On quantizing the state of the Muon Optimizer

Anonymous authors
Paper under double-blind review

Abstract

The Muon optimizer, based on matrix orthogonalization, has recently shown faster convergence and better computational efficiency over AdamW in LLM pre-training. However, the memory overhead of maintaining high-precision optimizer states remains a challenge for large-scale deployment. In this paper, we introduce the 8-bit Muon optimizer using blockwise quantization.

In extensive Chinchilla-optimal experiments on pre-training models of up to 2.7B in size and fine-tuning them for instruction following, we demonstrate that 8-bit Muon achieves parity with Muon in terms of validation loss and downstream benchmarks, while achieving up to a 62% reduction in optimizer state footprint. Crucially, we show that Muon's update mechanism is uniquely compatible with a simple linear quantization scheme, bypassing the complex dynamic scaling required for quantized AdamW. We supplement our empirical findings with a theoretical analysis of Muon's robustness to quantization noise.

1 Introduction

Scaling laws for large language models (LLMs) (Kaplan et al., 2020; Hoffmann et al., 2022) indicate that larger models generally achieve better out-of-distribution performance across diverse tasks. Yet, GPU high-bandwidth memory (HBM) capacity has not kept pace with parameter counts. During training, memory is dominated by model parameters, gradients, optimizer states, and activations. Systems work has therefore focused on distributing these tensors across devices via distributed data parallel (DDP), Fully Sharded Data Parallel (FSDP) (Zhao et al., 2023), ZeRO stage-3 in DeepSpeed (Rajbhandari et al., 2020), and tensor/model parallelism (Shoeybi et al., 2019) in order to improve training performance.

Orthogonal to sharding is compressing the optimizer state. AdamW (Loshchilov & Hutter, 2017; Kingma, 2014), the *de facto* optimizer for LLMs, maintains two FP32 moment buffers (first and second moments) per parameter. For an 8B-parameter model (e.g., an 8B Llama-3 variant (Dubey et al., 2024)), this alone occupies 64 GB ($\sim$80% of an NVIDIA H100's 80 GB HBM), leaving little headroom for parameters, gradients, and activations. To mitigate this, Dettmers et al. (2021) quantize Adam's optimizer states to 8 bits via block-wise *dynamic* (non-linear) quantization, preserving stability in the presence of extreme values while reducing optimizer memory by roughly $4\times$, enabling performant training under tight memory budgets.

Recently, there has been a surge of interest in moving beyond AdamW to improve training efficiency (Anil et al., 2020; Shazeer & Stern, 2018; Vyas et al., 2024). Among various advances, one particularly promising optimizer is **Muon** (Jordan et al., 2024)[1], which orthogonalizes the gradient momentum before updating the parameters. Equalizing the importance of all update directions results in improved stability and better convergence (Bernstein & Newhouse, 2024a; Bernstein, 2025). Several large-scale studies have confirmed Muon's ability to achieve improved convergence rate to a target validation loss compared to AdamW on a compute-optimal setup (Liu et al., 2025; Shah et al., 2025). Muon has also been used

[1]Muon is closely related to SGD with momentum, adding a per-layer matrix orthogonalization

to train extremely large models up to a trillion parameters in size, like Kimi K2 (Team et al., 2025) and GLM4.5 (Zeng et al., 2025). While Muon offers improved convergence over AdamW, it still maintains FP32 optimizer state that competes for scarce HBM alongside parameters, gradients, and activations. As models continue to scale, this memory pressure remains a practical constraint even when the underlying optimizer improves. See Appendix A for extended discussion of related work.

1.1 Contributions

In this paper, we introduce the **8-bit Muon** optimizer with blockwise quantization. Whereas 8-bit AdamW variants are usually stable only with dynamic quantization, we demonstrate that **8-bit Muon can handle both linear and dynamic quantization effectively**.

We systematically study approaches to quantizing the Muon optimizer by pretraining GPT models of up to 2.7B parameters with Chinchilla-optimal data ratios on the FineWeb-Edu set. Notably, we show that dynamic and linear quantization of Muon optimizer updates for pre-training workloads demonstrate parity in terms of quality - both for validation loss and downstream benchmarks - while achieving a *62%* reduction in memory usage at the 2.7B model size. This memory reduction can be improved to 75% by additionally applying dynamic quantization to non-hidden-layer parameters updated by AdamW, at the cost of up to a 0.5% increase in validation loss. We observe similar results under Supervised Fine-Tuning (SFT), where memory efficiency is especially important in memory-constrained deployment settings.

A key empirical finding is that *linear* quantization of Muon optimizer state achieves parity with dynamic quantization and full-precision Muon in training progress. This contrasts with AdamW, where even sophisticated dynamic quantization does not fully retain performance, and linear quantization results in model divergence. This motivates our mechanistic analysis below.

Prior work by Dettmers et al. (2021) observed that naive 8-bit linear quantization can make AdamW unstable, and motivated block-wise dynamic quantization by highlighting sensitivity to quantization error in the optimizer states, especially the second-moment buffer. In Theorem 1, we formalize this mechanism by showing that AdamW can exhibit unbounded error under linear quantization, and we identify the second-moment accumulator in the denominator as the critical driver. This points to a natural control case: SGD with momentum, which removes the second-moment term. In Theorem 2, we show that linearly-quantized SGD+M admits bounded error at each optimizer step, and empirically verify that it remains effective in practice. Finally, in Theorem 3 we extend the same perspective to Muon, proving an analogous error bound with an additional dependence on the spectrum of the momentum matrix.

Key takeaway: Linear 8-bit quantization provides a simple and practical default to compress Muon's optimizer state, achieving quality parity while substantially reducing optimizer state memory.

2 Background

2.1 Quantization for optimizers

Quantization is the process of reducing the precision of numerical representations by mapping a value in a larger set to one in a smaller discrete set. For example, representing a real number as a 32-bit floating-point value, or converting a 32-bit float into an 8-bit integer, are both forms of quantization. In deep learning, model parameters and optimizer states are typically stored as 32-bit floating-point numbers, making their quantization to lower-precision formats a primary goal for memory efficiency.

For instance, linear quantization is a method (among many) to quantize the state tensor $\mathbf{x}$ of an optimizer from 32-bit floats to 8-bit integers. Specifically, the 8-bit integer linear quantization of $\mathbf{z}$ may be computed as: $\mathbf{z} = \text{round}\left(\frac{127}{\max_i |\mathbf{x}_i|} \cdot \mathbf{x}\right)$, where round denotes rounding to the nearest integer entry-wise. Note that all entries of $\mathbf{x}$ are integers in the range -127 to 127 and hence may be exactly represented by an 8-bit signed integer. This state $\mathbf{z}$ may then be de-quantized by $\tilde{\mathbf{x}} = (\max_i |\mathbf{x}_i|/127) \cdot \mathbf{z}$, and maintains the property that $|\tilde{\mathbf{x}}_j - \mathbf{x}_j| \leq \|\mathbf{x}\|_\infty/254$ for all j, guaranteeing bounded reconstruction error.

For 8-bit quantization, the number of addressable states is 256. Dettmers et al. (2021) discuss another quantization codebook design - dynamic quantization. Dynamic quantization is a more sophisticated approach that quantizes non-uniformly by allocating more codes to regions with high densities and fewer codes to sparsely used regions. This increases resilience to non-uniform data.

A crucial recipe for reducing the effect of outliers is blockwise quantization (Dettmers et al., 2021), where one can segment the optimizer state into one or more blocks, and then perform quantization (linear, dynamic, or other schemes) separately within each block. A nice side effect of blockwise quantization is that each block can be processed in parallel. For a more detailed treatment of blockwise and dynamic quantization, please refer to (Dettmers et al., 2021).

