MOSES: GEOMETRIC PRECONDITIONING AND HYBRID QUANTIZATION FOR MEMORY-EFFICIENT LLM OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizer states are the largest static memory cost in large model training. We introduce MOSES, a memory-efficient optimizer that addresses this challenge through two core innovations. First, Geometry-Aware Adaptive Preconditioning integrates gradient orthogonalization with factorized second-moment estimation, leveraging the geometry of the gradient space to provide both spatial smoothing and temporal adaptivity with negligible memory overhead. Second, Distribution-Aware Hybrid 1/4/8-bit Quantization adaptively allocates precision to post-preconditioning momentum based on the spectral properties of EMA updates, enabling aggressive compression while preserving optimization quality. Together, these innovations reduce optimizer state memory by 12.3× compared to Adam. Across model scales from 125M to 6.7B parameters, Moses achieves a compelling memory–performance trade-off that existing memory-efficient methods struggle to match, particularly excelling in the small-learning-rate regime that is characteristic of large-scale production training. Comprehensive evaluations across pretraining and downstream tasks show that Moses achieves competitive optimization quality while significantly improving memory efficiency, thereby reducing the hardware requirements for large language model training.

1 INTRODUCTION

Training large language models (Brown et al., 2020; Achiam et al., 2023; Team et al., 2023; Guo et al., 2025) faces a fundamental memory bottleneck: Adam (Kingma & Ba, 2015) stores two full-precision optimizer states per parameter, consuming $4\times$ the memory of 16-bit weights alone. For a model with N parameters, this amounts to $2N$ additional 32-bit scalars—a cost that severely constrains hardware scalability.

Existing approaches each face a hard memory floor. Quantized Adam variants (Dettmers et al., 2021) compress states to lower precision, achieving $0.5N$ memory but retaining Adam's two-state structure. Factorization methods such as Adafactor (Shazeer & Stern, 2018) reduce second-moment memory to sub-linear, yet practitioners routinely re-introduce a full momentum buffer ($1N$) for stability. Gradient orthogonalization methods such as Muon (Liu et al., 2025) eliminate the second-moment state entirely but structurally rely on a full-precision momentum buffer ($1N$). Consequently, quantized methods achieve $0.5N$ through precision reduction, while structural alternatives converge to a $1N$ floor. This raises a fundamental question: *Can we push optimizer memory well below these floors while retaining competitive convergence quality?*

We approach this through co-designing the optimizer pipeline with distribution-aware compression. The gradient operation pipeline shapes the statistical distribution of optimizer states; by structuring it to produce quantization-friendly distributions, we enable hybrid-precision quantization far beyond traditional schemes.

We introduce **Moses**, which reduces optimizer state memory by $12.3\times$ over Adam via two innovations. **Geometry-Aware Adaptive Preconditioning** integrates Newton-Schulz gradient orthogonalization with factorized second-moment estimation, providing spatial smoothing and temporal adaptivity with negligible memory overhead. **Distribution-Aware Hybrid Precision Momentum** allocates mixed 1/4/8-bit precision based on momentum magnitude statistics, reserving higher pre-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

cision for outliers while compressing the bulk of states. The allocation parameters are fully configurable, covering uniform 1/4/8-bit schemes and any combination thereof.

Our contributions are:

- **Moses**: a novel optimizer co-designing Geometry-Aware Adaptive Preconditioning with Distribution-Aware Hybrid Precision Momentum, achieving $12.3\times$ optimizer state compression over Adam with competitive convergence quality.
- **Theory**: we show that our pipeline acts as a distribution regularizer—reducing kurtosis and dynamic range to yield quantization-friendly momentum distributions—and establish the spectral properties of EMA as the foundation for hybrid-precision allocation.
- **Empirical validation**: We implement Moses with mixed-precision compression and evaluate it on GPT pretraining at 125M–6.7B scale and LLaMA-3.1 8B fine-tuning. Across these settings, Moses provides a favorable memory–performance trade-off relative to strong baselines, while reducing peak end-to-end training memory across all tested model scales.

2 Related Work

Optimizer state compression. Adafactor (Shazeer & Stern, 2018) factorizes second-moment estimates into row and column statistics, reducing memory from $O(mn)$ to $O(m+n)$, though disabling momentum can degrade convergence. 8-bit optimizers (Dettmers et al., 2021) quantize Adam's two states via block-wise quantization, achieving $\sim 75\%$ memory reduction with minimal performance loss across diverse tasks. More aggressive 4-bit variants (Li et al., 2023) push further but have been validated mainly in fine-tuning. Both apply uniform precision across all states, leaving room for more adaptive strategies.

Geometric preconditioning. Muon (Jordan, 2024) introduces gradient orthogonalization via iterative Newton–Schulz updates, decorrelating gradient components while storing only N-sized states. The Moonshot variant (Liu et al., 2025) further incorporates μP-based learning rate scaling (Yang et al., 2022), substantially improving training performance.

Low-rank gradient projection. GaLore (Zhao et al., 2024) projects gradients into a low-rank subspace via periodic SVD, reducing optimizer memory while preserving full-parameter learning. Apollo (Zhu et al., 2025) approximates per-parameter learning rate scaling via random projection, significantly reducing memory overhead while maintaining competitive convergence. Both methods decouple adaptive updates from full-rank state storage via gradient projection, though the quality of approximation depends on the choice of subspace.

3 Method

MOSES processes each gradient through two sequential stages. Geometry-Aware Adaptive Preconditioning first orthogonalizes and factored-scales the raw gradient, producing preconditioned updates with quantization-friendly statistical properties. Distribution-Aware Hybrid-Precision Momentum then accumulates these updates as EMA states and compresses them using a mixed precision quantization scheme. The complete procedure is given in Algorithm 1.

3.1 Geometry-Aware Adaptive Preconditioning

Traditional adaptive optimizers such as Adam and Adafactor operate on raw gradients that conflate two distinct sources of optimization difficulty. Given the gradient matrix $G_t \in \mathbb{R}^{m\times n}$ at step t, we identify two independent pathologies: **Spectral Ill-Conditioning**, quantified by $\kappa(G_t) = \sigma_{\max}(G_t)/\sigma_{\min}(G_t)$, where $\kappa(G_t) \gg 1$ concentrates gradient information in a narrow subspace; and **Coordinate-wise Scale Heterogeneity**, quantified by $r_t = \max_{ij}|G_{ij}|/\mathrm{median}_{ij}|G_{ij}|$, where $r_t \gg 1$ causes disproportionately large updates for a small fraction of parameters while the majority stagnate. We decouple these phenomena through a two-stage preconditioning strategy: orthogonalization-based spatial normalization followed by factorized temporal estimation.

