ROUNDING IN PRECONDITIONER SPACE: REDESIGNING 4-BIT ADAMW OPTIMIZER-STATE QUANTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantizing AdamW's optimizer states reduces persistent storage, but quantization errors propagate through the moment recurrences and perturb subsequent adaptive updates. We redesign 4-bit optimizer-state quantization for AdamW from the perspective of *rounding space*: the coordinate in which a quantizer chooses between adjacent reconstruction levels. For the second-moment, a local analysis of the quantization cell adjacent to zero shows that small mean state error does not ensure a small mean error in the next-step preconditioner. A one-dimensional quadratic construction further shows qualitatively different optimization dynamics under state-space and preconditioner-space rounding. These results motivate Zero-Inclusive Preconditioner-space Stochastic Rounding (**ZIP-SR**), which retains zero in the second-moment codebook and computes stochastic-rounding probabilities in preconditioner space. As a complementary route, Zero-Excluding EDEN calibration (**ZE-EDEN**) uses a zero-excluding second-moment codebook and rescales the quantized second-moment block to mitigate the preconditioner distortion caused by the positive quantization floor. Both configurations use 4-bit NormalFloat (NF4) for the first moment, with targeted stochastic rounding of the LM-head first moment during the final 10% of training. Across GPT- and Llama-style pretraining experiments ranging from **130M** to **2.7B** parameters, both methods reduce TorchAO 4-bit AdamW's validation-loss gap to 32-bit AdamW at every evaluated scale, with the largest reported gap reduction reaching **70%**. In full-parameter supervised fine-tuning, both recipes achieve lower validation loss than TorchAO while remaining close to 32-bit AdamW on downstream tasks.

1 INTRODUCTION

Adam (Kingma & Ba, 2015) and AdamW (Loshchilov & Hutter, 2019) are widely used optimizers for large-scale neural network training, but their first- and second-moment buffers, usually stored in full 32-bit precision and requiring 8 B/parameter in total, can exacerbate memory contention during training, motivating research on optimizer-state compression. Block-wise dynamic quantization enabled practical 8-bit optimizers, implemented in bitsandbytes (Dettmers et al., 2022) and TorchAO (Or et al., 2025). Li et al. (2023) extended optimizer-state quantization to 4 bits and identified the *zero-point failure*, in which mapping a small positive second-moment value to zero produces an excessively large update, and showed that excluding zero from the second-moment codebook prevents it. This strictly positive codebook is implemented in TorchAO's 4-bit AdamW and retained in subsequent studies on low-bit AdamW, via the same linear grid or alternative positive codebooks (Xu et al., 2025; Topollai & Choromanska, 2026).

Our key insight is that, when the codebook includes zero, whether the zero-point failure occurs depends critically on the rounding rule. Conventional rounding chooses between neighboring levels by their values in second-moment state-space, but AdamW transforms the second moment nonlinearly before it scales the update. As a result, rounding that is accurate, or unbiased, in state space can still distort the resulting update. By defining rounding decisions in preconditioner space, we retain the same zero-inclusive codebook while reducing the probability of harmful zero selections. More broadly, optimizer-state quantizers should account for how stored states affect optimization, not solely for reconstruction error in state space.

Armed with this key insight, our contributions are twofold.

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.

- First, we characterize how the rounding coordinate affects near-zero second-moment quantization at fixed reconstruction levels. A local zero-cell analysis and a scalar-quadratic construction show that it affects both preconditioner error and optimization dynamics. This motivates Update-SR, which assigns stochastic-rounding (SR) probabilities using the preconditioners induced by the candidate reconstruction levels.
- Second, we develop two 4-bit AdamW configurations. Zero-Inclusive Preconditioner-space Stochastic Rounding (**ZIP-SR**) combines a zero-inclusive second-moment codebook with Update-SR, whereas Zero-Excluding EDEN calibration (**ZE-EDEN**) combines a zero-excluding codebook with EDEN block calibration (Vargaftik et al., 2022). Both use NF4 for the first moment and targeted LM-head stochastic rounding motivated by matched checkpoint interventions. At the same persistent moment-storage cost as TorchAO 4-bit AdamW, both configurations reduce its mean validation-loss gap to 32-bit AdamW across the evaluated pretraining settings and attain lower mean SFT validation loss.

2 PRELIMINARIES

2.1 TERMINOLOGY AND NOTATION

We write $Q_{\phi,\rho,\mathcal{C}}$ for the decoded quantizer with rounding coordinate ϕ, rounding rule ρ, and decoded codebook $\mathcal{C}$[1]. Bare Q abbreviates a quantizer fixed by the surrounding context. For $t \geq 1$ with gradient $\mathbf{g}_t$, AdamW (Loshchilov & Hutter, 2019) forms the first and second-moments by

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t, \qquad \mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\mathbf{g}_t^2$$

and then updates

$$\mathbf{w}_t = (1-\eta_t\lambda)\mathbf{w}_{t-1} - \eta_t \frac{\mathbf{m}_t/(1-\beta_1^t)}{\sqrt{\mathbf{v}_t/(1-\beta_2^t)}+\epsilon},$$

with learning rate η_t, weight-decay λ, moment decay coefficients β_1, β_2, and denominator stabilizer $\epsilon > 0$; squares, square roots, and divisions are applied elementwise. Optimizer-state quantization is applied when the moments are written back to the buffers retained between optimizer steps. Let $\tilde{\mathbf{m}}_t$ and $\tilde{\mathbf{v}}_t$ be the working moments immediately before storage. The working moments are computed from the previously decoded states as

$$\tilde{\mathbf{m}}_t = \beta_1 Q(\tilde{\mathbf{m}}_{t-1}) + (1-\beta_1)\mathbf{g}_t, \qquad \tilde{\mathbf{v}}_t = \beta_2 Q(\tilde{\mathbf{v}}_{t-1}) + (1-\beta_2)\mathbf{g}_t^2.$$

The parameter update uses the working moments $\tilde{\mathbf{m}}_t$ and $\tilde{\mathbf{v}}_t$ before their quantized representations are stored for the next step.

2.2 RELATED WORK

Low-precision optimizer states. Block-wise dynamic quantization enabled practical 8-bit Adam-state storage (Dettmers et al., 2022), and Li et al. (2023) extended this approach to 4-bit states while identifying the second-moment zero-point failure. Subsequent work has explored paired geometric encodings (Tian et al., 2025), logarithmic-coordinate stochastic rounding and moment-specific precision (Xu et al., 2025), and spatio-temporal precision allocation (Liu et al., 2026). COAT combines FP8 optimizer-state storage with low-precision activation training (Xi et al., 2025); FlashOptim applies moment-specific companding before 8-bit quantization, including a square-root transform for the second moment (Gonzalez Ortiz et al., 2026); and Full-Stack FP4 uses transformed NVFP4 representations for AdamW states within a broader low-precision training system (Ding et al., 2026).

Dynamics of quantized optimizer states. Recent studies examine how quantization errors feed back through the recurrent exponential moving averages. SOLO analyzes signal swamping in unsigned states and variance amplification in signed states (Xu et al., 2025). Topollai & Choromanska (2026) study state staleness and periodic resets, while BReD targets persistent drift caused by biased recurrent rounding (Zhan et al., 2026). Tang et al. (2026) establish convergence guarantees for adaptive optimizers under floating-point quantization.

[1] Formal definitions and details are given in Appendix A.2.

Complementary memory-efficient optimizers. Optimizer memory can also be reduced by modifying the structure of the stored statistics or the update rule. Adafactor factorizes second-moment estimates (Shazeer & Stern, 2018), SM3 shares structured adaptive accumulators (Anil et al., 2019), and Adam-mini shares adaptive learning rates within parameter blocks (Zhang et al., 2025); Q-Adam-mini combines the latter structure with 8-bit quantization (Han et al., 2025). Gefen shares second-moment estimates and quantizes the first moment using a learned codebook (Benedek et al., 2026), while GaLore and APOLLO maintain optimizer information in lower-dimensional auxiliary spaces (Zhao et al., 2024; Zhu et al., 2025). Low-bit state representations have also been developed for other preconditioners, including 4-bit Shampoo (Wang et al., 2024) and quantized Muon (Gupta et al., 2025).

3 Second-Moment Quantization

Near zero, second-moment quantization presents two distinct distortions. Selecting zero can produce an excessively large subsequent preconditioner, whereas excluding zero imposes a positive floor that can suppress large preconditioner entries. We study these cases separately. ZIP-SR retains zero and changes the rounding coordinate while holding the decoded levels fixed. ZE-EDEN follows the established zero-excluding approach (Li et al., 2023; Xu et al., 2025; Topollai & Choromanska, 2026) and calibrates the decoded block scale to mitigate the distortion introduced by its positive floor.

3.1 ZIP-SR: Zero-Inclusive Preconditioner-space Stochastic Rounding

To isolate the effect of the rounding configuration from that of the codebook, we fix the representable levels. We first compare next-step preconditioner bias and derive a rule-independent criterion for how one stored value affects the next-step state and preconditioner. Then, we specialize this criterion to round-to-nearest (RTN) and SR in both state- and preconditioner-space before giving a scalar-quadratic construction that illustrates their different optimization dynamics.

Under the update-before-storage convention in Section 2, quantizing the working second moment $\tilde{v}_t$ affects the next update, after entering the second-moment recurrence together with g_{t+1}^2. For $v \geq 0$, define the current step and next-step preconditioner maps by

$$h_t(v) := \frac{1}{\sqrt{v/(1-\beta_2^t)} + \epsilon}, \qquad r_{t+1}(v) := h_{t+1}\big(\beta_2 v + (1-\beta_2) g_{t+1}^2\big).$$

We compare storing $Q(\tilde{v}_t)$ with storing the unquantized working value $\tilde{v}_t$ through the mean state and preconditioner errors

$$\Delta_v := \mathbb{E}[Q(\tilde{v}_t)] - \tilde{v}_t, \qquad \Delta_r := \mathbb{E}[r_{t+1}(Q(\tilde{v}_t))] - r_{t+1}(\tilde{v}_t).$$

All expectations are conditional on the realized values of $\tilde{v}_t$ and g_{t+1} and are taken only over the step-t rounding randomness.