2.2 The Muon algorithm

The Muon update on a single hidden layer operates by accumulating the exponential average of the gradient and then determining the update direction by computing the polar factor of the momentum matrix approximately via Newton-Schulz Higham (2008). The vanilla version of Muon described above does not use weight decay. Additionally, it is not obvious whether it requires any hyperparameter tuning over AdamW baselines. Liu et al. (2025) introduced a variant of Muon that uses weight decay and also scales its update to match the update RMS of AdamW, producing the following version:

$$\begin{aligned} &\mathbf{M}^{(t)} := \beta\,\mathbf{M}^{(t-1)} + \nabla f\big(\mathbf{W}^{(t-1)}\big), \qquad \mathbf{O}^{(t)} := \text{NS}\big(\mathbf{M}^{(t)}\big), \\ &\mathbf{W}^{(t)} := (1-\alpha\lambda)\mathbf{W}^{(t-1)} - 0.2\alpha\sqrt{\max(m,n)}\mathbf{O}^{(t)}. \end{aligned} \tag{1}$$

where t is the iteration number and $\mathbf{M} \in \mathbb{R}^{m\times n}$ is the momentum of the gradient. NS stands for the Newton-Schulz iteration process (Bernstein & Newhouse, 2024b), used to find an approximation for the polar factor $\mathbf{U}\mathbf{V}^T$ where $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is the singular value decomposition (SVD) of $\mathbf{M}$. Orthogonalization equalizes the importance of each update direction by collapsing all singular values to 1. Liu et al. (2025) claim that with eqn. (1), hyperparameters such as learning rate and weight decay can be shared across matrix and non-matrix parameters. In the rest of the paper, any mention of Muon refers to the version in eqn. (1), unless stated otherwise. It is important to note that any non-matrix parameters and input/output parameters are optimized using AdamW, leaving Muon to focus on matrix-valued hidden parameters.

3 Experiments

This section presents a thorough empirical study of using linear and/or dynamic quantization to improve the memory efficiency of the Muon optimizer with minimal quality degradation. Recall that Muon is applied only to the two-dimensional hidden-layer weight matrices, while AdamW is used for all remaining parameters; see Table 4 in the Appendix for details.

3.1 The Quantized Muon algorithm

Throughout our experiments, we use the Muon variant of Liu et al. (2025), while compressing the optimizer state stored between iterations via linear or dynamic blockwise 8-bit quantization.

$$\begin{array}{llll}(1) & \widetilde{\mathbf{M}}^{(t-1)} := \text{Dequantize}_{\mathcal{S}}(\mathbf{Z}^{(t-1)}), & (2) & \mathbf{M}^{(t)} := \beta\,\widetilde{\mathbf{M}}^{(t-1)} + \nabla f(\mathbf{W}^{(t-1)}), \\ (3) & \mathbf{O}^{(t)} := \text{NS}(\mathbf{M}^{(t)}), & (4) & \mathbf{W}^{(t)} := (1-\alpha\lambda)\mathbf{W}^{(t-1)} - 0.2\alpha\sqrt{\max(m,n)}\,\mathbf{O}^{(t)}, \\ (5) & \mathbf{Z}^{(t)} := \text{Quantize}_{\mathcal{S}}(\mathbf{M}^{(t)}). & & \end{array} \tag{2}$$

where $\mathbf{Z}$ denotes the compressed momentum buffer and $\mathcal{S}$ denotes the associated auxiliary state used for dequantization. The remaining notation follows Equation 1.

3.2 Pre-training with 8-bit Muon

Architectures. For pre-training, we train a modified GPT-2 architecture (Radford et al., 2019) from scratch, replacing learned absolute positional embeddings with rotary positional embeddings (RoPE) (Su et al., 2024). To study scaling, we consider five model sizes: Small (162M), Medium (405M), Large (834M), XL (1.4B), and XXL (2.7B). For brevity, we refer to this architecture family as GPT throughout the paper. Additional architectural details are provided in Table 7 in Appendix C.1.

Datasets. We pre-train on FineWeb-Edu (Penedo et al., 2024). Following the compute-optimal scaling guidance of Hoffmann et al. (2022), we use approximately 20 tokens per parameter, yielding a training set size that increases with model size (the *Chinchilla-optimal* setting). For the 2.7B model, this corresponds to roughly 54B FineWeb tokens. We evaluate validation loss on 150K samples from the FineWeb validation split, totaling approximately 300M tokens. See Appendix C for pre-training and fine-tuning training details.

Evaluation criteria Our evaluation of pre-training is two-pronged (a) validation loss and (b) benchmark performance on six different tasks using the **lm-eval** harness (Gao et al., 2024). These tasks include MMLU (Hendrycks et al., 2020), LAMBADA (Paperno et al., 2016), BoolQ (Clark et al., 2019), HellaSwag (Zellers et al., 2019), ARC-Challenge and ARC-Easy (Clark et al., 2018).

3.2.1 Results

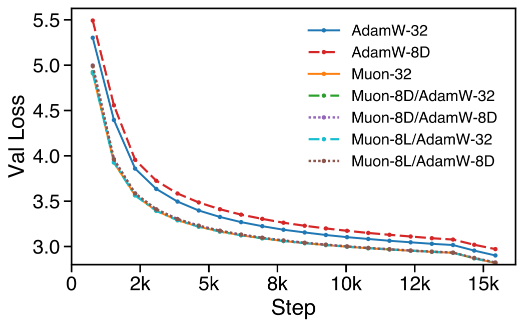

Figure 1: Validation loss for GPT-2 Medium across 7 variants.

Medium-scale ablation: stability and sensitivity to quantization. We first perform a controlled ablation on GPT-Medium, sweeping all stable 8-bit Muon variants from Table 4 and comparing them against AdamW baselines. Figure 1 plots validation loss versus training step. Under this setup, `AdamW-8L` diverges early (confirmed in separate runs and therefore omitted from the figure), consistent with the observations of Dettmers et al. (2021).

The AdamW baselines lag behind throughout training, with `AdamW-8D` consistently the worst among the plotted methods and `AdamW-32` also trailing the Muon family. Secondly, all Muon-based variants—including hybrids that quantize the AdamW-associated states and variants that use either dynamic or linear quantization for Muon-associated states—track `Muon-32` extremely closely, with near-overlapping trajectories across the full training horizon.

Scaling results: pretraining validation loss. Having established stability and quantization robustness at the Medium scale, we next scale to a broader sweep over model sizes. We compare the full-precision `Muon-32` baseline against two practically relevant 8-bit variants, `Muon-8L/AdamW-32` and `Muon-8L/AdamW-8D`. We focus on these `Muon-8L`-based implementations because they require minimal engineering overhead, making them straightforward to integrate into distributed training frameworks (e.g., Ahn et al. (2025)).

Table 1 reports the final pretraining validation loss for Small through XXL, aggregated over five seeds; the full training and validation curves are provided in Figures 3 and 4 in Appendix D.6. Across all sizes, `Muon-8L/AdamW-32` matches full-precision `Muon-32` up to run-to-run variability: the mean loss differs by at most 0.002 (e.g., Small: 3.120 vs. 3.122; XXL: 2.345 vs. 2.346). Meanwhile, Table 3 of Dettmers et al. (2021) implies a loss gap of 0.006 between 32-bit Adam and 8-bit Adam with block-wise dynamic quantization, both (i) with stable embeddings and (ii) without stable embeddings[2]. `Muon-8L/AdamW-8D` remains competitive but shows a small and consistent degradation relative to `Muon-32`, with the largest gap at XL (2.495 vs. 2.508). Overall, these results demonstrate that quantizing the Muon-associated state to 8-bit while keeping the AdamW component in full precision (`Muon-8L/AdamW-32`) preserves the optimization behavior of `Muon-32` across five model scales, while delivering the memory benefits of 8-bit state compression.

Downstream generalization We next assess generalization by evaluating each pretrained model on six established benchmarks. As shown in Table 5, hybrids that keep the AdamW-associated states in full precision (e.g., `Muon-8L/AdamW-32`) remain on par with the full-precision `Muon-32` baseline across model sizes, with differences in the six-task average that are within seed-level variation. In contrast, the fully 8-bit hybrid `Muon-8L/AdamW-8D` exhibits a small but consistent degradation in the averaged downstream score, most noticeably at Medium, XL and XXL, mirroring its slightly worse pretraining validation loss. These results suggest that aggressively quantizing the AdamW component can be the limiting factor for downstream transfer, whereas quantizing the Muon-associated state alone preserves generalization.

Takeaway. Across a targeted ablation (Figure 1) and a scale sweep (Tables 1–5), we find that (i) Muon-based 8-bit variants train reliably; and (ii) `Muon-8L` matches full-precision `Muon-32` in both pretraining validation loss and downstream generalization across model sizes. These findings support `Muon-8L/AdamW-32` as a strong default for memory-efficient pretraining without sacrificing quality.