Algorithm 1 MOSES Optimizer

Require: Learning rate η, momentum coefficients (β_1, β_2), stabilization term ε, clipping threshold τ_{clip}

Initialize: $M_0^{\text{packed}} \leftarrow \emptyset$, $R_0 \leftarrow 0 \in \mathbb{R}^m$, $C_0 \leftarrow 0 \in \mathbb{R}^n$, $V_0 \leftarrow 0$
for $t = 1, 2, \ldots$ **do**
  $G_t \leftarrow \nabla_\theta \mathcal{L}(\theta_{t-1})$
  if G_t is gradient of dense layer **then**
    $\hat{G}_t \leftarrow \texttt{NewtonSchulz}(G_t)$
  else
    $\hat{G}_t \leftarrow G_t$
  end if
  if $\dim(\hat{G}_t) \geq 2$ **then**
    $R_t \leftarrow \beta_1 R_{t-1} + (1-\beta_1)\,\mathbb{E}_j[\hat{G}_t^2 + \varepsilon]$
    $C_t \leftarrow \beta_1 C_{t-1} + (1-\beta_1)\,\mathbb{E}_i[\hat{G}_t^2 + \varepsilon]$
    $P_t \leftarrow \left(\frac{R_t}{\mathbb{E}[R_t]}\right)^{-1/2} \mathbf{1}^\top \odot \mathbf{1}\, C_t^{-1/2}$
    $U_t \leftarrow P_t \odot \hat{G}_t$
  else
    $V_t \leftarrow \beta_1 V_{t-1} + (1-\beta_1)(\hat{G}_t^2 + \varepsilon)$
    $U_t \leftarrow V_t^{-1/2} \odot \hat{G}_t$
  end if
  $\tilde{U}_t \leftarrow U_t \,/\, \max\!\big(1,\, \mathrm{RMS}(U_t)/\tau_{\text{clip}}\big)$
  if G_t is gradient of dense layer **then**
    $M_t \leftarrow \texttt{HybridDequantize}(M_{t-1}^{\text{packed}})$
    $M_t \leftarrow \beta_2 M_t + (1-\beta_2)\,\tilde{U}_t$
    $M_t^{\text{packed}} \leftarrow \texttt{HybridQuantize}(M_t)$
  else
    $M_t \leftarrow \beta_2 M_{t-1} + (1-\beta_2)\,\tilde{U}_t$
  end if
  $\theta_t \leftarrow \theta_{t-1} - \eta M_t$ {η adjusted for dense layers}
end for

3.1.1 Spatial Normalization via Gradient Orthogonalization

To eliminate spectral ill-conditioning, we apply Newton–Schulz iteration to orthogonalize gradient matrices directly. For $G_t \in \mathbb{R}^{m\times n}$ with $m \geq n$, we compute:

$$\hat{G}_t = \texttt{NewtonSchulz}(G_t), \tag{1}$$

where `NewtonSchulz` denotes 5 iterations of the Newton–Schulz algorithm. Each iteration applies the update rule:

$$X_{k+1} = \tfrac{1}{2}\, X_k \big(3I - X_k^\top X_k\big), \tag{2}$$

initialized with $X_0 = G_t / \|G_t\|_F$.

This orthogonalization process projects G_t onto the Stiefel manifold, producing $\hat{G}_t$ with approximately orthonormal columns, so that all singular values become approximately unity. By equalizing the singular values while preserving the principal singular vectors, orthogonalization removes directional bias in the gradient geometry without altering its orientation in parameter space.

Unlike Muon (Jordan, 2024), which applies Newton–Schulz to the exponential moving average of gradients, we orthogonalize the raw gradient G_t directly to decouple spatial normalization from temporal preconditioning. We apply orthogonalization exclusively to weight matrices in self-attention and feed-forward layers. As shown in the top-left panel of Figure 1, this reduces $\kappa(G_t)$ from $\sim 10^5$ to $\sim 10^2$, effectively eliminating spectral imbalance.