State-space versus preconditioner-space rounding. For adjacent levels $0 \leq a < b$ in a non-negative decoded codebook $\mathcal{C}_v$, following the notation in Appendix A.2, state-space rounding uses the coordinate $\phi(v) = v$ and preconditioner-space rounding uses $\phi(v) = h_t(v)$. We use h_t as the rounding coordinate at storage time while r_{t+1} is used only for analysis because it depends on the next gradient g_{t+1}. For $\rho \in \{\mathrm{RTN}, \mathrm{SR}\}$, we write State-$\rho$ for $Q_{\mathrm{id},\rho,\mathcal{C}_v}$ and Update-ρ for $Q_{h_t,\rho,\mathcal{C}_v}$. RTN selects the endpoint whose value in the chosen coordinate is closest to $\phi(v)$. For $v \in (a,b)$, SR selects the lower level a with probability

$$\mathbb{P}(Q_{\phi,\mathrm{SR},\mathcal{C}_v}(v) = a) = \begin{cases} \frac{b-v}{b-a}, & \text{if } \phi = \mathrm{id}, \\ \frac{h_t(v)-h_t(b)}{h_t(a)-h_t(b)}, & \text{if } \phi = h_t. \end{cases}$$

Thus State-SR preserves the mean state value, whereas Update-SR preserves the mean of the current step preconditioner, i.e., $\mathbb{E}[h_t(Q_{h_t,\mathrm{SR},\mathcal{C}_v}(\tilde{v}_t))] = h_t(\tilde{v}_t)$. The next proposition compares their means after one step of moment recurrence. Proofs are given in Appendix D.1.

Proposition 1 (Next-step preconditioner bias) *Fix $t \geq 1$ and $\beta_2 \in (0,1)$.*
(a) (Arbitrary gradient and cell) *For every $\epsilon > 0$, $g_{t+1} \in \mathbb{R}$, and $0 \leq a < \tilde{v}_t < b$, let Q_{U} and Q_{S} denote Update-SR and State-SR between a and b. Then*

$$r_{t+1}(b) < \mathbb{E}[r_{t+1}(Q_{\mathrm{U}}(\tilde{v}_t))] \leq r_{t+1}(\tilde{v}_t) \leq \mathbb{E}[r_{t+1}(Q_{\mathrm{S}}(\tilde{v}_t))].$$

(b) (Zero-rounding criterion) *Fix $b > \tilde{v}_t > 0$ independent of ϵ. As $\epsilon \downarrow 0$ with $|g_{t+1}| = \mathcal{O}(\epsilon)$, any (possibly randomized) quantizer Q with $Q(\tilde{v}_t) \in \{0, b\}$ has bounded preconditioner error Δ_r if and only if $\mathbb{P}(Q(\tilde{v}_t) = 0) = \mathcal{O}(\epsilon)$. If $\mathbb{P}(Q(\tilde{v}_t) = 0) = \Theta(1)$, then $\Delta_r \to +\infty$ at rate $\Theta(\epsilon^{-1})$.*

Part (a) also shows how retaining zero in the second-moment codebook can reduce the preconditioner bias caused by a positive second-moment floor. For $0 < \tilde{v}_t < b$, a zero-excluding quantizer with decoded minimum b stores b, giving a next-step preconditioner $r_{t+1}(b) < r_{t+1}(\tilde{v}_t)$. Compare this with Update-SR between 0 and the same positive level b. Denoting their conditional mean preconditioner errors by Δ_r^{NZ} and Δ_r^{U}, respectively, part (a) gives

$$\Delta_r^{\mathrm{NZ}} < \Delta_r^{\mathrm{U}} \leq 0.$$

Thus Update-SR strictly reduces the magnitude of the bias while keeping the conditional mean preconditioner at or below its unquantized value. This comparison holds at a matched decoded floor for every $\epsilon > 0$ and every next gradient g_{t+1}.

The zero-adjacent cell. Let $\mathcal{C}_{0b} := \{0, b\}$ denote the decoded zero-adjacent codebook. Under the assumptions of Proposition 1(b), State-SR is unbiased in v but rounds to zero with probability $1 - \tilde{v}_t/b = \Theta(1)$, so by Proposition 1(b), its next-step preconditioner error diverges as $\Theta(\epsilon^{-1})$; State-RTN behaves the same throughout the lower half of the cell, where it deterministically rounds to zero. By contrast, Update-SR with $a = 0$ rounds to zero with probability

$$\mathbb{P}(Q_{h_t,\mathrm{SR},\mathcal{C}_{0b}}(\tilde{v}_t) = 0 \,|\, \tilde{v}_t) = \frac{\epsilon\big(1 - \sqrt{\tilde{v}_t/b}\big)}{\sqrt{\tilde{v}_t/(1-\beta_2^t)} + \epsilon} = \Theta(\epsilon). \tag{1}$$

Hence Update-SR has bounded mean preconditioner error Δ_r. Update-RTN selects zero only within a $\Theta(\epsilon^2)$ neighborhood of the origin, so every fixed $\tilde{v}_t > 0$ eventually rounds to b for sufficiently small ϵ. We summarize the results in Table 1 with details in Appendix D.1.

Table 1: Conditional mean errors for fixed $0 < \tilde{v}_t < b/2$ as $\epsilon \downarrow 0$, with $|g_{t+1}| = \mathcal{O}(\epsilon)$.

Rule	$\mathbb{P}(Q(\tilde{v}_t) = 0 \mid \tilde{v}_t)$	**State error Δ_v**	**Precond. error Δ_r**
State-RTN	1	$-\tilde{v}_t$	$+\Theta(\epsilon^{-1})$
State-SR	$1 - \tilde{v}_t/b$	0	$+\Theta(\epsilon^{-1})$
Update-RTN	0	$b - \tilde{v}_t$	$-\Theta(1)$
Update-SR	$\Theta(\epsilon)$	$b - \tilde{v}_t - \Theta(\epsilon)$	$-\Theta(1)$

Thus, preconditioner-space rounding suppresses the probability of storing zero at the cost of upward state bias. The next proposition examines how this tradeoff can affect optimization on a scalar quadratic. Part (a) compares complete deterministic trajectories: State-RTN approaches a nonzero two-cycle, whereas Update-RTN converges geometrically to the minimizer. Part (b) demonstrates that, in the near-optimum regime, State-SR becomes locally *expansive* while Update-SR remains locally *contractive*. Proofs are provided in Appendix D.2.

Proposition 2 (Rounding space separation on a scalar quadratic) *Fix $b, \mu > 0$. Consider Adam on $\mathcal{L}(w) = \mu w^2/2$ with $\beta_1 = 0$, $\beta_2 \in (1/2, 1)$, and second-moment codebook $\mathcal{C}_{0b} = \{0, b\}$.*
(a) (RTN trajectory) *Fix $\epsilon > 0$ and omit bias correction in Adam. Let $\tau_\epsilon = b\epsilon^2/(\sqrt{b} + 2\epsilon)^2$. For any constant stepsize satisfying $2(\sqrt{\tau_\epsilon} + \epsilon)/\mu < \eta < 2(\sqrt{b/2} + \epsilon)/\mu$, there is an open interval $I \subset (0, \infty)$ such that, from every common initialization $w_0 \in I$ and $\tilde{v}_0 = 0$, State-RTN converges to a stable nonzero two-cycle, whereas Update-RTN converges geometrically to $w = 0$.*
(b) (SR one-step shift) *Fix $t \geq 1$ and consider a family of realized step-t configurations satisfying $\tilde{v}_t \to 0$, $g_{t+1}^2 = \Theta(\tilde{v}_t)$, $\epsilon = o(\tilde{v}_t)$, and $\eta_{t+1}\mu r_{t+1}(0) \to \alpha > 2$. Taking expectations only over the randomness of SR at step-t gives*

$$\mathbb{E}\left[\log \frac{|w_{t+1}|}{|w_t|} \,\middle|\, \tilde{v}_t, g_{t+1}\right] = \begin{cases} \log(\alpha - 1) + o(1), & \textit{State-SR} \\ -\Theta\big(\sqrt{\tilde{v}_t}\big), & \textit{Update-SR}. \end{cases}$$

Controlled comparison of rounding choices. With FP32 first moments and the same zero-inclusive Dyn4 second-moment codebook, State-RTN and State-SR develop large training-loss gaps, whereas both preconditioner-space rules avoid this deterioration (Figure 1). Thus, in this controlled run, changing the rounding space avoids the observed instability without removing zero; state-space stochastic rounding alone does not resolve it.

We further compare Update-RTN and Update-SR in single-seed GPT-style 834M pretraining with FP32, SDyn4, or NF4 first moments. Their final validation-loss differences, $L_{\mathrm{SR}} - L_{\mathrm{RTN}}$, are $(+1.08, +0.97, -0.10) \times 10^{-4}$, respectively, and do not identify a consistent endpoint winner. We select Update-SR because it tracks the matched 32-bit trajectory more closely during warmup in the FP32-first-moment control.

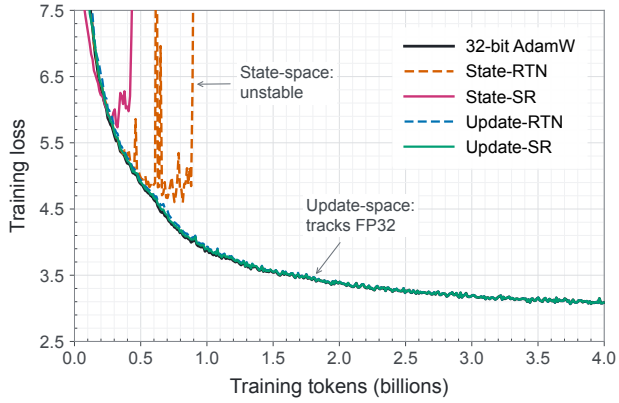

Figure 1: Rounding space controls training stability. Training loss over the first 4B tokens of GPT-style 834M pretraining, with FP32-$\mathbf{m}$ and zero-inclusive Dyn4-$\mathbf{v}$. State-RTN/SR become unstable and exceed the plotted range, while Update-RTN/SR closely track 32-bit AdamW.

3.2 ZE-EDEN: Zero-Excluding EDEN calibration

The second route excludes zero, preventing the zero-endpoint failure by construction but imposing a positive floor that inflates small second moments and suppresses the corresponding large preconditioner entries. For a second-moment block $\mathbf{x}$, we use the guarded EDEN-inspired calibration (Vargaftik et al., 2022) to mitigate this distortion:

$$\widetilde{\mathbf{x}} = c_\tau\, Q(\mathbf{x}) \quad \text{for} \quad \tau > 0,\ c_\tau = \frac{\|\mathbf{x}\|_2^2}{\max\{\langle \mathbf{x}, Q(\mathbf{x})\rangle, \tau\}}.$$

Here $\tau = 10^{-12}$ safeguards the denominator against zero or very small inner products, and c_τ is absorbed into the existing decoded block scale. The positive floor can overestimate small second-moment coordinates. When this makes $\langle \mathbf{x}, Q(\mathbf{x})\rangle > \|\mathbf{x}\|_2^2$, we have $c_\tau < 1$, so calibration lowers the decoded block scale and its positive floor. The resulting smaller second-moment values permit larger preconditioner entries, potentially restoring part of the tail suppressed by uncalibrated quantization.