Model size	Optimizer	Validation loss
Small	`Muon-32`	$3.120_{0.002}$
	`Muon-8L/AdamW-32`	$3.122_{0.002}$
	`Muon-8L/AdamW-8D`	$3.129_{0.001}$
Medium	`Muon-32`	$2.819_{0.001}$
	`Muon-8L/AdamW-32`	$2.820_{0.001}$
	`Muon-8L/AdamW-8D`	$2.824_{0.001}$
Large	`Muon-32`	$2.620_{0.001}$
	`Muon-8L/AdamW-32`	$2.621_{0.001}$
	`Muon-8L/AdamW-8D`	$2.623_{0.001}$
XL	`Muon-32`	$2.495_{0.001}$
	`Muon-8L/AdamW-32`	$2.496_{0.001}$
	`Muon-8L/AdamW-8D`	$2.508_{0.001}$
XXL	`Muon-32`	$2.345_{0.001}$
	`Muon-8L/AdamW-32`	$2.346_{0.001}$
	`Muon-8L/AdamW-8D`	$2.352_{0.001}$

Table 1: Pretraining validation loss across 5 seeds, reported as mean with standard deviation.

Comparison	WR (%)	LC WR (%)
`M-32` vs `A-32`	**59.67**	**59.17**
`M-8L/A-32` vs `A-32`	**60.48**	**59.93**
`M-8L/A-32` vs `M-32`	50.75	50.37
`M-8L/A-8D` vs `A-32`	58.47	57.91
`M-8L/A-8D` vs `M-32`	48.17	48.14

Table 2: Head-to-head AlpacaEval results on N=805 prompts. Win Rate is the raw preference rate; LC Win Rate is length-controlled. Values are percentages; 50% indicates a tie. Here, we abbreviate Muon and Adam as "M" and "A" respectively.

3.3 Fine-tuning with 8-bit Muon

Task Setup and Architecture We test the fine-tuning efficacy of the quantized Muon optimizer by using different variants to instruction-finetune the pre-trained models trained previously. Concretely, we choose two checkpoints of the GPT-XL model - one pre-trained

[2]We convert perplexity to loss via loss = ln(Perplexity)

with AdamW and the other pre-trained with Muon. It is empirically well-known that if a model is pre-trained with one optimizer, then empirically it is best to fine-tune it with the same optimizer Liu et al. (2025); Team et al. (2025). Thus, we then proceed to finetune the AdamW-pretrained checkpoint with `AdamW-32`, and the Muon-pretrained checkpoint with `Muon-32`, `Muon-8L/AdamW-32` , and `Muon-8L/AdamW-8D`.

Datasets We use the Alpaca instruction following dataset Taori et al. (2023). The dataset contains about 52k samples for instruction following, consisting of a diverse array of topics from code generation to creative writing.

Evaluation Criteria We use AlpacaEval 2.0 Dubois et al. (2024) (referred to hereinafter as AE2) as our main eval benchmark. AE2 is an automated evaluation framework that leverages a strong LLM-as-a-judge to compare the instruction following abilities of two different candidate LLMs on a variety of 805 prompts. AE2 reports both win rate and length-controlled (LC) win rate to mitigate verbosity bias. In our setup, we use GPT4 Turbo as the judge.

Once trained, we pit the four models against each other in a pairwise fashion to collect both win rate and LC win rate.

3.3.1 Results

Head-to-head AE2 results can be found in Table 2. The model fine-tuned with `Muon-32` consistently outperforms the model fine-tuned with `AdamW-32`, achieving 59.67% win rate (59.17% LC win rate). `Muon-8L/AdamW-32` performs similarly, achieving a large win over `AdamW-32` and being tied in performance with `Muon-32`. `Muon-8L/AdamW-8D` also outperforms `AdamW-32` and slightly underperforms when compared to `Muon-32`.

These findings mirror the results from pre-training, where we demonstrate that **quantizing the Muon-associated part of the optimizer results in essentially no loss in model quality**.

3.4 Memory footprint

Table 6 in the appendix compares the persistent High Bandwidth Memory footprint of the optimizer state for variants, when profiled for the GPT XL and XXL models. For the XXL model, `Muon-8D/AdamW-32` provides substantial relative savings of $\sim$**79%** and $\sim$**62%** when compared to `AdamW-32` and `Muon-32`. `Muon-8D/AdamW-8D` is more aggressive, stretching the advantage to $\sim$**86%** and $\sim$**75%**.

For the smaller XL model, the savings are slightly smaller. This is because the size of the embedding and lm-head matrices remains the same across model sizes because of a fixed vocabulary.

4 Understanding the stability of linear quantization

Our experimental results in Section 3.1 emphasize the effectiveness of applying linear quantization to the Muon optimizer state. This is in stark contrast to AdamW, where linear quantization results in divergence and even sophisticated dynamic quantization schemes may show performance degradation. In this section, we determine the mechanism driving these observations.

4.1 Why is 8-bit AdamW unstable under linear quantization?

Dettmers et al. (2021) observed that AdamW with naïve 8-bit *linear* quantization performs poorly because it allocates too little resolution to small-magnitude entries, yielding large relative errors precisely where optimizer states concentrate most of their mass. We make this behavior more precise in Theorem 1 by analyzing how quantization error propagates and is magnified through the second-moment accumulator $\mathbf{v}$. In particular, we show that if moderate-size gradient coordinates occur with non-negligible probability, then the expected squared error of one step of Adam with linearly quantized states *diverges* as the numerical stabilizer $\epsilon \to 0$. The result holds for standard Adam hyperparameters; constants are kept explicit to emphasize practical regimes (e.g., $\epsilon \approx 10^{-8}$) where the error is already proved to be orders of magnitude larger than the unquantized update norm.

Let Q denote the 8-bit linear quantization operator (Definition 1). Note that our error anlaysis is formalized over real numbers, and so Q composes the quantization and dequantization steps that perturb the optimizer state. We consider the base Adam algorithm without weight decay, though it extends to AdamW without loss of generality[3]. All algorithmic details, definitions, and proofs are provided in Appendix D. Note that we will use lower case letters when referring to one-dimensional vectorized parameters and upper case when operating on the two-dimensional hidden state parameters addressed by Muon, e.g., $\mathbf{w}$ versus $\mathbf{W}$.

Theorem 1 *Let $\mathbf{w}^{(1)}$ denote the parameters after one step of Adam as given in Algorithm 1, and let $\tilde{\mathbf{w}}^{(1)}$ denote the parameters after one step of the same algorithm with 8-bit linear quantization applied to the moment estimates (Definition 1), i.e.:*

$$\tilde{\mathbf{w}}^{(1)} = \mathbf{w}^{(0)} - \alpha \cdot \frac{Q(\mathbf{m}^{(1)})}{\sqrt{Q(\mathbf{v}^{(1)})} + \epsilon}.$$

Suppose that each entry of gradient $\mathbf{g}^{(1)} \in \mathbb{R}^d$ satisfies $\mathbb{P}\Big(\frac{\|\mathbf{g}\|_\infty}{60} < |\mathbf{g}_i| < \frac{\|\mathbf{g}\|_\infty}{16}\Big) \geq \nu$ and $\|\mathbf{g}\|_\infty \geq g_\infty > 256\,\epsilon$ with probability one. Then

$$\mathbb{E}\,\|\mathbf{w}^{(1)} - \tilde{\mathbf{w}}^{(1)}\|_2^2 \;\geq\; \frac{d\nu\,\alpha^2 g_\infty^2}{(256\,\epsilon)^2}.$$

We note that empirical validation of Theorem 1, as well as Theorems 2 and 3 below, can be found in Appendix D.5.

4.2 The curious case of 8-bit SGD with momentum

The proof of Theorem 1 shows that the instability of Adam with linear quantization arises primarily from error in the second-moment vector, which appears in the denominator of the update rule. This naturally raises the question: **does 8-bit linear quantization suffice when such a denominator is avoided?**

From a theoretical perspective, we show that, unlike Adam, SGD with momentum admits a uniform error bound under linear quantization. In particular, for any initialization of the weights and momentum, the quantization error remains bounded. Let $\eta > 0$ denote the step size and $\rho \in [0, 1)$ the momentum parameter.