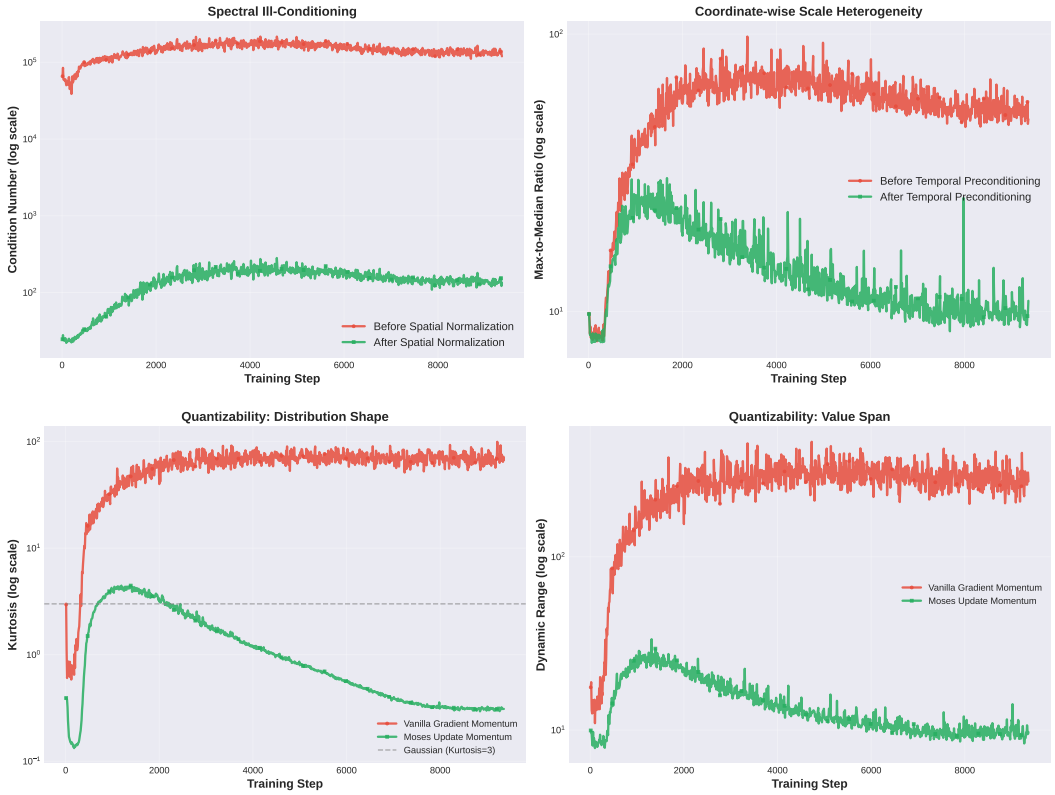


Figure 1: Effect of each preconditioning stage on gradient statistics (first dense MLP weight, GPT-3 350M pretraining). *Top row*: spectral condition number $\kappa(G_t)$ (left) reduced from $\sim 10^5$ to $\sim 10^2$ by gradient orthogonalization; max-to-median ratio r_t (right) normalized by factored adaptive scaling. *Bottom row*: kurtosis (left) reduced from ~ 60 to ~ 2 (Gaussian reference $= 3$); dynamic range (right) reduced by over $10\times$, enabling aggressive low-precision storage.

3.1.2 Temporal Preconditioning via Factored Variance Estimation

We then address Coordinate-wise Scale Heterogeneity through adaptive scaling based on temporal statistics. The key distinction from standard methods is that we estimate variance from the orthogonalized gradients $\hat{G}_t$ rather than raw gradients, ensuring that second-moment statistics reflect purely stochastic fluctuations on a geometrically normalized manifold. As shown in the top-right panel of Figure 1, factored adaptive scaling normalizes the coordinate-wise heterogeneity effectively.

We adopt the factorized variance representation from Adafactor (Shazeer & Stern, 2018), applied to $\hat{G}_t \in \mathbb{R}^{m\times n}$. Since orthogonalization has already smoothed the singular values of $\hat{G}_t$, the factored approximation $R_t \otimes C_t$ now captures the dominant structure of pure stochastic variance rather than a conflated mixture of geometry and noise. We track:

$$R_t = \beta_1 R_{t-1} + (1-\beta_1)\cdot \mathbb{E}_j[\hat{G}_t^2 + \varepsilon], \tag{3}$$

$$C_t = \beta_1 C_{t-1} + (1-\beta_1)\cdot \mathbb{E}_i[\hat{G}_t^2 + \varepsilon], \tag{4}$$

where $R_t \in \mathbb{R}^m$ and $C_t \in \mathbb{R}^n$ contain row-wise and column-wise second moments, reducing memory from $O(mn)$ to $O(m+n)$.

The preconditioned update $U_t = P_t \odot \hat{G}_t$ uses an adaptive scaling matrix $P_t = \mathbf{r}_t \mathbf{c}_t^\top$ with:

$$\mathbf{r}_t = \left(\frac{R_t}{\mathbb{E}[R_t]}\right)^{-1/2}, \qquad \mathbf{c}_t = C_t^{-1/2}. \tag{5}$$

The row normalization by $\mathbb{E}[R_t]$ ensures unit geometric mean, maintaining scale invariance while enabling per-parameter adaptation.

3.2 Distribution-Aware Hybrid-Precision Quantization

3.2.1 Exponential Moving Average of Preconditioned Updates

After obtaining $U_t = P_t \odot \hat{G}_t$ from Section 3.1, we apply RMS-based clipping:

$$\tilde{U}_t = \frac{U_t}{\max\big(1,\ \mathrm{RMS}(U_t)/\tau_{\text{clip}}\big)}, \tag{6}$$

where $\mathrm{RMS}(U_t) = \sqrt{\mathrm{mean}(U_t^2)}$ and τ_{clip} is a per-tensor threshold. While orthogonalization removes directional noise and factored preconditioning addresses coordinate-wise heterogeneity, the preconditioner itself adapts over time: as R_t, C_t evolve, P_t changes from step to step, causing $\tilde{U}_t$ to fluctuate even when the underlying gradient direction remains stable. Exponential moving average dampens these fluctuations, providing temporal smoothing for the adaptive preconditioning dynamics. We maintain:

$$M_t = \beta_2 M_{t-1} + (1-\beta_2)\,\tilde{U}_t, \tag{7}$$

with parameter update $\theta_t = \theta_{t-1} - \eta M_t$ and decoupled weight decay applied before the update step.

These sequential transformations fundamentally improve the quantizability of momentum states. As demonstrated in bottom panel of Figure 1, Moses update momentum exhibits **near-Gaussian** kurtosis ($\sim$2 vs. $\sim$60) and $10\times$ smaller dynamic range compared to vanilla gradient momentum as employed by Adam and Muon, enabling aggressive low-precision quantization that would be impractical for raw gradient moving averages.

Learning Rate Scaling for Dense Layers. Following the Moonshot version of Muon (Liu et al., 2025), we adjust the effective learning rate for a parameter matrix of shape $m \times n$ by a factor of $\rho \cdot \sqrt{\max(m,n)}$, where ρ is a configurable parameter. This follows the μP principle (Yang et al., 2022) that learning rates should scale with parameter matrix dimensionality.

3.2.2 Quantization Design

The EMA equation 7 acts as a first-order IIR low-pass filter: persistent optimization signals accumulate to large magnitudes while high-frequency stochastic noise is attenuated (Theorem A.1). We formalize two key consequences of this spectral structure; all proofs are deferred to Appendix A.

Theorem 3.1 (Quantization Error: Zero Bias and Bounded Variance)**.** *Let $\hat{M}_t = \beta_2 M_{t-1} + (1-\beta_2)\tilde{U}_t$ and $M_t = \mathcal{Q}(\hat{M}_t)$ with per-step error $e_t = M_t - \hat{M}_t$. Assuming $\{e_t\}$ are zero-mean and mutually independent with variance σ_e^2, the accumulated error $E_t = M_t - M_t^{\text{ideal}}$ satisfies:* (i) $\mathbb{E}[E_t] = 0$ *for all $t > 0$; and* (ii) $\mathrm{Var}(E_t) \to \sigma_e^2/(1-\beta_2^2)$ *at steady state.*

Theorem 3.1 implies that quantization inflates momentum variance but introduces *no systematic bias* in the descent direction, ensuring optimization validity in expectation.