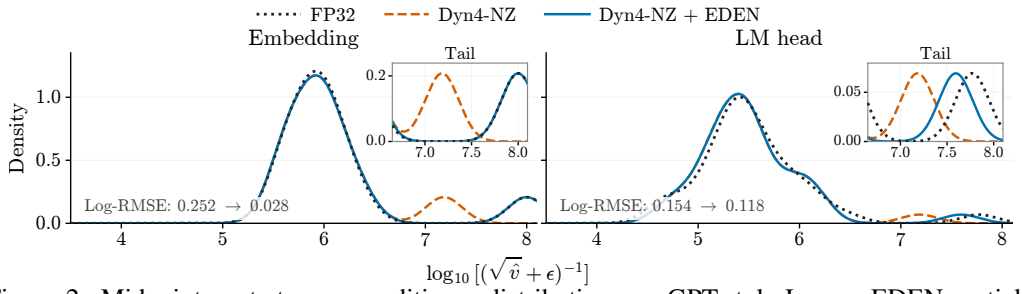

Figure 2: Midpoint next-step preconditioner distributions on GPT-style Large. EDEN partially restores the large-preconditioner tail suppressed by Dyn4-NZ in the embedding and LM-head states. Log-RMSE measures coordinate-wise error from FP32.

We denote zero-excluding dynamic 4-bit quantization by Dyn4-NZ. Figure 2 shows that its positive floor suppresses the large-preconditioner tail, whereas EDEN partially restores the FP32 distribution

for both the embedding and LM-head states. With FP32 first moments, EDEN lowers the final validation-loss gap of Dyn4-NZ from 0.0049 to 0.0038 (three-seed means). With NF4 or SDyn4 first moments, the uncalibrated gap rises to 0.0207 or 0.0312 (Table 8), motivating the first-moment study in Section 4; Section 5.1 measures EDEN's contribution in that setting. Additional diagnostics in Appendix C.3 extend the distributional comparison across training checkpoints and report one-step update errors for Lin4-NZ.

4 First-Moment Quantization

For a fixed preconditioner, the first moment enters the AdamW update linearly, making reconstruction error a useful initial criterion for codebook selection. However, its storage errors also recur through the exponential moving average, so small instantaneous error need not prevent persistent directional distortion. We therefore examine both codebook fit and the impact of the rounding rule on long-term training stability.

4.1 Codebook choice

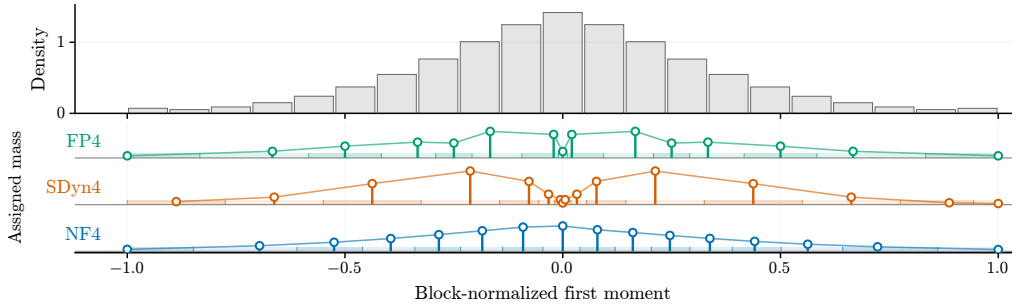

Figure 3: Empirical distribution of block-normalized FP32 first-moment values from the layer-12 MLP up-projection at step 15,908 of GPT-style Large pretraining, together with the probability mass assigned to each reconstruction level by FP4, SDyn4, and NF4 under RTN.

TorchAO 4-bit AdamW uses SDyn4 for the first moment. We revisit this choice using the block-normalized first-moment distribution in Figure 3. In this snapshot, FP4 and SDyn4 allocate several levels to narrow cells near zero, each receiving little probability mass, whereas NF4 spreads its levels across the high-density region. Under the same block-scaling rule, the mean squared reconstruction errors are 1.739×10^{-3}, 2.495×10^{-3}, and 1.056×10^{-3} for FP4, SDyn4, and NF4, respectively.

Complete single-seed pretraining of the GPT-style 834M model, with RTN for the first moment throughout, confirms this ordering. The final gaps for SDyn4, FP4, and NF4 are 0.0289, 0.0272, and 0.0183 under ZE-EDEN and 0.0239, 0.0215, and 0.0140 under ZIP-SR. Section 5.1 quantifies the NF4 change with three seeds.

4.2 Rounding rule for the first moment

We first use RTN to quantize the first moment $\mathbf{m}_t$ with the NF4 codebook. This choice remains stable in our smaller GPT-style pretraining experiments, where we observe no comparable late-stage degradation. In GPT-style 2.7B pretraining, however, the NF4-RTN trajectory departs sharply from 32-bit AdamW during the learning-rate cooldown, whereas NF4-SR remains stable (Figure 4a).

To identify the source of this instability, we conduct matched checkpoint interventions that vary the quantization of the LM-head and non-head first moments independently. These interventions localize the failure to deterministic rounding of the LM-head first moment: applying NF4-RTN only to this state reproduces the instability, while replacing RTN with SR restores the stable trajectory. The complete intervention design and results are reported in Appendix C.4. Layer-specific sensitivity to optimizer-state quantization has also been reported by Han et al. (2025), who use stochastic rounding for embedding-layer momentum to address weight-norm explosion.

Having localized the failure, we examine the directional structure of the LM-head quantization error. For a nonzero first-moment block $\mathbf{m}$, define its radial bias as

$$b_{\mathrm{rad}}(\mathbf{m}) = \langle Q(\mathbf{m}) - \mathbf{m}, \mathbf{m} \rangle / \|\mathbf{m}\|_2^2.$$

A negative value indicates contraction along the direction of $\mathbf{m}$, whereas a positive value indicates radial expansion. Tracking this statistic across iterations reveals whether the quantization error consistently contracts or expands the first moment along its current direction. Without saturation, coordinate-wise state-space SR has zero expected radial bias conditional on the block and scale. The trajectory diagnostic therefore examines whether systematic radial contraction occurs in the realized run.

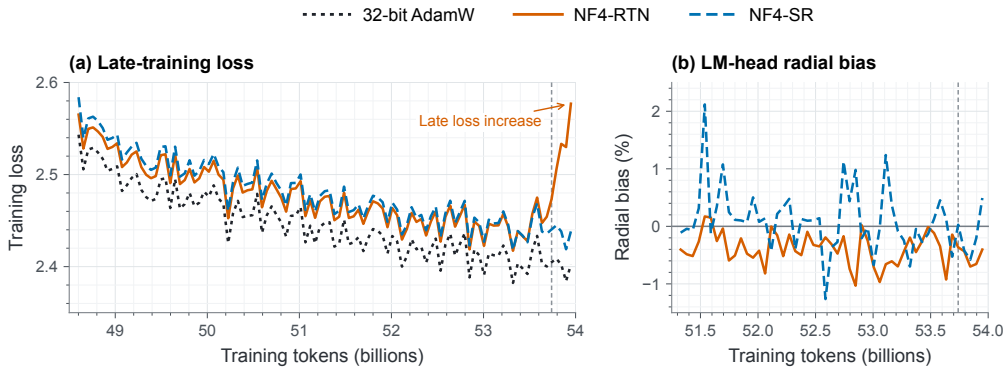

Figure 4: Late-training instability and LM-head directional bias in GPT-style 2.7B pretraining. (a) Cooldown training loss averaged in 100-update bins; NF4-SR uses SR on the LM-head first moment. (b) Radial bias shown every 100 updates in matched continuations, with FP32 non-head first moments and Dyn4/Update-SR second moments. The vertical dashed line marks the first global gradient-clipping event of the NF4-RTN continuation.

NF4-RTN produces a predominantly negative radial bias in the LM-head first moment, whereas the bias under NF4-SR remains centered near zero (Figure 4b). Since the quantized first moment is repeatedly fed back into its recurrence, the persistent inward bias under RTN can accumulate across iterations, providing a possible explanation for the subsequent loss instability. Additional diagnostics of the matched continuations, including LM-head gradient growth and comparisons with magnitude-based quantization errors, are provided in Appendix C.4. Guided by this diagnosis, our final methods use NF4-RTN for the first moment by default and switch only the LM-head first moment to NF4-SR during the final 10% of pretraining.

5 Experiments

We now evaluate the complete optimizer configurations in Table 2. The preceding controlled comparisons examine rounding space, EDEN calibration, and first-moment quantization separately; Section 5.1 first combines them into a single attribution at 834M, and the remaining experiments evaluate the complete configurations in pretraining and supervised fine-tuning.

Both methods quantize the first moment with NF4, using RTN by default and switching the LM-head first moment to SR during the final 10% of training. These shared choices are combined with two second-moment designs: ZIP-SR applies Update-SR with the zero-inclusive Dyn4 codebook, whereas ZE-EDEN applies State-RTN with the zero-excluding Dyn4-NZ codebook and EDEN calibration. Full implementation and training details are given in Appendix B.

Persistent optimizer-state storage. All three 4-bit configurations use packed 4-bit storage for eligible moment tensors and therefore have essentially the same persistent optimizer-state footprint. The codebook and rounding changes require no additional per-parameter state, and ZE-EDEN absorbs its calibration factor into the existing block scale. With block size 128 and one FP32 scale per block, storing both AdamW moments requires 1.0625 B/parameter on eligible tensors, compared with 8 B/parameter for FP32 moment storage, corresponding to a $7.53\times$ compression. At the scale of our experiments, this corresponds to approximately 19.2 GB less persistent moment-state storage for the 2.7B-parameter pretraining model and roughly 21–56 GB less for the 3B–8B full-parameter SFT models. To keep the storage comparison controlled, all 4-bit methods use the same tensor-eligibility criterion as the TorchAO baseline. In our model configurations, only normalization-layer moment tensors fall outside this criterion and therefore remain in FP32.

Table 2: Optimizer-state configurations evaluated in the end-to-end experiments. ZIP-SR and ZE-EDEN share the NF4 first-moment design established in Section 4, but use the two different second-moment quantization routes developed in Section 3. The superscript $*$ indicates a phase-dependent rule: the LM-head first-moment uses NF4-RTN before the final 10% of training and switches to NF4-SR only during that phase.

Configuration	First moment		Second moment		
	Non-LM-head	LM-head	Format	Rule	EDEN
32-bit	FP32	FP32	FP32	—	×
TorchAO 4-bit	SDyn4-RTN	SDyn4-RTN	Lin4-NZ	State-RTN	×
ZIP-SR 4-bit	NF4-RTN	NF4-RTN/SR*	Dyn4	Update-SR	×
ZE-EDEN 4-bit	NF4-RTN	NF4-RTN/SR*	Dyn4-NZ	State-RTN	✓

5.1 Component attribution

The second-moment format change alone does not explain the final gains in Table 3. Replacing Lin4-NZ with Dyn4-NZ for $\mathbf{v}$ increases the validation-loss gap (step 1). Switching $\mathbf{m}$ from SDyn4 to NF4 gives the largest stepwise reduction (step 2). Both branches from step 2 then reduce the gap further: EDEN does so while keeping Dyn4-NZ and State-RTN fixed (step 3a), and the Dyn4 Update-SR configuration yields a larger reduction (step 3b). The step 3b comparison changes both the second-moment format and rounding rule and uses a single run, so it does not isolate the effect of Update-SR. Relative to TorchAO, the gaps after steps 3a and 3b are 28.2% and 43.2% lower, respectively.