Theorem 2 *Consider a step of SGD with momentum with and without 8-bit linear quantization of the momentum:*

$$\tilde{\mathbf{w}}^{(t+1)} = \mathbf{w}^{(t)} - \eta(\mathbf{g}^{(t)} + \rho Q(\mathbf{m}^{(t)})) \quad \text{and} \quad \mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta(\mathbf{g}^{(t)} + \rho \mathbf{m}^{(t)}).$$

From any point $\mathbf{w}^{(t)}$ and any momentum state $\mathbf{m}^{(t)}$, then with linear quantization, Q (Definition 1),

$$\|\tilde{\mathbf{w}}^{(t+1)} - \mathbf{w}^{(t+1)}\|_2^2 \leq d\eta^2\rho^2 \left(\frac{\|\mathbf{m}^{(t)}\|_\infty}{127}\right)^2.$$

Empirically, we confirm this difference. We train a ResNet-50 model (He et al., 2016) on the ImageNet dataset (Deng et al., 2009), using a standard training regime of 90 epochs. We compare AdamW and variants of SGD with momentum. Results are in Table 3. **We observe that SGD with linear 8-bit quantization achieves the same high validation top-1 accuracy (Goyal et al., 2017) of 76%+ as full-precision SGD**. AdamW underperforms when compared to SGD (a well-known result on image classification training), while AdamW with linear quantization diverges immediately. See Appendix C.3 for extended details.

[3]The result extends immediately to AdamW, since the decay of $\mathbf{w}^{(0)}$ affects both $\tilde{\mathbf{w}}^{(1)}$ and $\mathbf{w}^{(1)}$ equally and therefore cancels out.

Together, these results emphasize that quantized Adam's instability is driven specifically by quantizing the second-moment term in the denominator for moderately small coordinates.

Method	SGD+M	AdamW
FP32	76.21	74.42
8-bit linear quant.	76.25	—

Table 3: Top-1 validation accuracy (%) after 90 epochs for SGD+M and AdamW in FP32 and with 8-bit linear quantization. "—" indicates that AdamW with linear quantization diverged.

4.3 Linear quantization of the Muon optimizer

Like SGD with momentum, the Muon optimizer only maintains a first order momentum state between iterations, avoiding the second moment normalization term in AdamW. Theorem 3 below shows that the Muon optimizer admits an analogous uniform error bound, theoretically explaining the stability of quantized Muon.

In Theorem 3 we show the following: when Muon uses an exact orthogonalization procedure (via the SVD), the quantization error bound for a single layer matches the SGD case up to an additional dependence on the smallest singular value, s, of the momentum matrix. This dependence is natural, since the conditioning of the momentum controls the stability of the orthogonal factor. Our analysis considers the exact polar factor to avoid introducing additional finite-iteration Newton–Schulz approximation error into the bound.

Theorem 3 *Consider a step of Muon with momentum (using the exact polar factor rather than the Newton–Schulz approximation) with and without 8-bit linear quantization, Q (Definition 1), applied to the momentum. Let the layer weights and momentum state be $\mathbf{W}^{(t-1)}$ and $\mathbf{M}^{(t-1)}$, each with d entries (see Appendix D.4 for full update formulas).*

Suppose that, after a single gradient update, both the original and quantized momentum matrices are full column rank with minimum singular value at least $s > 0$. Then

$$\|\widetilde{\mathbf{W}}^{(t)} - \mathbf{W}^{(t)}\|_{\mathrm{F}}^2 \;\leq\; \frac{d\alpha^2\beta^2}{s^2}\left(\frac{\|\operatorname{vec}(\mathbf{M}^{(t-1)})\|_\infty}{127}\right)^2,$$

where $\widetilde{\mathbf{W}}^{(t)}$ denotes the weights after the quantized update and $\mathbf{W}^{(t)}$ after the unquantized update.

5 Conclusions

In this paper, we introduced 8-bit Muon, a memory-efficient variant of the Muon optimizer based on blockwise quantization of its optimizer state. We find that 8-bit Muon can match full-precision Muon in both validation loss and downstream benchmarks on Chinchilla-optimal GPT pretraining up to 2.7B parameters, while substantially reducing optimizer-state memory. In contrast to quantized AdamW, Muon remains compatible with simple linear quantization, providing a practical and easy-to-implement default. We complement these empirical results with a mechanistic and theoretical analysis that explains the differing behavior of AdamW, SGD with momentum, and Muon under quantization. Future work includes extending these methods to lower-bit quantization and combining them with complementary memory-saving techniques such as low rank momentum updates.

References

Kwangjun Ahn, Byron Xu, Natalie Abreu, and John Langford. Dion: Distributed orthonormalized updates. *arXiv preprint arXiv:2504.05295*, 2025.

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

Kayhan Behdin, Ayan Acharya, Aman Gupta, Qingquan Song, Siyu Zhu, Sathiya Keerthi, and Rahul Mazumder. Quantease: Optimization-based quantization for language models. *arXiv preprint arXiv:2309.01885*, 2023.

Jeremy Bernstein. Deriving muon. `https://jeremybernste.in/writing/deriving-muon`, 2025. Accessed: 2025-09-20.

Jeremy Bernstein and Laker Newhouse. Modular duality in deep learning. *arXiv preprint arXiv:2410.21265*, 2024a.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024b.

Rajendra Bhatia. *Matrix analysis*, volume 169. Springer Science & Business Media, 2013.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, et al. Lora learns less and forgets less. *arXiv preprint arXiv:2405.09673*, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL `https://arxiv.org/abs/1905.10044`.

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, 2018. URL `https://arxiv.org/abs/1803.05457`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

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

Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, M Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models, 2023. *URL https://arxiv.org/abs/2311.08105*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pp. arXiv–2407, 2024.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

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

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

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

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

Nicholas J Higham. *Functions of matrices: theory and computation*. SIAM, 2008.

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.

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.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Keller Jordan et al. Muon: An optimizer for hidden layers in neural networks. `https://kellerjordan.github.io/posts/muon/`, 2024. Accessed 2025-09-18.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020. URL `https://arxiv.org/abs/2001.08361`.

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

Ren-Cang Li. New perturbation bounds for the unitary polar factor. *SIAM Journal on Matrix Analysis and Applications*, 16(1):327–332, 1995.

Wen Li and Weiwei Sun. New perturbation bounds for unitary polar factors. *SIAM journal on matrix analysis and applications*, 25(2):362–372, 2003.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of machine learning and systems*, 6:87–100, 2024.

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

Zechun Liu, Barlas Oguz, Changsheng Zhao, Ernie Chang, Pierre Stock, Yashar Mehdad, Yangyang Shi, Raghuraman Krishnamoorthi, and Vikas Chandra. Llm-qat: Data-free quantization aware training for large language models. *arXiv preprint arXiv:2305.17888*, 2023.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. URL `http://arxiv.org/abs/1711.05101`.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1525–1534, 2016.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37: 30811–30849, 2024.

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.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16. IEEE, 2020.

Ishaan Shah, Anthony M Polloreno, Karl Stratos, Philip Monk, Adarsh Chaluvaraju, Andrew Hojel, Andrew Ma, Anil Thomas, Ashish Tanwer, Darsh J Shah, et al. Practical efficiency of muon for pretraining. *arXiv preprint arXiv:2505.02222*, 2025.

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

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

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

Benjamin Thérien, Xiaolong Huang, Irina Rish, and Eugene Belilovsky. Muloco: Muon is a practical inner optimizer for diloco. *arXiv preprint arXiv:2505.23725*, 2025.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 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.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL `https://arxiv.org/abs/1905.07830`.

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

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection. *arXiv preprint arXiv:2403.03507*, 2024.

Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

A Related work

Efficient Optimizers There has been some prior work on alleviating the memory cost of training with optimizers like AdamW. Techniques like low-rank adaptation (LoRA) (Hu et al., 2022) allow a small subset of parameters to be fine-tuned for downstream tasks, but often fall behind in quality when compared to full parameter fine-tuning (Biderman et al., 2024). Adafactor (Shazeer & Stern, 2018) factorizes the second moment for matrices, reducing memory consumption compared to AdamW. Dettmers et al. (2021) introduced the 8-bit Adam optimizer, but it only works when combined with careful blockwise + dynamic quantization. Adam GaLore (Zhao et al., 2024) leverages the low-rank structure of the gradients to reduce state size and can be combined with state quantization. Our work is directly comparable to Dettmers et al. (2021)'s work, and can potentially be combined with low-rank updates.