Corollary 3.2 (Spectral Bias of Quantization Error)**.** *The error accumulation $E_t = \beta_2 E_{t-1} + e_t$ exhibits a DC-to-Nyquist gain ratio of $(1+\beta_2)/(1-\beta_2)$. For $\beta_2 = 0.9$, low-frequency errors are amplified $19\times$ relative to high-frequency ones.*

Corollary 3.2 motivates a magnitude-based bit allocation strategy: large-magnitude momentum entries correspond to dominant low-frequency signals that are disproportionately sensitive to quantization error and warrant high-precision storage, while small-magnitude entries are governed by suppressed high-frequency noise and are amenable to aggressive compression.

We partition momentum coordinates into three precision regions based on centered magnitude. Given momentum tensor m with mean μ and standard deviation σ, define $m_{\text{abs}} = |m - \mu|$ and thresholds $\tau_l = 0.5$, $\tau_h = 2.0$:

$$\mathrm{bits}\Big(m^{(i)}\Big) = \begin{cases} 8, & m_{\text{abs}}^{(i)} \geq \tau_h \cdot \sigma \quad (\mathcal{R}_{\text{high}}) \\ 4, & \tau_l \cdot \sigma \leq m_{\text{abs}}^{(i)} < \tau_h \cdot \sigma \quad (\mathcal{R}_{\text{mid}}) \\ 1, & m_{\text{abs}}^{(i)} < \tau_l \cdot \sigma \quad (\mathcal{R}_{\text{low}}) \end{cases} \tag{8}$$

Table 1: Ablation on the 1.3B GPT-3 model. All runs use 10K steps, a global batch of 128K tokens, and learning rate 2×10^{-4}. Memory compression is relative to Adam's optimizer state. **Bold** in (a) marks the default configuration used in subsequent experiments; Δ in (b) is the relative loss change from full Moses.

(a) Hyperparameters

			Block 64		Block 128	
τ_l	τ_h	ρ	**Mem.↑**	**Loss**	**Mem.↑**	**Loss**
0.5	2.0	0.10	12.0×	2.1768	12.3×	2.1764
0.5	2.0	0.05	12.0×	2.1428	**12.3×**	**2.1423**
0.5	2.5	0.10	12.3×	2.1809	12.7×	2.1800
0.5	2.5	0.05	12.3×	2.1424	12.7×	2.1444
1.0	2.0	0.10	14.9×	2.1767	15.2×	2.1755
1.0	2.0	0.05	14.9×	2.1625	15.2×	2.1619
1.0	2.5	0.10	15.4×	2.1745	15.7×	2.1742
1.0	2.5	0.05	15.4×	2.1633	15.7×	2.1636

(b) Components

Variant	**Loss↓**	Δ
Full Moses	**2.142**	–
No hybrid quant.	2.138	-0.2%
No RMS clipping	2.169	$+1.2\%$
No layerwise scaling	2.216	$+3.4\%$
No momentum	2.370	$+10.6\%$
NS $= 3$	2.315	$+8.1\%$
NS $= 2$	2.549	$+19.0\%$
No factored adapt.	4.240	$+98.0\%$
No orthogonalization	4.930	$+130.2\%$

A lightweight 2-bit metadata map $\mathcal{M}$ records region membership per element for efficient reconstruction. For $\mathcal{R}_{\text{high}}$ and $\mathcal{R}_{\text{mid}}$, we apply symmetric block-wise quantization with block size $B = 128$, computing local scaling factors to minimize quantization error. For $\mathcal{R}_{\text{low}}$, we apply binary quantization (1-bit) with a shared tensor-wise scaling factor $\alpha = \mathbb{E}[|m| \mid m \in \mathcal{R}_{\text{low}}]$, storing only the sign ($\text{sign}(0) = 1$).

3.2.3 Compression Analysis

For a near-Gaussian momentum distribution, approximately 38.3% of elements fall in $\mathcal{R}_{\text{low}}$, 57.1% in $\mathcal{R}_{\text{mid}}$, and 4.6% in $\mathcal{R}_{\text{high}}$; see Appendix B for the derivation and supporting histogram snapshots. The weighted payload bit-width is $1{\times}0.383 + 4{\times}0.571 + 8{\times}0.046 \approx 3.04$ bits. Including the 2-bit metadata map and block scaling overhead ($\approx$0.15 bits/element for $B = 128$), the total effective bit-width is:

$$\underbrace{3.04}_{\text{payload}} + \underbrace{2.00}_{\text{map}} + \underbrace{0.15}_{\text{scales}} \approx 5.19 \text{ bits per parameter.} \tag{9}$$

This corresponds to an effective compression ratio of approximately $12.3\times$ relative to Adam's 64-bit two-state optimizer representation.

4 Experiments

We evaluate Moses along three axes. First, we ablate its key hyperparameters and components on a 1.3B GPT-3 model. Second, we compare Moses with Adam (Kingma & Ba, 2015), Adafactor (Shazeer & Stern, 2018), Muon (Liu et al., 2025), GaLore (Zhao et al., 2024), Apollo (Zhu et al., 2025), 8-bit Adam (Dettmers et al., 2021), and 4-bit Adam (Li et al., 2023) on GPT-3 pretraining at 125M and 350M scales using the Pile (Gao et al., 2020) and the GPT-2 BPE tokenizer. We measure validation loss under a shared Chinchilla-optimal token budget (Hoffmann et al., 2022), and further include a 6.7B pretraining probe and supervised fine-tuning on LLaMA-3.1 8B (Grattafiori et al., 2024). Third, we analyze end-to-end training memory consumption and wall-clock time.

4.1 Ablation Analysis

We conduct hyperparameter and component ablations on a 1.3B GPT-3 model with 24 layers and hidden size 2048, as summarized in Table 1. All runs use the Pile, 10K training steps, a global batch of 128K tokens, and learning rate 2×10^{-4} following the GPT-3 training configuration. These ablations are used to select the default Moses configuration and to identify the contribution of each component.