The sample SDs for steps 1, 2, and 3a are 8×10^{-4}, 1.0×10^{-3}, and 1.5×10^{-3}, respectively. Quantized first moments use RTN throughout, including the LM head. Appendix C.1 reports the full grid across first-moment settings and both orders of changing the first- and second-moment choices.

Table 3: Component attribution on GPT-style 834M. Gaps are relative to matched 32-bit AdamW. Steps 1, 2, and 3a are means over three random seeds; steps 0 and 3b are single runs. Steps 3a and 3b both compare with step 2. Positive Gain denotes a smaller gap; derived values use unrounded gaps.

Step	Setting	First moment	Second moment			Gap	vs. parent	
		Format	Format	Rule	EDEN		Δ	Gain
0	TorchAO 4-bit	SDyn4	Lin4-NZ	State-RTN	×	0.0246	—	—
1	Dyn4-NZ control	SDyn4	Dyn4-NZ	State-RTN	×	0.0312	+0.0066	−26.9%
2	NF4 first moment	NF4	Dyn4-NZ	State-RTN	×	0.0207	−0.0105	33.7%
3a	ZE-EDEN	NF4	Dyn4-NZ	State-RTN	✓	0.0177	−0.0031	14.7%
3b	ZIP-SR	NF4	Dyn4	Update-SR	×	0.0140	−0.0067	32.5%

5.2 Pretraining

We pretrain GPT-style models at 162M, 405M, 1.4B, and 2.7B parameters and Llama-style models at 130M, 350M, and 1.1B with three paired seeds. Table 4 reports absolute 32-bit AdamW validation loss and paired 4-bit gaps. Both methods reduce TorchAO's mean gap at every model size. A single-seed GPT-style 834M experiment with cosine decay also finds lower gaps for ZE-EDEN and ZIP-SR than for TorchAO (Appendix C.2).

On GPT-style 2.7B, TorchAO undergoes a sharp late-cooldown divergence. At the validation checkpoint at 95% of training tokens, before the terminal regression its mean paired gap was $+0.0754\,(0.0034)$, compared with $+0.0471\,(0.0018)$ for ZE-EDEN and $+0.0286\,(0.0024)$ for ZIP-SR, where parentheses report sample SDs. Relative to TorchAO, these correspond to gap reductions of 37.6% for ZE-EDEN and 62.1% for ZIP-SR. Hence, both proposed optimizers were already closer to 32-bit AdamW before TorchAO's terminal regression. Figure 7, in the appendix, shows the GPT-style 2.7B validation loss trajectories. The largest reported percentage reduction is therefore the 70.1% reduction at 1.4B; we do not compute a percentage reduction from the divergent 2.7B endpoint.

Table 4: Pretraining validation loss: absolute 32-bit loss and mean paired 4-bit gaps to matched 32-bit runs (sample SD); bold marks the smallest gap per row. The final column reports the larger gap reduction achieved by ZE-EDEN or ZIP-SR. All rows use three seeds. The percentage reduction for GPT-style 2.7B is omitted because the †-marked TorchAO endpoint diverges late in cooldown.

Family	Size	32-bit loss	4-bit gap to 32-bit AdamW			Best reduction vs. TorchAO (%)
			TorchAO	ZE-EDEN	ZIP-SR	
GPT-style	162M	3.1335 (0.0016)	+0.0290 (0.0028)	+0.0200 (0.0021)	**+0.0195** (0.0020)	32.8
	405M	2.8197 (0.0008)	+0.0254 (0.0015)	**+0.0167** (0.0007)	+0.0180 (0.0005)	34.3
	1.4B	2.5186 (0.0012)	+0.0521 (0.0048)	+0.0184 (0.0010)	**+0.0156** (0.0026)	70.1
	2.7B	2.4044 (0.0019)	+1.5435† (0.1437)	+0.0535 (0.0024)	**+0.0343** (0.0027)	—
Llama-style	130M	2.6974 (0.0027)	+0.0205 (0.0015)	+0.0177 (0.0009)	**+0.0171** (0.0003)	16.6
	350M	2.4054 (0.0014)	+0.0290 (0.0014)	**+0.0135** (0.0013)	**+0.0135** (0.0007)	53.4
	1.1B	2.1467 (0.0002)	+0.0486 (0.0029)	+0.0226 (0.0026)	**+0.0198** (0.0003)	59.3

5.3 Supervised Fine-tuning

We run full-parameter Supervised Fine-tuning (SFT) on Tulu-3 (Lambert et al., 2025) using Qwen3-8B-Base (Yang et al., 2025) and Llama-3.2-3B (Meta, 2024) over five paired seeds, evaluating MMLU, GSM8K, HumanEval, and IFEval (Table 5). Both proposed methods attain lower mean validation loss than TorchAO 4-bit AdamW. Relative to matched 32-bit AdamW, ZE-EDEN's mean paired validation-loss gap is essentially zero on Qwen3-8B and slightly negative on Llama-3.2-3B. Downstream scores remain close to those of 32-bit AdamW, without a consistent advantage across tasks for either proposed method. ZE-EDEN attains the lower validation loss on both SFT models, whereas ZIP-SR generally achieves smaller gaps in the pretraining comparisons.

Table 5: All SFT runs use AdamW. The 32-bit values are five-seed means, while the 4-bit values are mean paired gaps relative to matched 32-bit runs; parentheses report sample SDs.

Model	Configuration	Tulu val. loss ↓	MMLU ↑	GSM8K ↑	HumanEval ↑	IFEval strict ↑
Qwen3-8B-Base	32-bit	0.55135 (0.00010)	78.5 (0.2)	82.8 (0.6)	72.6 (0.4)	49.5 (2.1)
	TorchAO 4-bit	+0.00060 (0.00009)	0.0 (0.1)	+0.2 (1.1)	−0.1 (1.2)	+0.6 (1.8)
	ZE-EDEN 4-bit	−0.00003 (0.00014)	+0.1 (0.1)	+0.2 (0.9)	+1.1 (1.3)	−0.1 (2.4)
	ZIP-SR 4-bit	+0.00042 (0.00016)	+0.1 (0.2)	+0.6 (0.6)	+0.5 (2.0)	−0.1 (1.9)
Llama-3.2-3B	32-bit	0.73255 (0.00044)	52.0 (2.0)	34.6 (1.2)	33.4 (1.8)	37.4 (1.9)
	TorchAO 4-bit	+0.00175 (0.00012)	+1.4 (1.5)	+0.4 (1.6)	+0.6 (1.4)	+0.4 (1.2)
	ZE-EDEN 4-bit	−0.00035 (0.00016)	−0.4 (1.7)	−0.2 (1.5)	0.0 (1.5)	+0.4 (2.4)
	ZIP-SR 4-bit	+0.00089 (0.00012)	+0.4 (1.1)	−0.4 (0.8)	0.0 (1.4)	+0.2 (1.3)

6 Conclusion

This work shows that optimizer-state quantization should account for how stored values affect optimization, rather than state-space reconstruction alone. For the second-moment, this perspective motivates Update-SR for the zero-inclusive route, while EDEN calibration separately improves the zero-excluding route by mitigating distortion in the induced preconditioner distribution. Together with NF4 and SR for the LM-head first-moment, these designs improve over TorchAO across pretraining and fine-tuning settings.

Limitations and future work. Our theory compares conditional one-step preconditioner biases and optimization dynamics on a toy model, but does not establish general end-to-end convergence for quantized AdamW; extending these results to broader settings remains future work. Targeted LM-head SR alleviates the observed late-training regression, but the cause of the directional first-moment bias and its possible relationship to learning-rate decay remain unresolved. Further directions include extending our methodology below 4-bit optimizer-state quantization and investigating whether it applies to other adaptive optimizers.

Ethics Statement

This work involves no human subjects or newly collected data. All datasets and base models are publicly available and used under their licenses. Lower optimizer memory reduces the hardware needed for full-parameter training, which applies to beneficial and harmful uses alike. The authors declare no competing interests.

Reproducibility Statement

Appendices A.1 and A.2 define all codebooks and rounding rules, and Table 2 lists the evaluated configurations of 4-bit AdamW. Appendix B reports model architectures, hyperparameters, learning-rate sweeps, data splits, seeds, and hardware.

AI Use Statement

Building on the authors' initial ideas to compare the optimizer state and preconditioner errors of different rounding rules, we used ChatGPT (GPT-6 and GPT-5.6 Sol) to help formulate the statements of Propositions 1 and 2 for theoretical justification and polish the proof in multiple rounds of discussion. We used ChatGPT and Claude to give feedback on the experimental design and to polish the writing and organization. The authors verified all AI-assisted proofs and text and take full responsibility for the manuscript.

REFERENCES

Rohan Anil, Vineet Gupta, Tomer Koren, and Yoram Singer. Memory-efficient adaptive optimization. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Nadav Benedek, Tomer Koren, and Ohad Fried. Gefen: Optimized stochastic optimizer. *arXiv preprint arXiv:2606.13894*, 2026. URL https://arxiv.org/abs/2606.13894.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. In *International Conference on Learning Representations*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 10088–10115. Curran Associates, Inc., 2023. doi: 10.52202/075280-0441.

Siyu Ding, Mingchuan Ma, Jiabo Tong, Xingrun Xing, Ziming Wang, and Guoqi Li. Full-stack FP4: Stable LLM pretraining with quantized projections, optimizers, and attention. *arXiv preprint arXiv:2607.04422*, 2026. doi: 10.48550/arXiv.2607.04422. URL https://arxiv.org/abs/2607.04422.

Jose Javier Gonzalez Ortiz, Abhay Gupta, Christopher Rinard, and Davis Blalock. FlashOptim: Optimizers for memory-efficient training. *arXiv preprint arXiv:2602.23349*, 2026. URL https://arxiv.org/abs/2602.23349.

Aman Gupta, Rafael Celente, Abhishek Shivanna, D. T. Braithwaite, Gregory Dexter, Shao Tang, Hiroto Udagawa, Daniel Silva, Rohan Ramanath, and S. Sathiya Keerthi. Effective quantization of Muon optimizer states. *arXiv preprint arXiv:2509.23106*, 2025. URL https://arxiv.org/abs/2509.23106.

