuracies under PEFT are summarized in Table 5. Across common-sense reasoning benchmarks, MoLS consistently matches or outperforms all baselines, achieving up to a 1.11 average accuracy improvement over Adam. Furthermore, MoLS integrates seamlessly with LoRA, yielding additional accuracy gains on MMLU and underscoring the versatility and robustness of the proposed approach.

Models	Methods	STEM	Soc.	Hum.	Other	Avg.
Gemma3-1B	Full	27.63	26.84	25.12	25.14	26.04
	LoRA	26.61	25.93	24.78	25.02	25.48
	GaLore	27.40	26.71	24.80	25.69	25.99
	APOLLO	27.47	27.14	25.61	25.76	**26.38**
	LoRA + MoLS	27.51	26.96	25.67	25.49	26.31
LLaMA3.2-3B	Full	46.55	65.81	49.16	63.17	55.48
	LoRA	47.22	63.41	49.56	61.57	54.86
	GaLore	46.62	65.58	48.37	63.33	55.22
	APOLLO	46.16	65.91	49.03	62.80	55.29
	LoRA + MoLS	47.34	65.77	49.61	62.41	**55.62**
Qwen2.5-7B	Full	OOM				-
	LoRA	67.69	82.91	66.29	75.72	72.41
	GaLore	68.32	82.87	66.08	76.10	72.55
	APOLLO	68.49	82.97	65.80	76.65	72.65
	LoRA + MoLS	68.71	82.87	66.41	77.21	**73.01**

Table 5: **PEFT results on MMLU tasks** (1 A100 40GB GPU). We report the best accuracy obtained by sweeping the learning rate with the range $\{1e{-}5,\ 2.5e{-}5,\ 5e{-}5,\ 1e{-}4,\ 1.5e{-}4,\ 2e{-}4\}$.

6 Additional Investigations

In this section, we present a set of supplementary empirical studies aimed at further evaluating the robustness and generalization of MoLS. Specifically, our analysis covers the following aspects: (i) pre-training on alternative model architectures, including GPT-2 [Radford *et al.*, 2019] and Qwen2.5 [Qwen *et al.*, 2024]; (ii) pre-training with extended sequence lengths; (iii) pre-training with larger training token budgets; (iv) comparisons against manually tuned module-level learning rates; and (v) ablation studies examining the role of SNR across different model modules. Moreover, we provide a training loss analysis and ablation studies on SNR estimation samples in the Appendix.

Pre-Training on GPT and Qwen Architectures. To further assess the generality of our approach, we extend our experiments beyond the LLaMA family and pretrain GPT-2 [Radford *et al.*, 2019] and Qwen2.5 [Qwen *et al.*, 2024] models on the C4 dataset. The quantitative results are summarized in Table 6. Across both architectures, MoLS consistently achieves lower final validation PPL. These results demonstrate that MoLS is not specific to a particular model family, but is broadly effective across diverse architectures.

Methods	GPT2-125M	GPT2-355M	Qwen2.5-350M
Adam	31.09	22.19	16.87
NAdam	30.26	**21.55**	16.48
LAMB	30.50	21.70	16.72
Muon	30.26	22.39	16.78
Adam+MoLS	**29.65**	21.66	**16.34**
Training Tokens	1.1B	2.2B	6.4B

Table 6: Final validation PPL for pre-training GPT/Qwen on C4.

Pre-Training in the Long-Context Regime. Long-context modeling is essential for effective reasoning in large language models, yet training with extended sequence lengths is more susceptible to optimization instabilities such as gradient explosion or vanishing. To examine the robustness of our method in this regime, we evaluate its performance under long-context pre-training. Specifically, we pretrain a LLaMA-130M model using a sequence length of 1024, while keeping other experimental settings identical to Section 5.1. The resulting validation PPL learning curves are presented in Figure 4. As shown, MoLS consistently outperforms the baselines even in the long-context setting, highlighting its robustness and effectiveness for large-scale LLM pre-training.

Pre-Training with Extended Token Budgets. Because MoLS applies module-wise learning-rate scaling, certain components are trained with larger effective learning rates, which could render comparisons at a fixed number of training iterations potentially unfair. To mitigate this concern, we evaluate all baseline methods under extended training schedules, ensuring that optimizers with smaller learning rates are afforded a sufficient number of optimization steps. Concretely, we pretrain LLaMA-130M for 100k iterations, corresponding to approximately $5\times$ the number of training tokens recommended by the Chinchilla scaling law [Hoffmann *et al.*, 2022]. The resulting performance curves are reported in Figure 5. Even under prolonged training with substantially increased token budgets, MoLS consistently maintains a clear advantage. These results indicate that the observed gains are not an artifact of faster early optimization, but persist over long training horizons, effectively alleviating concerns about insufficient optimization steps for the baselines.