Table 2: Validation loss on the Pile. Each optimizer is trained with the same five learning rates, $\{10^{-2}, 5\times10^{-3}, 10^{-3}, 5\times10^{-4}, 10^{-4}\}$. We report its best validation loss and the corresponding learning rate. The 125M (20K steps, $\sim$2.5B tokens) and 350M (60K steps, $\sim$7B tokens) runs use Chinchilla-optimal budgets and 128K tokens per step. The 6.7B run is a 10K-step probe ($\sim$0.66B tokens, 64K tokens per step). GaLore and Apollo use rank 256. Memory compression is relative to Adam's optimizer state. **Bold** denotes the lowest loss at each scale; — indicates that no learning rate in the sweep converged.

	125M, Chinchilla			**350M, Chinchilla**			**6.7B, 10K probe**		
Optimizer	**Mem.↑**	**Loss**	**Best LR**	**Mem.↑**	**Loss**	**Best LR**	**Mem.↑**	**Loss**	**Best LR**
Full precision									
Adam	1.0×	**2.26**	5×10^{-3}	1.0×	**2.07**	10^{-3}	1.0×	2.50	10^{-4}
Adafactor	2.0×	**2.26**	5×10^{-3}	2.0×	**2.07**	10^{-3}	2.0×	2.49	10^{-4}
Muon	2.0×	**2.26**	5×10^{-3}	2.0×	**2.07**	10^{-3}	2.0×	**2.28**	5×10^{-4}
Quantized									
Adam-8bit	4.0×	2.28	5×10^{-3}	4.0×	2.08	10^{-3}	4.0×	2.63	10^{-4}
Adam-4bit	8.0×	2.59	10^{-4}	8.0×	—	—	8.0×	2.79	10^{-4}
Low-rank									
GaLore	2.0×	2.36	10^{-2}	2.67×	2.20	5×10^{-3}	10.67×	2.46	5×10^{-3}
Apollo	3.0×	2.36	5×10^{-3}	4.0×	2.12	5×10^{-3}	16.0×	2.40	5×10^{-3}
Moses	12.3×	2.31	5×10^{-4}	12.3×	2.12	5×10^{-4}	12.3×	2.39	10^{-4}

Hyperparameters. We ablate the quantization block size B, thresholds τ_l and τ_h, and dense-layer update scale ρ. The lower threshold τ_l controls the main compression–quality trade-off: increasing it from 0.5 to 1.0 improves compression from about $12.3\times$ to over $15\times$, but consistently worsens validation loss. In contrast, B and τ_h have smaller effects. A larger block size slightly reduces scaling overhead, and increasing τ_h only mildly changes the compression ratio and loss. The update scale ρ has a clearer optimization effect: $\rho = 0.05$ consistently outperforms $\rho = 0.10$ without changing memory. We therefore use $B = 128$, $\tau_l = 0.5$, $\tau_h = 2.0$, and $\rho = 0.05$ as the default configuration for all subsequent experiments.

Components. The component ablation shows that orthogonalization and factored adaptation are essential, increasing loss by 130.2% and 98.0% when removed. Momentum and Newton–Schulz depth are also important: removing momentum increases loss by 10.6%, while reducing Newton–Schulz iterations from five to three or two increases loss by 8.1% and 19.0%, respectively. RMS clipping and layerwise scaling provide smaller but consistent gains. Finally, removing hybrid quantization changes loss by only 0.2%, indicating that the 1/4/8-bit momentum representation preserves optimization quality while providing $12.3\times$ optimizer-state compression. Under the same 1.3B protocol, directly applying Moses's hybrid quantizer to Muon increases validation loss from 2.150 to 2.215, suggesting that the quantizer is most effective when used together with Moses's full update structure.

4.2 Main Results

4.2.1 Pretraining

Because different optimizers may favor different learning-rate ranges, we train every optimizer with the same five learning rates following the sweep protocol of GaLore and Apollo, and report its best validation loss and corresponding learning rate. The 125M and 350M runs use Chinchilla-optimal token budgets, whereas the 6.7B run is a 10K-step probe due to limited computational budget.

Moses provides a strong memory–quality trade-off at $12.3\times$ optimizer-state compression. At 125M and 350M, Moses incurs only a moderate loss increase relative to full-precision and 8-bit Adam baselines, while using substantially less optimizer memory. It also matches or improves upon the

Table 3: Downstream performance after three epochs of Alpaca SFT from a pretrained LLaMA-3.1 8B base checkpoint. Tasks span three categories: commonsense reasoning (ARC-Challenge and ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020)), reading comprehension (BoolQ (Clark et al., 2019)) and factual truthfulness (TruthfulQA (Lin et al., 2022)). Avg. is the macro-average over the seven tasks. **Bold** denotes the best result in each column.

Optimizer	ARC-C	ARC-E	HSwag	WinoG	BoolQ	TruthQA	PIQA	Avg.
Adam	43.52	73.15	67.35	59.91	77.16	49.57	75.14	63.69
Adafactor	**45.39**	72.05	**71.38**	62.59	81.83	48.11	**76.44**	**65.40**
Muon	45.22	71.59	69.29	63.14	78.01	49.43	76.28	64.71
Adam-8bit	44.28	72.47	69.05	60.69	82.39	49.67	76.22	64.97
Adam-4bit	43.69	73.61	66.59	62.12	73.79	48.06	75.14	63.29
Apollo	41.64	69.57	68.77	61.01	80.76	50.63	75.79	64.02
GaLore	43.17	70.24	68.11	62.83	77.19	47.77	74.65	63.42
Moses	43.00	**74.66**	67.39	**63.22**	**83.67**	**50.67**	74.70	65.33

low-rank methods at these scales. The advantage becomes more apparent in the 6.7B probe: Moses achieves the second-lowest loss overall, behind only Muon, while using over $6\times$ less optimizer-state memory than Muon. Moses also consistently outperforms Adam-4bit, which fails to converge at 350M, indicating that aggressive uniform quantization is insufficient for stable pretraining.

The selected learning rates reveal a clear scaling trend. GaLore and Apollo favor relatively large learning rates, typically 5×10^{-3} or higher, whereas Moses performs best at 5×10^{-4} or lower. The Adam family shifts from 5×10^{-3} at 125M to 10^{-4} at 6.7B, and Muon shows a similar decrease with scale. This trend is favorable to Moses: under μP, the optimal learning rate decreases with model width (Yang et al., 2022), suggesting that larger-scale training may naturally move toward the small-learning-rate regime where Moses remains competitive.