Yizhou Han, Chaohao Yang, Xingjian Wang, Congliang Chen, and Ruoyu Sun. Q-Adam-mini: Memory-efficient 8-bit quantized optimizer for large language model training. In *ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models (ICML Workshop)*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022. doi: 10.52202/068431-2176.

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

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Conference on Language Modeling*, 2025.

Bingrui Li, Jianfei Chen, and Jun Zhu. Memory efficient optimizers with 4-bit states. In *Advances in Neural Information Processing Systems*, volume 36, pp. 15136–15171. Curran Associates, Inc., 2023. doi: 10.52202/075280-0666.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. PyTorch Distributed: Experiences on accelerating data parallel training. *Proceedings of the VLDB Endowment*, 13(12):3005–3018, 2020. doi: 10.14778/3415478.3415530.

Minglu Liu, Cunchen Hu, Liangliang Xu, Fengming Tang, Ruijia Wang, and Fu Yu. STQuant: Spatio-temporal adaptive framework for optimizer quantization in large multimodal model training. *arXiv preprint arXiv:2604.06836*, 2026. URL https://arxiv.org/abs/2604.06836.

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

Meta. Llama-3.2-3B. Model card, 2024. URL https://huggingface.co/meta-llama/Llama-3.2-3B.

Andrew Or, Apurva Jain, Daniel Vega-Myhre, Jesse Cai, Charles David Hernandez, Zhenrui Zheng, Driss Guessous, Vasiliy Kuznetsov, Christian Puhrsch, Mark Saroufim, Supriya Rao, Thien Tran, and Aleksandar Samardžić. TorchAO: PyTorch-native training-to-serving model optimization. *arXiv preprint arXiv:2507.16099*, 2025. URL https://arxiv.org/abs/2507.16099.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language-models.pdf.

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. doi: 10.1109/SC41405.2020.00024.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4596–4604. PMLR, 2018. URL https://proceedings.mlr.press/v80/shazeer18a.html.

Xuan Tang, Jichu Li, and Difan Zou. A convergence analysis of adaptive optimizers under floating-point quantization. In *International Conference on Learning Representations*, 2026.

Zhen Tian, Xin Zhao, and Ji-Rong Wen. Irrational complex rotations empower low-bit optimizers. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Kristi Topollai and Anna Choromanska. Understanding quantization of optimizer states in LLM pre-training: Dynamics of state staleness and effectiveness of state resets. *arXiv preprint arXiv:2603.16731*, 2026. doi: 10.48550/arXiv.2603.16731. URL https://arxiv.org/abs/2603.16731.

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

Shay Vargaftik, Ran Ben-Basat, Amit Portnoy, Gal Mendelson, Yaniv Ben-Itzhak, and Michael Mitzenmacher. EDEN: Communication-efficient and robust distributed mean estimation for federated learning. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 21984–22014. PMLR, 2022.

Sike Wang, Pan Zhou, Jia Li, and Hua Huang. 4-bit Shampoo for memory-efficient network training. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-4033.

Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. COAT: Compressing optimizer states and activations for memory-efficient FP8 training. In *International Conference on Learning Representations*, 2025.

Cong Xu, Wenbin Liang, Mo Yu, Anan Liu, Ke-Yue Zhang, Shunli Wang, Lizhuang Ma, Jianyong Wang, Jun Wang, and Wei Zhang. Pushing the limits of low-bit optimizers: A focus on EMA dynamics. *arXiv preprint arXiv:2505.00347*, 2025. doi: 10.48550/arXiv.2505.00347. URL https://arxiv.org/abs/2505.00347.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

Heshen Zhan, Youhan Huang, Yunke Peng, Yao Wang, Linghui Kong, Ziwei Zhu, Qingyu Han, Yaoyuan Wang, Congliang Chen, and Ruoyu Sun. BReD: Stabilizing quantized EMA dynamics for memory-efficient large-scale training. In *Fourth Workshop on High-dimensional Learning Dynamics (HiLD at ICML)*, 2026. URL https://openreview.net/forum?id=ATXhZIy5PM.

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

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61121–61143. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhao24s.html.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

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. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/hash/437bc4ccafd3fc6d4289bd10940be42b-Abstract-Conference.html.

A Quantization Background: Codebooks and Rounding Rules

A.1 4-Bit Codebooks

All named format codebooks introduced below contain normalized reconstruction values.

Definition 3 (Decoded 4-bit quantization) *Let $(c_0, \dots, c_{15})$ be a normalized 4-bit lookup table. For a block with scale $s_B > 0$, let*

$$\mathcal{C} := \{s_B c_j : 0 \le j < 16\}$$

be its set of distinct decoded reconstruction values. A scalar encoder selects a stored code $q(x) \in \{0, \dots, 15\}$, possibly at random. The selected code decodes to

$$Q(x) = s_B c_{q(x)} \in \mathcal{C}.$$

Adjacency refers to adjacent values in $\mathcal{C}$ under the usual order on $\mathbb{R}$, not adjacent code indices.

Dynamic codebook. Li et al. (2023) use the dynamic-exponent (DE) mapping, which allocates more reconstruction levels near zero. We briefly review its construction before giving the resulting 4-bit codebooks. For $F \ge 0$, define uniformly spaced boundaries on $[0.1, 1]$ by

$$p_{F,j} := 0.1 + \frac{0.9j}{2^F}, \qquad j = 0, \dots, 2^F,$$

and let

$$r_{F,k} := \frac{p_{F,k} + p_{F,k+1}}{2}, \qquad k = 0, \dots, 2^F - 1,$$

denote the corresponding interval midpoints.

For an unsigned b-bit DE mapping, a nonexceptional code contains E leading zeros, one indicator bit, and $F = b - E - 1$ fraction bits. Its reconstruction value is $10^{-E} r_{F,k}$. Including the two exceptional reconstruction values 0 and 1, the resulting normalized codebook is

$$\mathcal{C}_{\mathrm{Dyn},b} = \{0, 1\} \cup \bigcup_{E=0}^{b-2} \left\{10^{-E} r_{b-E-1,k} : 0 \le k < 2^{b-E-1}\right\}.$$

For signed DE, one bit is additionally reserved for the sign, leaving $F = b - E - 2$ fraction bits. The normalized codebook is

$$\mathcal{C}_{\mathrm{SDyn},b} = \{0, 1\} \cup \bigcup_{E=0}^{b-2} \left\{\pm 10^{-E} r_{b-E-2,k} : 0 \le k < 2^{b-E-2}\right\}.$$

The exceptional assignment includes 1 but not -1; consequently, the signed DE codebook is not exactly sign symmetric.

We refer to the unsigned and signed 4-bit instances as Dyn4 and SDyn4, respectively. Substituting $b = 4$ gives $\mathcal{C}_{\mathrm{Dyn4}}$ and $\mathcal{C}_{\mathrm{SDyn4}}$.

FP4 (E2M1). The 4-bit E2M1 floating-point format represents $\{0, 0.5, 1, 1.5, 2, 3, 4, 6\}$. Normalizing by its largest magnitude gives the distinct normalized reconstruction levels

$$\mathcal{C}_{\mathrm{FP4}} = \left\{0, \pm\frac{1}{12}, \pm\frac{1}{6}, \pm\frac{1}{4}, \pm\frac{1}{3}, \pm\frac{1}{2}, \pm\frac{2}{3}, \pm 1\right\}.$$

The positive- and negative-zero codes decode to the same reconstruction, so the 16 codes produce 15 distinct values.

Zero-excluding codebook. Li et al. (2023) consider two distinct ways of excluding zero from the unsigned second-moment codebook. The first, called DE-0 in their paper, directly removes zero from the DE mapping. We refer to its 4-bit instance as Dyn4-NZ: $\mathcal{C}_{\mathrm{Dyn4\text{-}NZ}} = \mathcal{C}_{\mathrm{Dyn4}} \setminus \{0\}$. Its smallest normalized reconstruction value is 0.00325, and the mapping has 15 distinct normalized reconstruction values. Li et al. (2023) subsequently propose a different zero-excluding linear mapping,

$$\mathcal{T}_{\mathrm{Lin},b}(i) = \frac{i+1}{2^b}, \qquad i = 0, \dots, 2^b - 1.$$

At four bits, this gives the 16 distinct positive levels

$$\mathcal{C}_{\text{Lin4-NZ}} = \left\{ \frac{1}{16}, \frac{2}{16}, \ldots, \frac{16}{16} \right\}.$$

This linear mapping, rather than DE-0, is the zero-excluding second-moment mapping proposed and used in their final 4-bit AdamW configuration. In this work, Dyn4-NZ and Lin4-NZ therefore denote two different positive codebooks.

NormalFloat. NF4 is constructed from quantiles of a standard normal distribution, with an exact zero included and the resulting values normalized to $[-1, 1]$ (Dettmers et al., 2023). Its normalized reconstruction levels are approximately

$$\begin{aligned}\mathcal{C}_{\text{NF4}} = \{ &-1,\ -0.696193,\ -0.525073,\ -0.394917,\ -0.284441,\ -0.184773,\ -0.091050,\\ &0,\ 0.079580,\ 0.160930,\ 0.246112,\ 0.337915,\ 0.440710,\ 0.562617,\ 0.722957,\ 1\}.\end{aligned}$$

The displayed values are rounded for readability; our implementation uses the corresponding floating-point lookup table.

A.2 Round-to-Nearest and Stochastic Rounding

Fix a decoded codebook $\mathcal{C}$ as in Definition 3. A rounding coordinate is a scalar map

$$\phi : [\min \mathcal{C}, \max \mathcal{C}] \to \mathbb{R}$$

that is either strictly increasing or strictly decreasing. For an input $x \in \mathbb{R}$ strictly between adjacent values $a < b$ in $\mathcal{C}$, define the lower-endpoint interpolation weight

$$p_\phi(x; a, b) := \frac{|\phi(x) - \phi(b)|}{|\phi(a) - \phi(b)|} \in [0, 1].$$

Definition 4 (Round-to-nearest) *Round-to-nearest writes*

$$Q_{\phi,\text{RTN},\mathcal{C}}(x) = \begin{cases} a, & \text{if } p_\phi(x; a, b) > \frac{1}{2}, \\ b, & \text{if } p_\phi(x; a, b) < \frac{1}{2}. \end{cases}$$

If $p_\phi(x; a, b) = 1/2$, RTN resolves the tie using a predetermined rule.

Definition 5 (Stochastic rounding) *Stochastic rounding writes*

$$Q_{\phi,\text{SR},\mathcal{C}}(x) = \begin{cases} a, & \text{with probability } p_\phi(x; a, b), \\ b, & \text{with probability } 1 - p_\phi(x; a, b). \end{cases}$$

For one fresh endpoint draw,

$$\mathbb{E}[\phi(Q_{\phi,\text{SR},\mathcal{C}}(x))] = \phi(x),$$

where x, $\mathcal{C}$, and ϕ are fixed.