Gradient Orthogonalization The Muon optimizer (Jordan et al., 2024) has sparked interest in algorithms that take advantage of the orthonormalization of the gradient of matrix-valued parameters. Muon makes each weight matrix update orthonormal through polar decomposition (Newton-Schulz), giving direction-only, spectrally controlled steps for hidden layers. To scale it to large LLMs, recent works add decoupled weight decay and careful per-parameter update scaling (Liu et al., 2025). Dion (Ahn et al., 2025) leverages orthonormalization but is built for distributed training: using low-rank orthonormalization with device-local momentum/error-feedback to avoid reconstruction or synchronization of full matrices.

Quantization Quantization is a versatile tool for managing the memory cost of training large models. While we apply quantization to the state of the Muon optimizer, it has also been successfully applied to model weights in two settings - post-training quantization (PTQ) and quantization-aware training (QAT). PTQ uses calibration data to quantize the weights of large models in one shot to k bits, where k can be as low as 1 or 2 (Tseng et al., 2024; Frantar et al., 2022; Lin et al., 2024; Behdin et al., 2023). QAT involves training with quantized weights (Liu et al., 2023). Both PTQ and QAT require hardware support to realize the full benefits of quantization. Another interesting work is MuLoCo (Thérien et al., 2025), where the authors apply Muon as the inner (local) optimizer in a DiLoCo-style (Douillard et al.) loop, geared toward compressing parameter updates during distributed training. Thus, MuLoCo/DiLoco use quantization only for gradient updates.

B Additional tables

B.1 Quantization state variants

Model	**Muon state**	**AdamW state**	**Shorthand**
32-bit AdamW	—	32b	`AdamW-32`
32-bit Muon	32b	32b	`Muon-32`
8-bit AdamW (dynamic)	—	8b D	`AdamW-8D`
8-bit Muon (linear), 8-bit AdamW (dynamic)	8b L	8b D	`Muon-8L/AdamW-8D`
8-bit Muon (linear), 32-bit AdamW	8b L	32b	`Muon-8L/AdamW-32`
8-bit Muon (dynamic), 8-bit AdamW (dynamic)	8b D	8b D	`Muon-8D/AdamW-8D`
8-bit Muon (dynamic), 32-bit AdamW	8b D	32b	`Muon-8D/AdamW-32`

Table 4: Optimizer variants considered for pre-training and SFT. D and L denote dynamic and linear quantization, respectively. The 8-bit Muon family comprises five studied variants. AdamW-8L related variants are unstable under Theorem 1, consistent with empirical observations, and hence omitted from the table.

B.2 Pre-training downstream performance

Model size	Optimizer	ARC-C	ARC-E	BoolQ	HellaSwag	LAMBADA	MMLU	Avg.
Small	`Muon-32`	$0.204_{0.008}$	$0.503_{0.004}$	$0.571_{0.028}$	$0.285_{0.002}$	$0.203_{0.006}$	$0.230_{0.001}$	0.332
	`Muon-8L/AdamW-32`	$0.211_{0.003}$	$0.505_{0.007}$	$0.574_{0.031}$	$0.285_{0.001}$	$0.202_{0.008}$	$0.230_{0.001}$	**0.335**
	`Muon-8L/AdamW-8D`	$0.205_{0.009}$	$0.506_{0.008}$	$0.580_{0.023}$	$0.283_{0.001}$	$0.202_{0.006}$	$0.230_{0.000}$	0.334
Medium	`Muon-32`	$0.236_{0.005}$	$0.564_{0.009}$	$0.606_{0.010}$	$0.316_{0.001}$	$0.272_{0.004}$	$0.231_{0.004}$	0.371
	`Muon-8L/AdamW-32`	$0.247_{0.009}$	$0.563_{0.004}$	$0.613_{0.007}$	$0.315_{0.002}$	$0.274_{0.006}$	$0.232_{0.003}$	**0.374**
	`Muon-8L/AdamW-8D`	$0.235_{0.008}$	$0.559_{0.010}$	$0.595_{0.011}$	$0.314_{0.002}$	$0.272_{0.006}$	$0.230_{0.002}$	0.368
Large	`Muon-32`	$0.281_{0.008}$	$0.620_{0.014}$	$0.606_{0.012}$	$0.355_{0.002}$	$0.342_{0.005}$	$0.250_{0.006}$	0.409
	`Muon-8L/AdamW-32`	$0.278_{0.006}$	$0.626_{0.007}$	$0.610_{0.010}$	$0.356_{0.002}$	$0.338_{0.004}$	$0.251_{0.009}$	**0.410**
	`Muon-8L/AdamW-8D`	$0.272_{0.008}$	$0.628_{0.005}$	$0.611_{0.007}$	$0.355_{0.002}$	$0.335_{0.007}$	$0.245_{0.013}$	0.408
XL	`Muon-32`	$0.296_{0.006}$	$0.653_{0.012}$	$0.607_{0.015}$	$0.382_{0.002}$	$0.376_{0.007}$	$0.255_{0.006}$	**0.428**
	`Muon-8L/AdamW-32`	$0.298_{0.004}$	$0.651_{0.011}$	$0.603_{0.017}$	$0.381_{0.002}$	$0.373_{0.003}$	$0.253_{0.009}$	0.427
	`Muon-8L/AdamW-8D`	$0.293_{0.005}$	$0.649_{0.007}$	$0.586_{0.037}$	$0.379_{0.001}$	$0.368_{0.004}$	$0.254_{0.006}$	0.422
XXL	`Muon-32`	$0.354_{0.010}$	$0.713_{0.003}$	$0.618_{0.005}$	$0.428_{0.001}$	$0.424_{0.003}$	$0.261_{0.006}$	**0.466**
	`Muon-8L/AdamW-32`	$0.352_{0.004}$	$0.707_{0.004}$	$0.618_{0.008}$	$0.428_{0.002}$	$0.422_{0.008}$	$0.256_{0.006}$	0.464
	`Muon-8L/AdamW-8D`	$0.347_{0.008}$	$0.706_{0.013}$	$0.625_{0.009}$	$0.425_{0.002}$	$0.418_{0.005}$	$0.260_{0.010}$	0.463

Table 5: Downstream performance across 5 seeds, reported as mean with standard deviation.

B.3 Optimizer State Memory Usage

Model Size	`AdamW-32`	`Muon-32`	`AdamW-8D`	`M-8L(D)/A-32`	`M-8L(D)/A-8D`
XL	10.54	6.04	2.63	2.66	**1.51**
XXL	20.67	11.30	5.17	4.27	**2.82**

Table 6: Optimizer states memory (GiB) for two model configurations (lower is better). `M-8L(D)/A-8D` yields the smallest optimizer-state footprint in both cases.

C Training details

C.1 Pre-training Details

We train five GPT2-style models using 8–32 B200 GPUs with PyTorch Distributed DataParallel (DDP). Table 7 summarizes the model architectures, and Table 8 summarizes the pretraining hyperparameters. The general optimizer setup is as follows:

We pre-train the GPT models from scratch on FineWeb. For AdamW, we use $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\epsilon = 10^{-8}$. For Muon, we set the momentum parameter to 0.95. For quantized AdamW and Muon variants, we use a block size of 2048. We fix the decoupled weight decay to 0.1 in all experiments.

We adopt the warmup-stable-decay (WSD) learning rate schedule (Hu et al., 2024), with linear warmup from 0 to the peak learning rate over the first 10% of training steps, and linear decay to zero over the final 10% of steps. For each model size, we use the peak learning rate and global batch size recommended by Brown et al. (2020), which were tuned for GPT-style models trained using AdamW. For Muon, we do not tune the peak learning rate; instead, we reuse the AdamW peak learning rates, consistent with our goal that Muon should serve as a drop-in replacement for AdamW without requiring learning-rate or weight-decay re-tuning (Liu et al., 2025).