4.2.2 Supervised Fine-tuning

We fine-tune a pretrained LLaMA-3.1 8B base checkpoint (Grattafiori et al., 2024) on Alpaca (Taori et al., 2023) for three epochs and evaluate it on seven downstream benchmarks, as shown in Table 3. Moses achieves a macro-average of **65.33%**, closely matching Adafactor (65.40%) and exceeding Adam by 1.64 points. It obtains the best result on four of the seven tasks: ARC-Easy, WinoGrande, BoolQ, and TruthfulQA. The largest gain over Adam appears on BoolQ, where Moses improves accuracy from 77.16% to 83.67%.

Moses also outperforms both quantized baselines, Adam-8bit and Adam-4bit, in macro-average accuracy. This indicates that its memory savings do not come at the cost of downstream quality. Together with the pretraining results, the SFT results suggest that Moses transfers well across model architectures, training stages, and evaluation settings.

4.3 System Performance

We evaluate system performance on a single NVIDIA H800 server using Megatron-LM Shoeybi et al. (2019). The memory experiments use data parallelism without other forms of parallelism, while the GPT-3 13B runtime measurements use tensor parallelism of 4 and gradient accumulation.

Memory efficiency. Moses compresses optimizer states by $12.3\times$ relative to Adam. As shown in Table 4(a), this state compression translates into lower end-to-end peak GPU memory after accounting for model parameters, gradients, and activations. At batch size 1, which minimizes activation memory and exposes the static optimizer-state footprint, Moses uses the least memory at every model scale. At 1.3B, it consumes 14.35 GB, compared with 22.12 GB for Adam, 15.39 GB for Adam-4bit, and 14.55 GB for Apollo. Under the large-batch configuration, Moses similarly reduces peak memory from Adam's 42.12 GB to 33.50 GB. The end-to-end memory reduction over Adam increases from 3.4% at 125M to 8.5% at 350M and 20.5% at 1.3B, as optimizer states account for an increasing fraction of the measured memory footprint.

Table 4: Left: peak GPU memory across model scales and batch sizes, where BS is the per-rank batch size. BS=1 highlights optimizer-state memory, while BS=64 represents large-batch training. Right: 13B runtime with the GPT-3 global batch size of 2M tokens, reported as average seconds per step. Overhead is relative to Adam. Runtime numbers are averaged over 100 steps, except Adam-4bit, which uses 32 steps due to later overflow. **Bold** denotes the best memory result in each column.

(a) Peak GPU memory (GB) ↓

	BS=1			BS=64		
Opt.	125M	350M	1.3B	125M	350M	1.3B
Adam	2.18	6.02	22.12	21.43	26.37	42.12
Adafactor	1.99	5.08	17.38	20.97	25.05	37.23
Muon	2.14	5.28	18.37	21.12	25.24	37.62
Adam-8bit	1.99	4.76	16.17	20.97	24.74	35.44
Adam-4bit	2.06	4.64	15.39	20.90	24.43	34.28
GaLore	2.15	5.11	16.04	21.14	25.08	35.75
Apollo	2.08	4.82	14.55	21.06	24.78	34.43
Moses	**1.74**	**4.20**	**14.35**	**20.70**	**24.14**	**33.50**

(b) GPT-3 13B runtime ↓

Optimizer	**s/step**	**Overhead**
Adam	91.33	–
Adafactor	91.59	+0.3%
Apollo	91.78	+0.5%
Adam-8bit	91.83	+0.6%
Adam-4bit	92.33	+1.1%
Muon	92.82	+1.6%
GaLore	94.36	+3.3%
Moses	**92.95**	**+1.8%**

Runtime efficiency. Newton–Schulz orthogonalization and mixed-precision decoding, quantization, and packing introduce additional cost to each Moses update. However, optimizer computation depends primarily on the parameter count N, whereas forward and backward computation also grows with the number of tokens T processed per update. For optimizers invoked once per global batch, increasing the token batch therefore increases model computation without proportionally increasing optimizer computation.

Counting each multiply–add as two FLOPs, the forward and backward passes require approximately $2NT$ and $4NT$, respectively. With activation recomputation, an additional forward pass contributes another $2NT$, giving $C_{\text{model}} \approx 8NT$. For any optimizer, the additional cost scales as $\Delta C_{\text{opt}} \approx \alpha N$, where α captures both the arithmetic operations and implementation efficiency. For a fixed implementation, the relative overhead compared with Adam therefore scales as

$$\text{Overhead} \approx \frac{\Delta C_{\text{opt}}}{C_{\text{model}}} \approx \frac{\alpha N}{8NT} = \frac{\alpha}{8T}. \tag{10}$$

Thus, the relative optimizer overhead decreases as the token batch grows and need not dominate end-to-end training time.

Table 4(b) directly validates this large-batch regime on 13B training using the global batch size 2M tokens adopted by GPT-3. Moses takes 92.95 seconds per step, corresponding to only 1.8% overhead relative to Adam, while retaining its $12.3\times$ optimizer-state compression. This overhead is comparable to Muon, which incurs 1.6% overhead, showing that Moses achieves substantially stronger optimizer-state compression without introducing a larger runtime penalty than a practical orthogonalization-based optimizer. Overall, Adafactor, Apollo, Adam-8bit, Adam-4bit, Muon, and Moses all remain within 1.8% overhead over Adam, whereas GaLore incurs 3.3% overhead. These results further support our theoretical analysis: in practical large-scale LLM training, optimizer computation accounts for only a small fraction of end-to-end iteration time, so Moses can substantially reduce optimizer-state memory while introducing negligible overall runtime overhead.

5 Conclusion

We presented MOSES, a memory-efficient optimizer that reduces optimizer-state memory by $12.3\times$ compared with Adam. By combining geometry-aware preconditioning with distribution-aware hybrid quantization, MOSES reshapes momentum states into quantization-friendly representations, enabling low-precision storage without compromising convergence.