Inputs below or above the represented interval saturate to its lower or upper decoded boundary, respectively. State-RTN and State-SR use the strictly increasing coordinate $\phi(v) = v$. Update-RTN and Update-SR use the strictly decreasing coordinate $\phi(v) = h_t(v)$ and require a nonnegative second-moment codebook.

B EXPERIMENTAL DETAILS

B.1 SHARED SETTINGS

All pretraining and supervised fine-tuning (SFT) runs use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, denominator stabilizer $\epsilon = 10^{-8}$, and global gradient-norm clipping at 1.0. Both tasks use BF16 mixed-precision computation with FP32 master parameters. Task-specific weight decay, learning-rate schedules, batch sizes, and training budgets are specified below.

The 32-bit AdamW baseline stores both moments in FP32. All 4-bit configurations use blocks of 128 elements, packed 4-bit indices, and one FP32 scale per block. A moment tensor is eligible for quantization if it contains at least 4096 elements and its size is divisible by 128; ineligible moment tensors remain in FP32. The eligibility criterion is identical across the 4-bit methods. The parameter update uses the working moments before their quantized representations are stored for the next step, as described in Section 2. ZE-EDEN uses the denominator safeguard $\tau = 10^{-12}$ and absorbs its calibration factor into the block scale.

The main pretraining and SFT comparisons use the configurations in Table 2. For ZIP-SR and ZE-EDEN, the first moment uses NF4-RTN by default, with only the LM-head first moment switching to NF4-SR after 90% of training. In the GPT-style 834M component comparisons, quantized first moments use RTN throughout training. The LM-head rule is therefore inactive in Tables 3 and 8. We report absolute mean metrics for 32-bit AdamW and mean paired differences for the 4-bit methods. Each paired difference subtracts the matched 32-bit result for the same seed; parentheses denote sample standard deviations. Single-seed component results are identified separately.

B.2 PRETRAINING

Models and data. The architectures are listed in Table 6. The main comparison in Table 4 uses GPT-style models at 162M, 405M, 1.4B, and 2.7B parameters and Llama-style models at 130M, 350M, and 1.1B parameters. The GPT-style 834M model is used for the component studies. All models are pretrained on FineWeb-Edu (Penedo et al., 2024). GPT-style models use the GPT-2 tokenizer (Radford et al., 2019) and context length 2048; Llama-style models use the Llama 2 tokenizer (Touvron et al., 2023) and context length 1024.

Table 6: Architectures used for pretraining and component studies. Size denotes the rounded parameter count; L, d_{model}, heads, n_{kv}, and FFN denote the layer count, hidden width, attention heads, key/value heads, and feed-forward width. GPT-style models use RoPE/GELU with untied embeddings. Llama-style models use RoPE/SwiGLU/RMSNorm, including an output norm, with $n_{kv} = 4$.

Family	Size	L	d_{model}	Heads	n_{kv}	FFN
GPT-style	162M	12	768	12	—	3072
	405M	24	1024	16	—	4096
	834M	24	1536	16	—	6144
	1.4B	24	2048	16	—	8192
	2.7B	32	2560	32	—	10240
Llama-style	130M	12	768	12	4	2304
	350M	22	1024	16	4	3392
	1.1B	22	2048	32	4	5632

Training details. Each model is trained with a budget of 20 tokens per parameter (Hoffmann et al., 2022). The effective batch size is 256 sequences across eight GPUs. GPT-style 405M, 1.4B, and 2.7B and Llama-style 1.1B use a per-GPU micro-batch of 16 with two gradient-accumulation steps. The remaining configurations use a per-GPU micro-batch of 32 without gradient accumulation.

Pretraining uses weight decay $\lambda = 0.1$ and a warmup-stable-decay (WSD) schedule. The learning rate increases linearly to its peak during the first 10% of updates, remains constant for the next 80%, and decays to zero during the final 10%. During cooldown, the multiplier is $1 - \sqrt{u}$, where $u \in [0, 1]$ is the normalized progress through cooldown. Table 7 reports the FP32 AdamW learning-rate sweeps used to select the model-specific peak rates. For GPT-style 2.7B, we fix the peak rate at 1.5×10^{-4}

across methods and seeds without a sweep. For each model size, the selected learning rate and the remaining training hyperparameters are shared across optimizer configurations.

Table 7: Peak learning rates for pretraining. Each sweep uses 20 training tokens per parameter. The final column reports the minimum observed validation loss at the precision retained in the experiment logs. The GPT-style 2.7B rate is fixed without a sweep.

Family	Size	Candidate peak LRs	Selected LR	Val. loss
GPT-style	162M	$\{2, 1, 0.6, 0.3\} \times 10^{-3}$	1×10^{-3}	3.135
	405M	$\{1, 0.6, 0.3, 0.15\} \times 10^{-3}$	6×10^{-4}	2.818
	834M	$\{0.6, 0.3, 0.25, 0.15\} \times 10^{-3}$	6×10^{-4}	2.641
	1.4B	$\{0.3, 0.2, 0.15, 0.1\} \times 10^{-3}$	2×10^{-4}	2.517
	2.7B	Not swept	1.5×10^{-4}	—
Llama-style	130M	$\{4, 3, 2, 1, 0.6, 0.3\} \times 10^{-3}$	2×10^{-3}	2.698
	350M	$\{2, 1.5, 1, 0.6, 0.3, 0.15\} \times 10^{-3}$	1×10^{-3}	2.406
	1.1B	$\{0.6, 0.3, 0.2, 0.1\} \times 10^{-3}$	3×10^{-4}	2.148

Each run uses one node with eight H200 GPUs and activation checkpointing and PyTorch DistributedDataParallel (DDP) (Li et al., 2020), except that GPT-style 2.7B pretraining runs use one node with eight B200 GPUs.

Evaluation. Each pretokenized cache has a fixed validation split of $50{,}000$ sequences. The main pretraining comparisons use three paired seeds and the reporting convention in Appendix B.1.

B.3 Supervised fine-tuning

Models and data. We perform full-parameter SFT of Qwen3-8B-Base (Yang et al., 2025) and Llama-3.2-3B (Meta, 2024) on the Tulu-3 SFT dataset (Lambert et al., 2025). We first reserve 2% of the Tulu-3 mixture for validation using a deterministic split with seed 42. From the remaining pool, we select a fixed subset of $50{,}000$ training examples, shared across methods and training seeds. Examples are tokenized with the corresponding model tokenizer, truncated to 2048 tokens, and processed without sequence packing. The loss is computed only on the final assistant response; preceding conversation and padding tokens are masked.

Training details. Each run trains for one epoch with an effective batch size of 64 examples. A per-device batch size of one across eight GPUs, with eight gradient-accumulation steps, gives 782 optimizer updates, including the final partial batch.

Both models use zero weight decay and a peak learning rate of 10^{-5}. This rate is selected by validation loss in a seed-42 FP32 AdamW sweep and then shared across optimizer variants. The schedule uses linear warmup during the first 3% of updates, followed by cosine decay to zero.

Each run uses eight H200 GPUs and activation checkpointing. Qwen3-8B uses DeepSpeed ZeRO-2 (Rajbhandari et al., 2020) for every optimizer, whereas Llama-3.2-3B uses PyTorch DistributedDataParallel (DDP) (Li et al., 2020).

Evaluation. We evaluate the final model from each run over five paired training seeds, 42–46. We report Tulu validation loss, 5-shot MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and prompt-level strict IFEval (Zhou et al., 2023). MMLU is macro-averaged over its 57 subjects. For IFEval, Qwen3-8B uses its corresponding chat template with thinking disabled, whereas Llama-3.2-3B uses the plain template used in training. MMLU, GSM8K, and HumanEval use the standard task prompts without an applied chat template. Results follow the paired reporting convention in Appendix B.1.

C Additional Diagnostics

We first report the GPT-style 834M component grid underlying Table 3, followed by a cosine learning-rate schedule ablation study. Next, we examine EDEN calibration with Lin4-NZ and compare one-step update errors for Lin4-NZ and Dyn4-NZ across training checkpoints. Finally, we examine GPT-style 2.7B training dynamics through additional rounding diagnostics, and validation trajectories for the complete optimizer configurations.

C.1 Component attribution on GPT-style 834M pretraining

Table 8 expands the Dyn4-NZ and Dyn4/Update-SR branches of Table 3 across first-moment settings, including the EDEN ablation with FP32 first moments. The TorchAO Lin4-NZ reference appears as step 0 of the main table; Lin4-NZ was not crossed with the other first-moment settings in this grid. The FP4 runs are reported in Section 4, the state-space rules, which become unstable with zero-inclusive Dyn4, in Figure 1, and the Update-RTN comparison in Section 3.1.

Order sensitivity. The grid lets us change the first-moment codebook before or after modifying the second moment. For EDEN, the order changes the observed reduction in the mean gap: adding EDEN under SDyn4 lowers it by 0.39×10^{-3}, whereas adding it after switching to NF4 lowers it by 3.05×10^{-3}. Replacing Dyn4-NZ State-RTN with Dyn4 Update-SR lowers the displayed gap by about 7×10^{-3} under either first-moment codebook. The Dyn4/Update-SR endpoints are single runs compared with three-seed Dyn4-NZ means. This replacement also changes the second-moment codebook, the rounding coordinate, and RTN to SR together.

Table 8: GPT-style 834M component grid. Validation-loss gaps to 32-bit AdamW. Rows vary the first moment (RTN throughout); columns vary the second moment. Entries with a sample SD in parentheses are three-seed means; the others are single runs.

	Dyn4-NZ (State-RTN)		Dyn4
m	**Raw**	**+ EDEN**	**Update-SR**
FP32	0.0049 (0.0001)	0.0038 (0.0003)	—
SDyn4	0.0312 (0.0008)	0.0308 (0.0027)	0.0239
NF4	0.0207 (0.0010)	0.0177 (0.0015)	0.0140

C.2 Sensitivity to the learning-rate schedule

The main pretraining experiments use warmup-stable-decay (WSD). We also evaluate the complete optimizer configurations on GPT-style 834M with cosine learning-rate decay. The ZE-EDEN and ZIP-SR runs include the LM-head first-moment switch to NF4-SR during the final 10% of training.

Table 9: GPT-style 834M pretraining with cosine learning-rate decay. Final validation loss at update 31,810 for seed 42. Gaps to the 32-bit AdamW run are shown directly.

Configuration	**Validation loss**	**Gap to 32-bit**
32-bit AdamW	2.6274	—
TorchAO 4-bit	2.6554	0.0280
ZE-EDEN	2.6421	0.0147
ZIP-SR	2.6385	0.0111

With cosine decay, both recipes have a smaller gap to 32-bit AdamW than TorchAO in this seed, as they do in the WSD pretraining experiments. This run tests whether the ordering persists under cosine decay; the WSD component runs at 834M use RTN for the first moment throughout and therefore do not provide matched schedule comparisons for the complete recipes.