C.2 Fine-tuning

We use SFT on the GPT-XL model using the learning rate of 2×10^{-5}. For each optimizer with AdamW, we use $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We fixed the weight decay to be 0.01.

Model size	**Parameters**	d_{model}	n_{layers}	n_{heads}	**FF Ratio**
Small	162M	768	12	12	4
Medium	405M	1024	24	16	4
Large	834M	1536	24	16	4
XL	1.4B	2048	24	16	4
XXL	2.7B	2560	32	32	4

Table 7: Model sizes used for pre-training. Parameter counts are slightly different than those reported by (Brown et al., 2020) as the full architectural details were not reported. As a result, subtle differences in architecture as vocabulary size, GQA vs MHA and others may impact final model size.

Model size	**LR**	**Local BS**	**Grad. Acc. Steps**	**N. GPUs**	**Global BS**
Small	$6 \cdot 10^{-4}$	65536	1	8	0.5M
Medium	$3 \cdot 10^{-4}$	65536	1	8	0.5M
Large	$2.5 \cdot 10^{-4}$	65536	1	8	0.5M
XL	$2 \cdot 10^{-4}$	32768	1	32	1M
XXL	$1.6 \cdot 10^{-4}$	32768	2	32	1M

Table 8: Training configuration for the pre-training task. LR refers to the learning rate, and BS refers to the batch size. Both local and global batch sizes reported are in number of tokens, as all models were trained with a context length of 2048.

C.3 ImageNet training details

We trained ResNet-50 on ImageNet for 90 epochs using two H100 GPUs with PyTorch DistributedDataParallel. The schedule was the standard 90-epoch multi-step regime with learning rate decays at epochs 30, 60, and 80. For SGD with momentum we used a batch size of 128 per GPU (256 total), momentum 0.9, and weight decay 10^{-4}. For AdamW we used a learning rate of 3×10^{-3} and weight decay 10^{-2}. Hyperparameters were held fixed across FP32 and quantized runs. Training images were augmented with random resized crops to 224×224 and random horizontal flips. At evaluation time, images were resized to 256 pixels on the short side and center-cropped to 224×224, followed by normalization with the standard ImageNet mean and variance.

In the quantized variants, optimizer states were stored in 8-bit linear form with per-tensor scaling (Definition 1 applied layer-wise). For SGD with momentum, only the momentum buffer was quantized. For AdamW, both the first- and second-moment estimates were quantized. At each step, stored values were dequantized for computation, updated, and then requantized. Model weights, gradients, and activations were always maintained in FP32.

Figure 2 reports validation accuracy during training. Quantized SGD matches the FP32 baseline throughout. AdamW with FP32 optimizer states achieves slightly lower accuracy, while the quantized AdamW variant diverged immediately and is not shown.

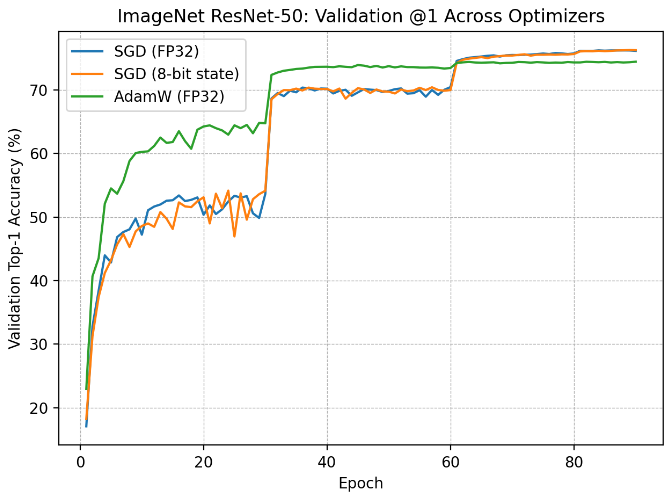

Figure 2: Validation top-1 accuracy on ImageNet for ResNet-50. Quantized SGD overlaps with the FP32 baseline. AdamW with FP32 states underperforms slightly, while the quantized AdamW variant diverged at the first step and is not shown.

D Quantization error bounds

In this appendix we formalize the 8-bit linear quantization operator and provide detailed proofs of the error bounds for Adam and SGD with momentum under quantization. We adopt the same notation as in Algorithm 1: $\mathbf{g}$ is the stochastic gradient, $\mathbf{m}$ and $\mathbf{v}$ are the first- and second-moment accumulators, $\mathbf{g}^2$ denotes the entrywise square, $\sqrt{\mathbf{v}}$ denotes the entrywise square root, and vec($\mathbf{M}$) denotes the vectorization of $\mathbf{M}$.

D.1 Quantization operator

Definition 1 *(Linear quantization). For a given vector $\mathbf{x} \in \mathbb{R}^d$ denoting the optimizer state of some algorithm, the 8-bit linear quantization is denoted by $Q : \mathbb{R}^d \to \mathbb{R}^d$, where:*

$$[Q(\mathbf{x})]_i = \frac{\|\mathbf{x}\|_\infty}{127} \cdot \operatorname{round}\left(\frac{127 \cdot \mathbf{x}_i}{\|\mathbf{x}\|_\infty}\right).$$

That is, each coordinate of $\mathbf{x}$ is mapped to the nearest grid point in a uniform partition of $[-\|\mathbf{x}\|_\infty, \|\mathbf{x}\|_\infty]$ into 256 representable levels (corresponding to signed 8-bit integers from -128 to 127), then rescaled back to floating point. This is the standard max-abs scaling scheme used in prior 8-bit quantization work. Note that, although we work over the reals, this definition effectively models the quantization and de-quantization steps applied to optimizer states between iterations.

D.2 Proofs

D.2.1 Proof of Theorem 1

Proof

As stated in the theorem, we work at the first step ($t = 1$) and suppress iterate superscripts for notational clarity. By the moment definitions in Algorithm 1, we have $\mathbf{m} = \mathbf{g}$ and $\mathbf{v} = \mathbf{g}^2$ (entry-wise), so for each coordinate i we have $\sqrt{\mathbf{v}_i} = |\mathbf{g}_i|$. We first lower bound the per-coordinate deviation $\left|\frac{\mathbf{m}_i}{\sqrt{\mathbf{v}_i}+\epsilon} - \frac{Q(\mathbf{m}_i)}{\sqrt{Q(\mathbf{v}_i)}+\epsilon}\right|$ and then sum over i.

$$\mathbb{P}\left(\left(\frac{\mathbf{m}_i}{\sqrt{\mathbf{v}_i}+\epsilon} - \frac{Q(\mathbf{m}_i)}{\sqrt{Q(\mathbf{v}_i)}+\epsilon}\right)^2 \geq t^2\right) = \mathbb{P}\left(\left|\frac{\mathbf{m}_i}{\sqrt{\mathbf{v}_i}+\epsilon} - \frac{Q(\mathbf{m}_i)}{\sqrt{Q(\mathbf{v}_i)}+\epsilon}\right| \geq t\right)$$
$$\geq \mathbb{P}\left(\frac{|\mathbf{g}_i| - \|\mathbf{g}\|_\infty/127}{\sqrt{Q(\mathbf{v}_i)}+\epsilon} - \frac{|\mathbf{g}_i|}{\sqrt{\mathbf{v}_i}+\epsilon} \geq t \text{ and } \mathbf{v}_i \geq Q(\mathbf{v}_i)\right)$$

Note that $\sqrt{\mathbf{v}_i} = \sqrt{\mathbf{g}_i^2} = |\mathbf{g}_i|$ and $Q(\mathbf{v}_i) = 0$ implies $\mathbf{v}_i \geq Q(\mathbf{v}_i)$. Hence, following from above,

$$\mathbb{P}\left(\left(\frac{\mathbf{m}_i}{\sqrt{\mathbf{v}_i}+\epsilon} - \frac{Q(\mathbf{m}_i)}{\sqrt{Q(\mathbf{v}_i)}+\epsilon}\right)^2 \geq t^2\right) \geq \mathbb{P}\left(\frac{|\mathbf{g}_i| - \|\mathbf{g}\|_\infty/127}{\epsilon} - \frac{|\mathbf{g}_i|}{|\mathbf{g}_i|+\epsilon} \geq t \text{ and } Q(\mathbf{v}_i) = 0\right)$$
$$\geq \mathbb{P}\left(\frac{|\mathbf{g}_i| - \|\mathbf{g}\|_\infty/127}{\epsilon} - 1 \geq t \text{ and } Q(\mathbf{v}_i) = 0\right)$$