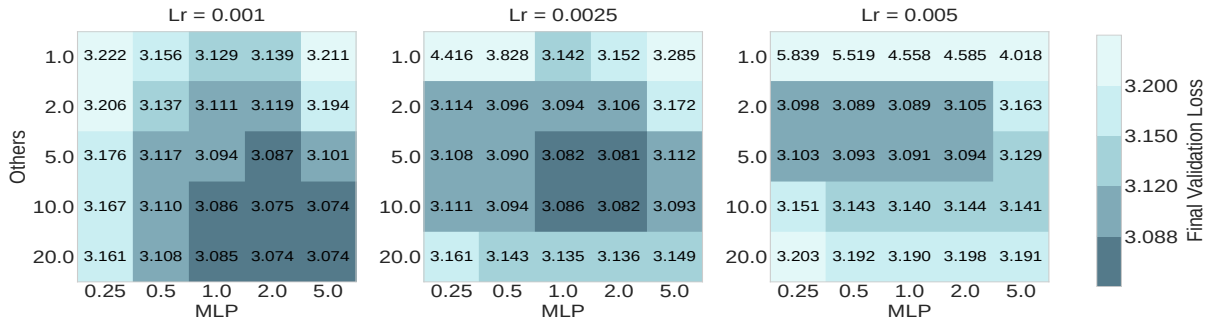


Figure 3: **Final validation loss for pre-training LLaMA-130M on C4 by hand-tuned module-wise learning rate of Adam.** We tune lr_{base}, α_{MLP}, and α_{others}. Our MoLS achieves a final validation loss of 3.088, which is close to the optimal value (3.074).

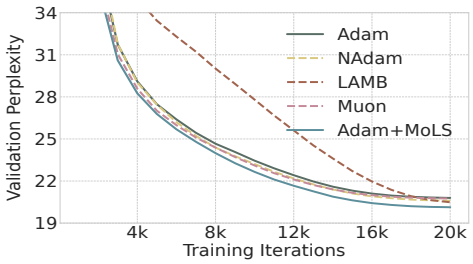


Figure 4: Pre-training LLaMA-130M with context length of 1024.

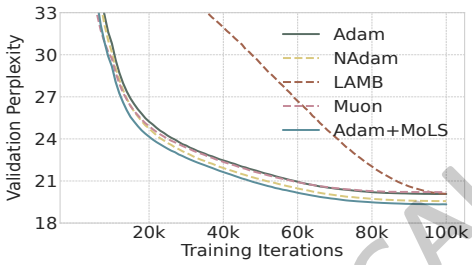


Figure 5: Pre-training LLaMA-130M on 5× Chinchilla tokens.

Comparison with Hand-Tuned Module-Wise Learning Rates. Our approach introduces an adaptive learning rate at the module level. To assess whether the resulting scaling is competitive with carefully selected alternatives, we compare against manually tuned module-wise learning rates as a baseline. Owing to the prohibitive computational cost of exhaustive hyperparameter search, we restrict the manual tuning to a small set of parameter groups guided by prior empirical findings. Specifically, we set a base learning rate for the QK and VO modules as $lr_{\text{base}} \in \{1\text{e}{-3}, 2.5\text{e}{-3}, 5\text{e}{-3}\}$. The learning rate for the MLP modules is defined as a multiple of the base rate, $lr_{\text{mlp}} = \alpha_{\text{mlp}} \cdot lr_{\text{base}}$, $\alpha_{\text{mlp}} \in \{0.25, 0.5, 1.0, 2.0, 5.0\}$, while the remaining modules (e.g., embeddings and output head) use $lr_{\text{others}} = \alpha_{\text{others}} \cdot lr_{\text{base}}$, $\alpha_{\text{others}} \in \{1.0, 2.0, 5.0, 10.0, 20.0\}$. The results are summarized in Figure 3. Overall, our method outperforms the majority of manually tuned configurations and achieves performance close to the best hand-selected setting.

Ablation Study on Modules. We conduct ablation experiments on module-scaling configurations, using different grouping strategies to pretrain LLaMA-130M for SNR estimation; the results are shown in Table 7. We can see that finer-grained grouping leads to better improvements.

Grouping strategies	Loss	Reduction
Vanilla	3.127	-
Attn, MLP	3.139	**+0.012**
Attn, MLP, Emb	3.103	**- 0.024**
QK, VO, MLP, Emb	3.092	**- 0.035**
QK, VO, MLP, Emb, Head	**3.088**	**- 0.039**

Table 7: Ablation study on grouping strategies for pre-training LLaMA-130M on C4.