This reduction not only saves HBM memory, but also eases state sharding, checkpointing, and communication in distributed training. Together with negligible runtime overhead, MOSES provides a practical path toward more efficient large-scale model training.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the text to improve grammatical clarity and readability. We have not used generative AI tools for generating core research ideas, writing the primary text, or synthesizing experimental datasets, and the rest of the required disclosure tasks are not applicable to this work. We have reviewed all AI-assisted work. Every text refinement provided by the language models was carefully examined and manually verified by the authors. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we have provided rigorous and self-contained descriptions of all proposed algorithms within the manuscript. Specifically, the core mathematical formulations, algorithmic steps, and theoretical foundations are clearly presented in the main text. Furthermore, all necessary implementation details, including training datasets, hyperparameter configurations, and empirical setups, are comprehensively documented in the Experiments section and detailed further in the Appendix.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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, pp. 7432–7439, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

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.

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

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.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Keller Jordan. Muon optimizer. GitHub repository, 2024. URL https://github.com/KellerJordan/Muon.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

Bingrui Li, Jianfei Chen, and Jun Zhu. Memory efficient optimizers with 4-bit states. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=nN8TnHB5nw.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 3214–3252, 2022.

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.

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.

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.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

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.

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. *Proceedings of Machine Learning and Systems*, 7, 2025.

A Theoretical Analysis of Quantization Error Dynamics

We provide proofs for Theorem 3.1 and Corollary 3.2 stated in Section 3.2. We first establish the low-pass filtering property of EMA as a foundational result, which both proofs rely on.

Theorem A.1 (EMA as a Low-Pass Input Signal Filter)**.** *The EMA update $M_t = \beta_2 M_{t-1} + (1-\beta_2)\tilde{U}_t$ defines a first-order IIR low-pass filter with transfer function $H(z) = \frac{1-\beta_2}{1-\beta_2 z^{-1}}$ and power spectral density gain:*

$$\left|H(e^{j\omega})\right|^2 = \frac{(1-\beta_2)^2}{1+\beta_2^2-2\beta_2\cos\omega}. \tag{11}$$

For $\beta_2 = 0.9$, the filter provides unit gain at DC ($\omega = 0$) and ≈ -25.6 dB attenuation at the Nyquist frequency ($\omega = \pi$).

Proof of Theorem 3.1. The quantized momentum evolves as $M_t = \hat{M}_t + e_t = \beta_2 M_{t-1} + (1-\beta_2)\tilde{U}_t + e_t$, while the ideal counterpart satisfies $M_t^{\text{ideal}} = \beta_2 M_{t-1}^{\text{ideal}} + (1-\beta_2)\tilde{U}_t$. Subtracting, the accumulated error obeys $E_t = \beta_2 E_{t-1} + e_t$, which unrolls to:

$$E_t = \sum_{i=0}^{t} \beta_2^{t-i} e_i. \tag{12}$$

Since $\mathbb{E}[e_i] = 0$ for all i, linearity of expectation gives $\mathbb{E}[E_t] = 0$, proving (i). Under mutual independence of $\{e_t\}$ with common variance σ_e^2:

$$\mathrm{Var}(E_t) = \sum_{i=0}^{t} \beta_2^{2(t-i)}\sigma_e^2 \xrightarrow{t\to\infty} \sigma_e^2 \sum_{j=0}^{\infty}(\beta_2^2)^j = \frac{\sigma_e^2}{1-\beta_2^2}, \tag{13}$$

proving (ii). For $\beta_2 = 0.9$, the steady-state amplification factor is $1/(1-0.81) \approx 5.26$. □

Remark A.2. While quantization increases the variance of the momentum estimate, the zero-bias guarantee ensures the descent direction remains valid in expectation, provided the learning rate is appropriately scaled to account for the inflated variance.

Proof of Corollary 3.2. The error recursion $E_t = \beta_2 E_{t-1} + e_t$ is a first-order IIR system driven by e_t. By Theorem A.1 (with $(1-\beta_2)$ set to unity), its frequency response magnitude is:

$$\left|H(e^{j\omega})\right| = \left(1 - 2\beta_2\cos\omega + \beta_2^2\right)^{-1/2}. \tag{14}$$

Evaluating at $\omega = 0$ (DC) and $\omega = \pi$ (Nyquist):

$$\frac{|H(e^{j0})|}{|H(e^{j\pi})|} = \frac{(1-\beta_2)^{-1}}{(1+\beta_2)^{-1}} = \frac{1+\beta_2}{1-\beta_2}. \tag{15}$$

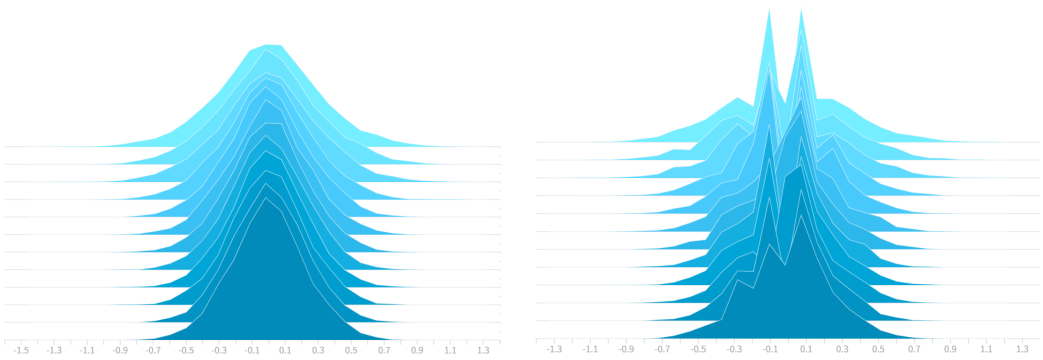
For $\beta_2 = 0.9$, this ratio equals $1.9/0.1 = 19$. □

Remark A.3 (Magnitude as a Proxy for Signal Dominance). Corollary 3.2 implies that large momentum magnitudes serve as a reliable proxy for dominant low-frequency components: the EMA filter structurally prevents high-frequency noise from accumulating to significant magnitudes via spectral suppression. This validates the magnitude-based thresholds $(\tau_l, \tau_h) = (0.5\sigma, 2.0\sigma)$ in Eq. equation 8—prioritizing bit-width for the large-magnitude regime preserves high-SNR components representing consistent optimization directions.