Define the event $E_i := \{\frac{\|\mathbf{g}\|_\infty}{60} < |\mathbf{g}_i| \leq \frac{\|\mathbf{g}\|_\infty}{16}\}$. By the assumption of the theorem, $\mathbb{P}(E_i) \geq \nu$, and on E_i we have both $Q(\mathbf{v}_i) = 0$ and

$$\frac{|\mathbf{g}_i| - \|\mathbf{g}\|_\infty/127}{\epsilon} \geq \frac{\|\mathbf{g}\|_\infty/60 - \|\mathbf{g}\|_\infty/127}{\epsilon}$$
$$\geq \frac{\|\mathbf{g}\|_\infty}{128\epsilon}.$$

Since $\|\mathbf{g}\|_\infty \geq g_\infty$ almost surely and $g_\infty \geq 256\epsilon$,

$$\frac{\|\mathbf{g}\|_\infty}{128\epsilon} - 1 \geq \frac{g_\infty}{128\epsilon} - 1$$
$$\geq \frac{1}{2} \cdot \frac{g_\infty}{128\epsilon} = \frac{g_\infty}{256\epsilon}.$$

Set $\tau_0 := \frac{g_\infty}{256\epsilon}$. Then, for $t = \tau_0$,

$$\mathbb{P}\left(\left|\frac{\mathbf{m}_i}{\sqrt{\mathbf{v}_i}+\epsilon} - \frac{Q(\mathbf{m}_i)}{\sqrt{Q(\mathbf{v}_i)}+\epsilon}\right| \geq t\right) \geq \mathbb{P}(E_i)$$
$$\geq \nu.$$

Therefore, using $\mathbb{E}[X^2] \geq t^2\mathbb{P}(X \geq t)$ for nonnegative X,

$$\mathbb{E}\left[\left(\frac{\mathbf{m}_i}{\sqrt{\mathbf{v}_i}+\epsilon} - \frac{Q(\mathbf{m}_i)}{\sqrt{Q(\mathbf{v}_i)}+\epsilon}\right)^2\right] \geq \nu\,\tau_0^2 = \nu\frac{g_\infty^2}{(256\epsilon)^2}.$$

Summing over $i = 1, \ldots, d$ and using $\mathbf{w}^{(1)} - \tilde{\mathbf{w}}^{(1)} = \alpha\left(\frac{\mathbf{m}}{\sqrt{\mathbf{v}}+\epsilon} - \frac{Q(\mathbf{m})}{\sqrt{Q(\mathbf{v})}+\epsilon}\right)$,

$$\mathbb{E}\|\mathbf{w}^{(1)} - \tilde{\mathbf{w}}^{(1)}\|_2^2 \geq \alpha^2 d\nu \frac{g_\infty^2}{(256\epsilon)^2}.$$

■

D.2.2 Proof of Theorem 2

Proof We compare the two updates and isolate the effect of quantizing the momentum. The gradient terms cancel, leaving

$$\tilde{\mathbf{w}}^{(t+1)} - \mathbf{w}^{(t+1)} = -\eta(\mathbf{g}^{(t)} + \rho Q(\mathbf{m}^{(t)})) + \eta(\mathbf{g}^{(t)} + \rho\mathbf{m}^{(t)})$$
$$= -\eta\rho\,(Q(\mathbf{m}^{(t)}) - \mathbf{m}^{(t)}).$$

Taking squared norms and expanding coordinatewise yields

$$\|\tilde{\mathbf{w}}^{(t+1)} - \mathbf{w}^{(t+1)}\|_2^2 = \eta^2\rho^2 \sum_{i=1}^{d} \left(Q(\mathbf{m}^{(t)})_i - \mathbf{m}_i^{(t)}\right)^2.$$

By Definition 1, each coordinate is perturbed by at most $\|\mathbf{m}^{(t)}\|_\infty/127$, i.e., $|Q(\mathbf{m}^{(t)})_i - \mathbf{m}_i^{(t)}| \leq \|\mathbf{m}^{(t)}\|_\infty/127$. Applying this inside the sum gives

$$\begin{aligned}\|\tilde{\mathbf{w}}^{(t+1)} - \mathbf{w}^{(t+1)}\|_2^2 &\leq \eta^2\rho^2 \sum_{i=1}^{d} \left(\|\mathbf{m}^{(t)}\|_\infty/127\right)^2 \\ &= d\,\eta^2\rho^2 \left(\|\mathbf{m}^{(t)}\|_\infty/127\right)^2,\end{aligned}$$

which is the claimed bound. ■

D.2.3 Proof of Theorem 3

Proof From the two updates,

$$\widetilde{\mathbf{W}}^{(t)} - \mathbf{W}^{(t)} \;=\; -\alpha\big(\widetilde{\mathbf{O}}^{(t)} - \mathbf{O}^{(t)}\big) \quad\Rightarrow\quad \|\widetilde{\mathbf{W}}^{(t)} - \mathbf{W}^{(t)}\|_{\mathrm{F}} \;=\; \alpha\,\|\widetilde{\mathbf{O}}^{(t)} - \mathbf{O}^{(t)}\|_{\mathrm{F}}.$$

The momentum matrices $\mathbf{M}^{(t)}$ and $\widetilde{\mathbf{M}}^{(t)}$ are assumed to be full column rank with $\sigma_{\min}(\mathbf{M}^{(t)}), \sigma_{\min}(\widetilde{\mathbf{M}}^{(t)}) \geq s > 0$. For full-column-rank matrices, the (rectangular) polar-factor map satisfies

$$\|\widetilde{\mathbf{O}}^{(t)} - \mathbf{O}^{(t)}\|_{\mathrm{F}} \;\leq\; \frac{2}{\sigma_{\min}(\mathbf{M}^{(t)}) + \sigma_{\min}(\widetilde{\mathbf{M}}^{(t)})}\,\|\widetilde{\mathbf{M}}^{(t)} - \mathbf{M}^{(t)}\|_{\mathrm{F}} \;\leq\; \frac{1}{s}\,\|\widetilde{\mathbf{M}}^{(t)} - \mathbf{M}^{(t)}\|_{\mathrm{F}},$$

(see (Bhatia, 2013, Thm. VII.5.1(a)) and its extension to full column rank in (Li, 1995, Thm. 2)). Hence

$$\|\widetilde{\mathbf{W}}^{(t)} - \mathbf{W}^{(t)}\|_{\mathrm{F}} \;\leq\; \alpha\,\frac{1}{s}\,\|\widetilde{\mathbf{M}}^{(t)} - \mathbf{M}^{(t)}\|_{\mathrm{F}} \;=\; \alpha\,\frac{\beta}{s}\,\|Q(\mathbf{M}^{(t-1)}) - \mathbf{M}^{(t-1)}\|_{\mathrm{F}}.$$

By Definition 1, each entry changes by at most $\|\operatorname{vec}(\mathbf{M}^{(t-1)})\|_\infty/127$, so $\|Q(\mathbf{M}^{(t-1)}) - \mathbf{M}^{(t-1)}\|_{\mathrm{F}}^2 \;\leq\; d\big(\|\operatorname{vec}(\mathbf{M}^{(t-1)})\|_\infty/127\big)^2$. Squaring both sides completes the proof. ■

D.3 Adam algorithm

For completeness, we reproduce the base Adam algorithm below (Algorithm 1 in Kingma (2014)).