C.3 EDEN calibration for zero-excluding second-moment quantization

The main text examines EDEN calibration with Dyn4-NZ for the second moment. Here we extend the distributional analysis to Lin4-NZ, the zero-excluding linear codebook described in Appendix A.1, and compare one-step update errors for both codebooks across training.

Using the GPT-style 834M model pretraining, we replay one update from 32-bit AdamW checkpoints saved at 25%, 50%, and 75% of training. Within each codebook, the raw and EDEN-calibrated branches share stored indices, the recorded post-clipping gradient, and FP32 fallback coordinates; only the effective decoded block scales differ.

Preconditioner distributions. Figure 5 compares next-step preconditioner distributions for FP32, Lin4-NZ, and Lin4-NZ with EDEN calibration, using embedding and LM-head states. At all three sampled checkpoints, uncalibrated Lin4-NZ produces floor-induced peaks and suppresses the large-preconditioner tail. EDEN calibration partially restores this tail, although residual distributional distortion remains.

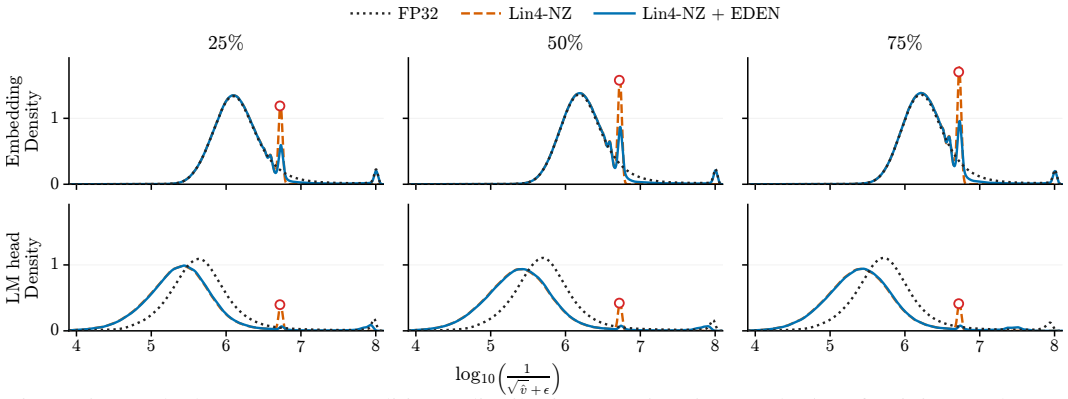

Figure 5: Matched next-step preconditioner distributions at 25%, 50%, and 75% of training. Columns show checkpoints and rows show embedding and LM-head states. Curves compare FP32, Lin4-NZ, and Lin4-NZ with EDEN; open circles mark the Lin4-NZ floor atom.

One-step update errors. At the first resumed update, we measure the relative full-vector adaptive-update error $\|u_{4\text{bit}} - u_{32}\|_2/\|u_{32}\|_2$, where u excludes the common learning-rate multiplier and weight decay. At the 25%, 50%, and 75% checkpoints, EDEN reduces the Lin4-NZ relative error from 30.99% to 30.58%, from 28.80% to 28.34%, and from 27.87% to 27.38%, respectively. For Dyn4-NZ, the corresponding changes are 15.45% to 15.41%, 12.83% to 12.77%, and 11.83% to 11.79%. Thus, EDEN yields small but consistent reductions at these three checkpoints, while Dyn4-NZ has lower error than Lin4-NZ under this one-step replay metric.

C.4 GPT-STYLE 2.7B TRAINING DYNAMICS AND LM-HEAD DIAGNOSTICS

Checkpoint interventions. Following the comparison in Figure 4, we examine matched checkpoint interventions to isolate the effect of LM-head first-moment rounding. To distinguish insufficient four-bit capacity from the deterministic rounding path itself, we branch matched continuations from the same accumulated step-97,831 checkpoint, before the sharp loss increase. The continuations preserve the model weights, numerical optimizer state, data order, and learning-rate schedule; only the stated future first-moment storage rule changes. Every branch keeps the second-moment fixed to Dyn4 Update-SR with an FP32 current read, thereby isolating the LM-head first-moment intervention.

Table 10: Localization of the late-training instability in GPT-style 2.7B pretraining. Matched continuations vary the quantization of the LM-head and non-head first moments while holding the remaining training configuration fixed.

LM-head $\mathbf{m}_t$	Non-head $\mathbf{m}_t$	Final train loss	Final val. loss
FP32	NF4-RTN	2.392	2.440
NF4-RTN	FP32	2.529	2.579
NF4-SR	FP32	2.391	2.440

Keeping the LM-head first moment in FP32 prevents the loss increase even when the remaining first moments use NF4-RTN. Conversely, applying NF4-RTN only to the LM-head first moment reproduces the instability. Replacing RTN with SR for this state restores the original training trajectory, showing that the failure is associated with deterministic LM-head rounding rather than 4-bit LM-head quantization itself.

Gradient growth. The main text identifies a persistent inward radial bias in the LM-head first moment under NF4-RTN. We next examine the corresponding gradient dynamics of the matched continuations. Their LM-head gradient norms remain comparable initially but separate rapidly near the end of cooldown (Figure 6). The LM-head pre-clipping gradient norm under NF4-RTN reaches 2.044, whereas NF4-SR remains below 0.388. This separation connects the LM-head quantization behavior examined in the main text with the gradient growth that accompanies the subsequent loss increase.

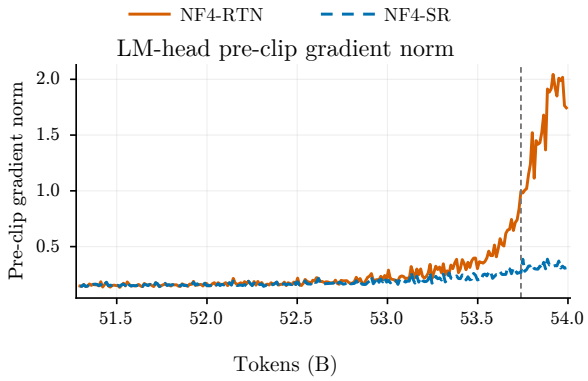

Figure 6: LM-head gradient growth in the matched continuations. The solid orange curve denotes NF4-RTN and the dashed blue curve denotes NF4-SR. The vertical dashed line marks step 102,500 (53.74B tokens), at which the NF4-RTN continuation first triggers global gradient clipping.

Magnitude-based diagnostics. The stability of NF4-SR is not explained by smaller instantaneous quantization errors. We report in Table 11 that NF4-SR has larger errors under each conventional diagnostic, despite avoiding both the persistent inward bias and the subsequent gradient growth.

Table 11: Magnitude-based diagnostics for the matched LM-head continuations. Values are aggregated over 91 recorded evaluations; lower values indicate smaller instantaneous quantization error.

Diagnostic	NF4-RTN	NF4-SR
Relative RMS quantization error (%)	9.21	14.75
Sampled shadow-state error (%)	20.95	32.18
Nonzero-to-zero fraction (%)	9.16	12.08

NF4-RTN performs better under all three magnitude-based diagnostics but nevertheless develops the late-training instability. These aggregate error measures therefore do not distinguish the stable continuation from the unstable one. The persistent radial bias and subsequent gradient growth provide complementary diagnostics of the observed failure.

End-to-end validation dynamics. Figure 7 shows the validation loss curve of the complete optimizer configurations in Table 2. The methods remain closely aligned before cooldown. During cooldown, ZE-EDEN and ZIP-SR remain close to 32-bit AdamW, while TorchAO undergoes a sharp terminal regression.

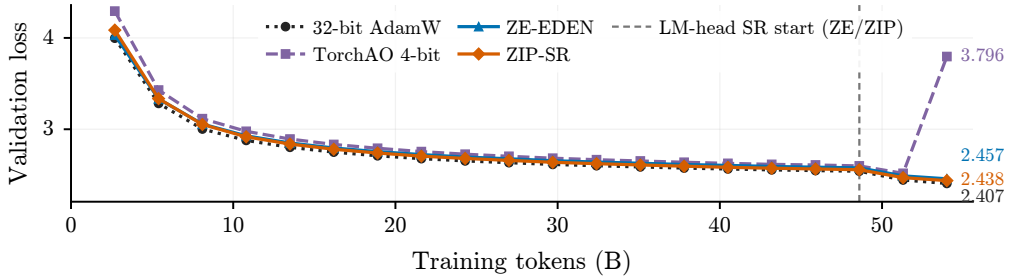

Figure 7: Validation loss during GPT-style 2.7B pretraining. The dashed line marks the switch from NF4-RTN to NF4-SR for the LM-head first moment at 90% of training in ZE-EDEN and ZIP-SR. TorchAO exhibits a sharp regression, whereas ZE-EDEN and ZIP-SR remain close to 32-bit AdamW.

D Omitted Proofs in Section 3.1

D.1 One-Step State and Preconditioner Errors

Proof of Proposition 1. (a) Fix the quantities in part (a), and write $v = \tilde{v}_t$ and $r = r_{t+1}$. The map r is convex on $[0, \infty)$: the scalar map $u \mapsto (\sqrt{u} + \epsilon)^{-1}$ is convex there, and its argument in r is affine. Since $\mathbb{E}[Q_{\mathrm{S}}(v)] = v$, Jensen's inequality gives $\mathbb{E}[r(Q_{\mathrm{S}}(v))] \geq r(v)$.

For Update-SR, we prove concavity of $F := r \circ h_t^{-1}$ by establishing convexity of its increasing inverse. Define

$$\kappa := \sqrt{\frac{\beta_2(1-\beta_2^t)}{1-\beta_2^{t+1}}} \in (0,1), \qquad c := \frac{(1-\beta_2)g_{t+1}^2}{1-\beta_2^{t+1}} \geq 0.$$

Since r and h_t are strictly decreasing, F is strictly increasing. For $0 < z \leq (\epsilon + \sqrt{c})^{-1}$, its inverse is

$$F^{-1}(z) = \frac{\kappa z}{D(z)}, \qquad D(z) := \sqrt{(1-\epsilon z)^2 - cz^2} + \kappa\epsilon z.$$

The square-root term is concave by direct computation, so D is concave. Moreover, $D(z) > 0$ and $D(z) \leq 1 - \epsilon(1-\kappa)z \leq D(0) = 1$. By concavity, this bound also implies that D is nonincreasing. The ratio $\kappa z / D(z)$ is therefore convex by the univariate convex-ratio rule (Boyd & Vandenberghe, 2004, Exercise 3.32(c)). Thus F^{-1} is convex and increasing, and F is concave.