7 Conclusion

In this paper, we propose **Module-wise Learning Rate Scaling via SNR (MoLS)**, a plug-and-play, automated module-wise learning rate calibration method for large language models that addresses the excessive suppression of effective update magnitudes in low SNR modules under Adam-based optimizers. MoLS estimates the SNR of each functional module during a brief warm-up phase and applies module-level scaling to align the effective updates of noise-dominated modules with those of high-SNR reference modules, such as value/output projections. Importantly, MoLS introduces no additional hyperparameters and enables principled module-wise learning rate allocation without manual tuning. Extensive experiments on both pre-training and supervised fine-tuning across multiple model families demonstrate that the scaling factors estimated by MoLS closely match manually optimized module-wise learning rates, leading to improved convergence speed and generalization. Furthermore, MoLS is compatible with memory-efficient optimizers such as Adam-mini, making it suitable for large-scale and resource-constrained training settings. Overall, MoLS provides a simple yet effective mechanism for correcting structural optimization imbalances in Transformer-based models and highlights the importance of accounting for gradient noise when designing optimization strategies for LLMs.

Acknowledgements

Tao Sun is supported in part by the National Natural Science Foundation of China (Grant Nos. 62522610, 62376278), and NUDT Foundational Research Funding (JS25-02).

References

[Bernstein *et al.*, 2018] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International conference on machine learning*, pages 560–569. PMLR, 2018.

[Bisk *et al.*, 2020] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439, 2020.

[Chen *et al.*, 2023] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

[Clark *et al.*, 2018] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[Clark *et al.*, 2019] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[Dettmers *et al.*, 2021] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.

[Dozat, 2016] Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

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

[Grattafiori *et al.*, 2024] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[Hendrycks *et al.*, 2020] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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

[Hu *et al.*, 2022] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[Hwang, 2024] Dongseong Hwang. Fadam: Adam is a natural gradient optimizer using diagonal empirical fisher information. *arXiv preprint arXiv:2405.12807*, 2024.

[Jordan *et al.*, 2024] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024.

[Kingma, 2014] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kunstner *et al.*, 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.

[Kunstner *et al.*, 2024] Frederik Kunstner, Alan Milligan, Robin Yadav, Mark Schmidt, and Alberto Bietti. Heavy-tailed class imbalance and why adam outperforms gradient descent on language models. *Advances in Neural Information Processing Systems*, 37:30106–30148, 2024.

[Li *et al.*, 2025] Siyuan Li, Juanxi Tian, Zedong Wang, Xin Jin, Zicheng Liu, Wentao Zhang, and Dan Xu. Taming llms by scaling learning rates with gradient grouping. *arXiv preprint arXiv:2506.01049*, 2025.

[Liu *et al.*, 2020] Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. *arXiv preprint arXiv:2004.08249*, 2020.

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

[Loshchilov and Hutter, 2017] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[Mihaylov *et al.*, 2018] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

[Ormaniec *et al.*, 2024] Weronika Ormaniec, Felix Dangel, and Sidak Pal Singh. What does it mean to be a transformer? insights from a theoretical hessian analysis. *arXiv preprint arXiv:2410.10986*, 2024.

[Orvieto and Gower, 2025] Antonio Orvieto and Robert M. Gower. In search of adam's secret sauce. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[Qwen *et al.*, 2024] Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, and Fei Huang. Qwen2.5 technical report. 2024.

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

[Raffel *et al.*, 2020] 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.

[Sakaguchi *et al.*, 2021] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

[Sap *et al.*, 2019] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

[Team *et al.*, 2025] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[Touvron *et al.*, 2023] 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*, 2023.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[Wang *et al.*, 2025a] Jinbo Wang, Mingze Wang, Zhanpeng Zhou, Junchi Yan, Lei Wu, et al. The sharpness disparity principle in transformers for accelerating language model pre-training. *arXiv preprint arXiv:2502.19002*, 2025.

[Wang *et al.*, 2025b] Mingze Wang, Jinbo Wang, Jiaqi Zhang, Wei Wang, Peng Pei, Xunliang Cai, Lei Wu, et al. Gradpower: Powering gradients for faster language model pre-training. *arXiv preprint arXiv:2505.24275*, 2025.

[Xu *et al.*, 2024] 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*, 2024.

[You *et al.*, 2020] 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. In *International Conference on Learning Representations*, 2020.

[Zellers *et al.*, 2019] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

[Zeng *et al.*, 2022] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

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

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

[Zhao *et al.*, 2024] Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In *Forty-first International Conference on Machine Learning*, 2024.

[Zheng *et al.*, 2024] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. Llamafactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, 2024.

[Zhu *et al.*, 2024] 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, 2024.