B Compression Ratio Derivation

For a standard Gaussian $Z \sim \mathcal{N}(0,1)$, the fraction of elements in each precision region under thresholds $\tau_l = 0.5$ and $\tau_h = 2.0$ is:

$$\begin{aligned} p_{\text{low}} &= P(|Z| < 0.5) \approx 0.383, \\ p_{\text{mid}} &= P(0.5 \le |Z| < 2.0) \approx 0.571, \\ p_{\text{high}} &= P(|Z| \ge 2.0) \approx 0.046. \end{aligned} \tag{16}$$



(a) Original Momentum (b) Quantized Momentum

Figure 2: Histogram snapshots of Moses update momentum for the first dense weight in an MLP layer during 350M GPT-3 pretraining. **Left:** The original momentum distribution is centered around zero and approximately Gaussian-like. **Right:** The quantized distribution reflects the three-tier mixed-precision representation, where small-magnitude values are stored with 1-bit precision and larger-magnitude values are stored with 4-bit or 8-bit precision.

The weighted average data payload bit-width is:

$$\bar{b} = 1 \cdot p_{\text{low}} + 4 \cdot p_{\text{mid}} + 8 \cdot p_{\text{high}} = 0.383 + 2.284 + 0.368 \approx 3.04 \text{ bits}. \tag{17}$$

A practical implementation incurs two additional overheads:

- **Metadata map** $\mathcal{M}$: 2 bits per element to record region membership, enabling efficient reconstruction without sorting or indexing.
- **Block scaling factors**: For $\mathcal{R}_{\text{high}}$ and $\mathcal{R}_{\text{mid}}$ with block size $B = 128$, each block stores one FP16 scale (16 bits), contributing $16/128 \approx 0.125$ bits per element. Weighted by $p_{\text{mid}} + p_{\text{high}} \approx 0.617$, the per-element overhead is ≈ 0.15 bits.

The total effective bit-width per parameter is therefore:

$$3.04 \text{ (payload)} + 2.00 \text{ (map)} + 0.15 \text{ (scales)} \approx 5.19 \text{ bits}. \tag{18}$$

Compared to Adam's two FP32 momentum states ($2 \times 32 = 64$ bits per parameter), MOSES achieves a compression ratio of $64 \,/\, 5.19 \approx \mathbf{12.3}\times$.

Figure 2 provides representative histogram snapshots collected during 350M GPT-3 pretraining. The original Moses update momentum is centered around zero and exhibits an approximately Gaussian-like shape. After quantization, the distribution reflects the intended three-tier mixed-precision structure: small-magnitude values are assigned to the 1-bit region, while larger-magnitude values are represented using 4-bit or 8-bit precision.

C Training Loss Curves: 6.7B Pretraining Probe

Figure 3 presents the full training dynamics of the 6.7B GPT-3 model on the Pile dataset. Across 10K pretraining steps, Moses maintains stable and smooth convergence. More importantly, Moses substantially outperforms all quantized optimizer baselines, including both Adam-8bit and Adam-4bit, and achieves competitive final validation loss among memory-efficient low-precision methods. Its final validation loss is second only to full-precision Muon, which does not impose the same aggressive optimizer-state compression. These results show that Moses delivers strong large-scale pretraining performance while substantially reducing optimizer-state memory.

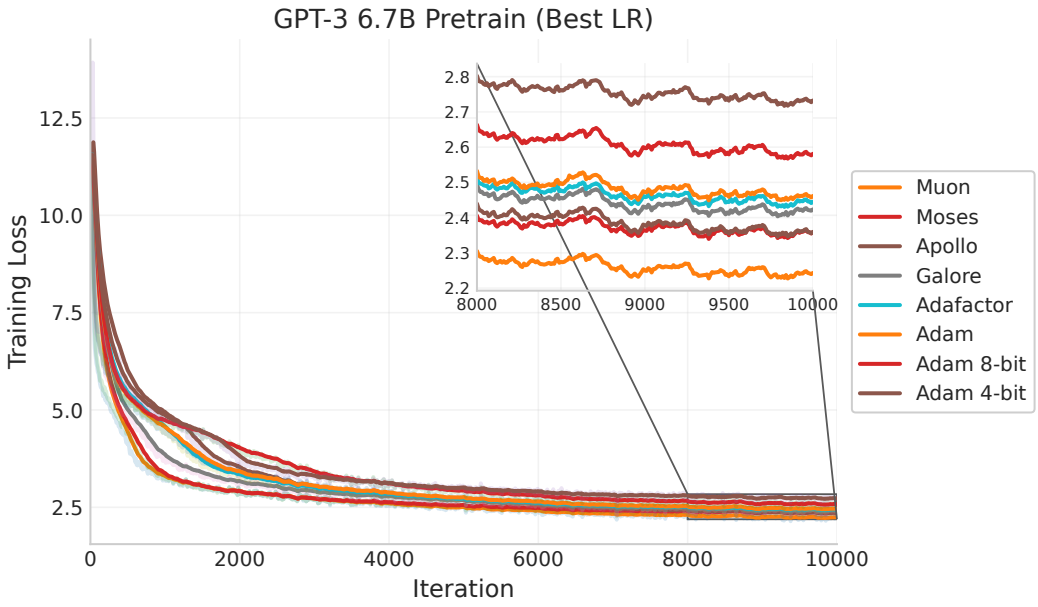


Figure 3: Training loss curves for the 6.7B GPT-3 model on the Pile dataset over 10K steps with a global batch size of 64K tokens. Each optimizer is evaluated at its best learning rate.

D Experimental Details

All pretraining experiments are conducted on GPT-3 models using the Pile dataset with a sequence length of 1024 tokens. We adopt a cosine learning rate schedule with a linear warmup over the first 5% of training steps, followed by decay to a minimum learning rate of 10% of the peak value over 90% of the total steps. Training is conducted in `fp16` with activation checkpointing enabled to reduce memory pressure independently of optimizer states. No dropout is applied, following standard practice for large-scale language model pretraining.

For the instruction-tuning experiments, we fine-tune LLaMA-3.1 8B Base on Alpaca-Cleaned with the LLaMA-3 chat template. The training hyperparameters follow the official Alpaca 7B recipe (Taori et al., 2023): a global batch size of 128 sequences, learning rate 2×10^{-5}, 3 epochs, weight decay 0, and a cosine schedule with 3% linear warmup. The maximum sequence length is 2048. Training is conducted in `bf16` with gradient checkpointing. For baseline optimizers, we follow the hyperparameter settings reported in their respective papers for the corresponding setting.