Algorithm 1 Adam (using $\mathbf{w}, \mathbf{g}, \mathbf{m}, \mathbf{v}$)

```
1: Input: step size $\alpha$, decay rates $\beta_1, \beta_2 \in [0,1)$, $\epsilon > 0$
2: Initialize: $\mathbf{w}^{(0)}$, $\tilde{\mathbf{m}}^{(0)} = \mathbf{0}$, $\tilde{\mathbf{v}}^{(0)} = \mathbf{0}$
3: for $t = 1, 2, \ldots$ do
4:     $\mathbf{g}^{(t)} \leftarrow \nabla f_t(\mathbf{w}^{(t-1)})$
5:     $\tilde{\mathbf{m}}^{(t)} \leftarrow \beta_1 \tilde{\mathbf{m}}^{(t-1)} + (1-\beta_1)\mathbf{g}^{(t)}$
6:     $\tilde{\mathbf{v}}^{(t)} \leftarrow \beta_2 \tilde{\mathbf{v}}^{(t-1)} + (1-\beta_2)\,(\mathbf{g}^{(t)})^2$
7:     $\mathbf{m}^{(t)} \leftarrow \tilde{\mathbf{m}}^{(t)}/(1-\beta_1^t)$
8:     $\mathbf{v}^{(t)} \leftarrow \tilde{\mathbf{v}}^{(t)}/(1-\beta_2^t)$
9:     $\mathbf{w}^{(t)} \leftarrow \mathbf{w}^{(t-1)} - \alpha\,\mathbf{m}^{(t)}\big/(\sqrt{\mathbf{v}^{(t)}} + \epsilon)$
10: end for
```

D.4 Exact Muon optimizer

For completeness, we provide the update formula for the exact Muon algorithm (without weight decay) here. As opposed to eqn. (1), this formula uses the exact polar factor $\mathbf{O} = \mathbf{U}^{(t)}\mathbf{V}^{(t)\top}$

$$\begin{aligned} \mathbf{M}^{(t)} &:= \beta\, \mathbf{M}^{(t-1)} + \mathbf{G}_t, \\ \mathbf{M}^{(t)} &= \mathbf{U}^{(t)}\mathbf{S}^{(t)}\mathbf{V}^{(t)\top} \text{ (thin SVD)}, \\ \mathbf{O}^{(t)} &:= \mathbf{U}^{(t)}\mathbf{V}^{(t)\top}, \\ \mathbf{W}^{(t)} &= \mathbf{W}^{(t-1)} - \alpha\, \mathbf{O}^{(t)}. \end{aligned} \tag{3}$$

In the quantized variant, only the previous momentum is quantized and then de-quantized, leading to the following updates where Q is defined in Definition 1:

$$\begin{aligned} \widetilde{\mathbf{M}}^{(t)} &:= \beta\, Q\big(\mathbf{M}^{(t-1)}\big) + \mathbf{G}_t, \\ \widetilde{\mathbf{M}}^{(t)} &= \widetilde{\mathbf{U}}^{(t)}\widetilde{\mathbf{S}}^{(t)}\widetilde{\mathbf{V}}^{(t)\top} \text{ (thin SVD)}, \\ \widetilde{\mathbf{O}}^{(t)} &:= \widetilde{\mathbf{U}}^{(t)}\widetilde{\mathbf{V}}^{(t)\top}, \\ \widetilde{\mathbf{W}}^{(t)} &= \mathbf{W}^{(t-1)} - \alpha\, \widetilde{\mathbf{O}}^{(t)}. \end{aligned} \tag{4}$$

D.5 Empirical Validation

In this section, we provide empirical validation for the theorems of Section 4.1 using a simple experimental setup. We train a two-layer fully-connected network with one hidden layer of width 256 and ReLU activation, mapping 28×28 pixel inputs to 10 logits. Inputs are normalized with the standard MNIST mean and variance $(0.1307, 0.3081)$, and we use the standard training split from `torchvision` with a batch size of 128 and cross-entropy loss.

Theorem 1 - We measure that approximately 29% of gradient entries in the loss gradient, $\mathbf{g}$, fall within the range $[\|\mathbf{g}\|_\infty/60, \|\mathbf{g}\|_\infty/16]$ from a Kaiming Uniform initialization of the model weights. This indicates that the gradient coordinate assumption of Theorem 1 holds in practice, and hence provides a realistic explanation for the divergence observed after a single step of Adam.

Theorem 2 - We measure the quantization error $\|\tilde{\mathbf{w}}^{(t+1)} - \mathbf{w}^{(t+1)}\|_2^2$ as a proportion of the error bound in Theorem 2 for the SGD+M optimizer with learning rate 0.1 and momentum 0.9. For all t over 1400 training steps, the ratio of the true squared ℓ_2 quantization error and the error bound stays between 0.06 and 0.085. By taking the square root of both sides in the error bound, we see that the true ℓ_2-norm quantization error is accurately predicted by the theoretical bound up to a small constant factor.

Theorem 3 - We measure the quantization error $\|\widetilde{\mathbf{W}}^{(t)} - \mathbf{W}^{(t)}\|_F^2$ of the first layer weight matrix for the Muon optimizer with $\alpha = 0.2$ and $\beta = 0.95$. We observe that the ratio of the true squared Frobenius-norm quantization error to the bound of Theorem 3 stays between 2×10^{-6} and 1.4×10^{-5}. The gap is explained by the fact that the proof of our theorem depends on the weak assumption that original and quantized matrices have full column rank r with the r-th singular value of each matrix lower bounded by s. In fact, this condition is pessimistic, as the perturbation of the weight matrix is not aligned with the r-th singular vector. More complex assumptions on the spectrum of $\mathbf{W}$, paired with probabilistic assumptions on the quantization noise and a polar-factor perturbation analysis that accounts for additional spectral information (see Li & Sun (2003), for example), would produce a tighter bound. However, this analysis is outside the scope of this work.

D.6 Training and validation loss during pretraining

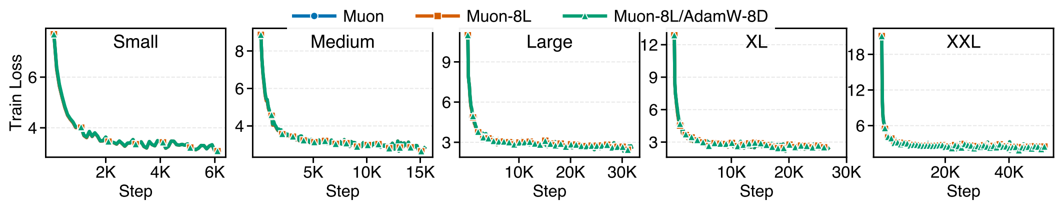

Figure 3: Pretraining training-loss curves for GPT-2 Small, Medium, Large, XL, and XXL with three optimizers: Muon, Muon-8L, and Muon-8L/AdamW-8D. Curves show the mean over 5 random seeds; error bars (seed-to-seed variation) are present but visually negligible.

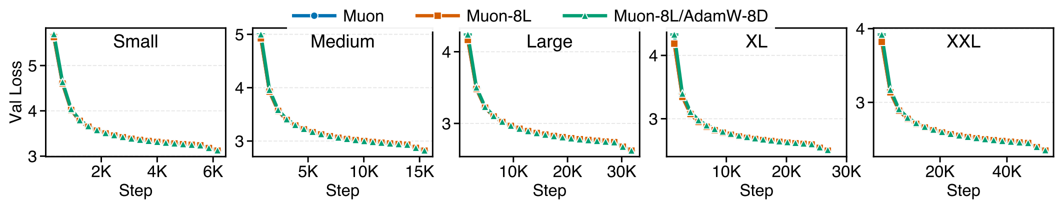

Figure 4: Pretraining validation-loss curves for GPT-2 Small, Medium, Large, XL, and XXL with three optimizers: Muon, Muon-8L, and Muon-8L/AdamW-8D. Curves show the mean over 5 random seeds; error bars (seed-to-seed variation) are present but visually negligible.

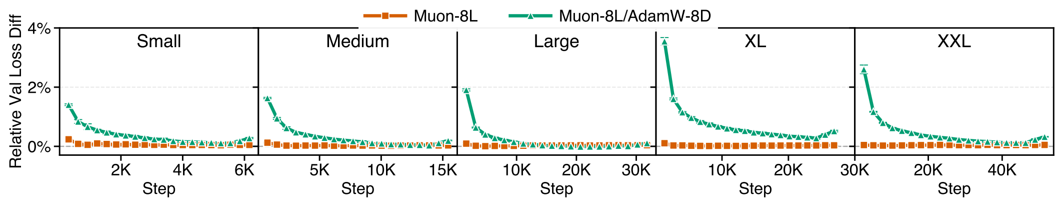

Figure 5: Relative pairwise validation-loss differences relative to the Muon baseline during pretraining for GPT-2 Small, Medium, Large, XL, and XXL.