Using $\mathbb{E}[h_t(Q_{\mathrm{U}}(v))] = h_t(v)$, Jensen's inequality gives

$$\mathbb{E}[r(Q_{\mathrm{U}}(v))] = \mathbb{E}[F(h_t(Q_{\mathrm{U}}(v)))] \leq F(\mathbb{E}[h_t(Q_{\mathrm{U}}(v))]) = r(v).$$

Finally, r is strictly decreasing and $Q_{\mathrm{U}}(v)$ assigns positive probability to $a < b$, so $\mathbb{E}[r(Q_{\mathrm{U}}(v))] > r(b)$. Together these inequalities prove part (a).

(b) Under the gradient assumption, $r(0) = \Theta(\epsilon^{-1})$, whereas $r(v)$ and $r(b)$ remain bounded. For $p_0 = \mathbb{P}(Q(v) = 0) \in [0,1]$,

$$\Delta_r = p_0 r(0) + (1-p_0)r(b) - r(v) = p_0 r(0) + \mathcal{O}(1),$$

with a remainder uniform in p_0. Since $\epsilon r(0)$ is bounded above and away from zero, Δ_r is bounded if and only if $p_0 = \mathcal{O}(\epsilon)$. If $p_0 = \Theta(1)$, the same identity gives $\Delta_r = +\Theta(\epsilon^{-1})$. ■

Comparison with a matched positive floor. For $0 < v = \tilde{v}_t < b$, a zero-excluding quantizer with decoded minimum b stores b deterministically. Write $r = r_{t+1}$ and let Q_{U} denote Update-SR between 0 and b. The corresponding mean errors $\Delta_r^{\mathrm{NZ}} = r(b) - r(v)$ and $\Delta_r^{\mathrm{U}} = \mathbb{E}[r(Q_{\mathrm{U}}(v))] - r(v)$ satisfy $\Delta_r^{\mathrm{NZ}} < \Delta_r^{\mathrm{U}} \leq 0$ by part (a). With $p_0 = \mathbb{P}(Q_{\mathrm{U}}(v) = 0) \in (0,1)$, the reduction in absolute mean error is

$$|\Delta_r^{\mathrm{NZ}}| - |\Delta_r^{\mathrm{U}}| = p_0\big[r(0) - r(b)\big] > 0.$$

This comparison holds for every $\epsilon > 0$ and every next gradient at the same decoded floor b.

Derivation of Table 1. Fix $0 < v = \tilde{v}_t < b/2$ and assume $|g_{t+1}| = \mathcal{O}(\epsilon)$. Write $r = r_{t+1}$ and define

$$A = \sqrt{1-\beta_2^t}\left(\frac{1}{\sqrt{v}} - \frac{1}{\sqrt{b}}\right) > 0, \qquad \kappa = \sqrt{\frac{\beta_2(1-\beta_2^t)}{1-\beta_2^{t+1}}} \in (0,1).$$

Let $\gamma_\epsilon = (|g_{t+1}|/\epsilon)\sqrt{(1-\beta_2)/(1-\beta_2^{t+1})}$, which is bounded by the gradient assumption. Evaluating r at zero and expanding at the fixed positive inputs v, b yields

$$r(0) = \frac{1}{\epsilon(1+\gamma_\epsilon)}, \qquad r(v) - r(b) = \frac{A}{\kappa} + \mathcal{O}(\epsilon).$$

For a zero-write probability p_0, the errors are $\Delta_v = b - v - bp_0$ and $\Delta_r = r(b) - r(v) + p_0[r(0) - r(b)]$. State-RTN has $p_0 = 1$ and hence $\Delta_v = -v$. State-SR has $p_0 = 1 - v/b$ and hence $\Delta_v = 0$. Part (b) gives $\Delta_r = +\Theta(\epsilon^{-1})$ for both rules.

The Update-RTN boundary solves $h_t(v) = (h_t(0) + h_t(b))/2$, giving

$$v_{\mathrm{th}} = \frac{b\epsilon^2}{\left(\sqrt{b/(1-\beta_2^t)} + 2\epsilon\right)^2} = (1-\beta_2^t)\epsilon^2 + \mathcal{O}(\epsilon^3). \tag{2}$$

For sufficiently small ϵ, the fixed input $v > 0$ exceeds this boundary, so $p_0 = 0$, $\Delta_v = b - v$, and $\Delta_r = -A/\kappa + \mathcal{O}(\epsilon) = -\Theta(1)$.

For Update-SR, (1) gives $p_0 = A\epsilon + \mathcal{O}(\epsilon^2)$. Substituting into the error identities yields

$$\Delta_v = b - v - bA\epsilon + \mathcal{O}(\epsilon^2),$$
$$\Delta_r = -A\left(\frac{1}{\kappa} - \frac{1}{1+\gamma_\epsilon}\right) + \mathcal{O}(\epsilon).$$

Since $1/\kappa - 1/(1+\gamma_\epsilon) \geq 1/\kappa - 1 > 0$ and γ_ϵ is bounded, $\Delta_r = -\Theta(1)$. This establishes the remaining row of the table.

D.2 Rounding-Space Separation on a Scalar Quadratic

Proof of Proposition 2(a). Fix a stepsize η and set $c = \sqrt{1-\beta_2}$. For either RTN rule, let Q denote its quantizer and $(\tilde{v}_k, w_k)$ its working moments and iterates. In this part, Update-RTN uses the coordinate $h(v) = (\sqrt{v} + \epsilon)^{-1}$. Starting from $\tilde{v}_0 = 0$, the recurrences are

$$\tilde{v}_k = \beta_2 Q(\tilde{v}_{k-1}) + (c\mu w_{k-1})^2, \qquad w_k = w_{k-1}\left(1 - \frac{\eta\mu}{\sqrt{\tilde{v}_k} + \epsilon}\right).$$

Define the candidate cycle amplitude by $w_\eta = (\eta\mu/2 - \epsilon)/(c\mu)$. The stepsize bounds give $\sqrt{\tau_\epsilon} < c\mu w_\eta < \sqrt{b/2}$. Choose $\delta > 0$ sufficiently small that the closed interval $J = [w_\eta - \delta, w_\eta + \delta] \subset (0, \infty)$ satisfies

$$\tau_\epsilon < (c\mu y)^2 < b/2, \qquad c\mu y + \epsilon < \eta\mu \quad \text{for all } y \in J,$$

and take $I = \operatorname{int} J$. Fix a common initialization $w_0 \in I$.

For State-RTN, suppose $Q(\tilde{v}_{k-1}) = 0$ and $y = |w_{k-1}| \in J$. Then $\tilde{v}_k = (c\mu y)^2 < b/2$, so the next write is again zero, and the update multiplier is negative. Since $c\mu w_\eta + \epsilon = \eta\mu/2$,

$$|w_k| - w_\eta = (y - w_\eta)\frac{\epsilon - c\mu y}{\epsilon + c\mu y}.$$

The constant $q = \max_{y\in J} |\epsilon - c\mu y|/(\epsilon + c\mu y)$ is less than one. Hence the amplitudes remain in J, and induction gives $\big||w_k| - w_\eta\big| \leq q^k |w_0 - w_\eta|$. The signs alternate, so $w_{2k} \to w_\eta$ and $w_{2k+1} \to -w_\eta$. This proves local attraction of the two-cycle.

For Update-RTN, the first working moment satisfies $\tilde{v}_1 = (c\mu w_0)^2 > \tau_\epsilon$, so the first write is b. Every subsequent write also equals b, since a preceding write of b gives $\tilde{v}_k \geq \beta_2 b > b/2 > \tau_\epsilon$. The two rules share the same first iterate w_1. Because $\eta\mu < 2(\sqrt{\beta_2 b} + \epsilon)$, the contraction factor

$$\rho := \max_{0\leq y\leq |w_1|} \left|1 - \frac{\eta\mu}{\sqrt{\beta_2 b + (c\mu y)^2} + \epsilon}\right|$$

is less than one by continuity on the compact interval. Induction yields $|w_k| \leq \rho^{k-1}|w_1|$ for $k \geq 1$, proving geometric convergence to zero from the same interval I. ∎

Remark before proof of Proposition 2(b). A zero write does not affect the current update, but it removes the historical second-moment term from the next recurrence. If the state and the next squared gradient shrink together into the zero-adjacent cell, the resulting zero-endpoint read grows while the positive-endpoint read remains bounded. A learning rate that decays with the state scale can therefore leave the zero branch expansive while making the positive branch weakly contractive. State-SR selects the zero branch with probability approaching one, whereas Update-SR makes it sufficiently rare for the positive-branch contraction to dominate.

Proof of Proposition 2(b). For the quadratic objective, $g_{t+1} = \mu w_t$, and $\beta_1 = 0$ makes the working first moment equal to this gradient. The first update influenced by a quantized value $z \in \{0, b\}$ is therefore

$$w_{t+1} = (1 - \eta_{t+1}\mu r_{t+1}(z))\, w_t.$$

For either rounding rule, let Δ_t denote the conditional logarithmic drift appearing in the proposition. The locally tracked regime and the definition of r_{t+1} give

$$r_{t+1}(0) = \Theta\big(\tilde{v}_t^{-1/2}\big), \qquad r_{t+1}(b) = \Theta(1), \qquad \frac{r_{t+1}(b)}{r_{t+1}(0)} = \Theta\big(\sqrt{\tilde{v}_t}\big).$$

The assumed scaling of the zero endpoint implies $\eta_{t+1} = \Theta(\sqrt{\tilde{v}_t}) \to 0$. Consequently,

$$\log|1 - \eta_{t+1}\mu r_{t+1}(0)| \longrightarrow \log(\alpha - 1),$$
$$\log|1 - \eta_{t+1}\mu r_{t+1}(b)| = -\Theta\big(\sqrt{\tilde{v}_t}\big).$$

State-SR selects zero with probability $1 - \tilde{v}_t/b$ and b with probability $\tilde{v}_t/b$. Therefore

$$\Delta_t = \left(1 - \frac{\tilde{v}_t}{b}\right) \log|1 - \eta_{t+1}\mu r_{t+1}(0)| + \frac{\tilde{v}_t}{b} \log|1 - \eta_{t+1}\mu r_{t+1}(b)|$$
$$\longrightarrow \log(\alpha - 1) > 0.$$

For Update-SR, the probability of selecting zero is

$$p_0 = \frac{\epsilon(1 - \sqrt{\tilde{v}_t/b})}{\sqrt{\tilde{v}_t/(1 - \beta_2^t)} + \epsilon} = o\big(\sqrt{\tilde{v}_t}\big),$$

where the final relation follows from $\epsilon = o(\tilde{v}_t)$. Hence

$$\Delta_t = p_0 \log|1 - \eta_{t+1}\mu r_{t+1}(0)| + (1 - p_0) \log|1 - \eta_{t+1}\mu r_{t+1}(b)| = -\Theta\big(\sqrt{\tilde{v}_t}\big),$$

which is negative for all sufficiently small $\tilde{v}_t$. ■