BReD: Stabilizing Quantized EMA Dynamics for Memory-Efficient Large-Scale Training

Heshen Zhan HESHENZHAN@LINK.CUHK.EDU.CN
The Chinese University of HongKong, Shenzhen

Youhan Huang YOUHANHUANG@LINK.CUHK.EDU.CN
The Chinese University of HongKong, Shenzhen

Yunke Peng PENGYUNKE@HUAWEI.COM
Huawei

Yao Wang WANGYAO123@HUAWEI.COM
Huawei

Linghui Kong KONGLINGHUI21@MAILS.UCAS.SC.CN
Huawei

Ziwei Zhu ZIWEIZHU@LINK.CUHK.EDU.CN
The Chinese University of HongKong, Shenzhen

Qingyu Han QINGYUHAN2@LINK.CUHK.EDU.CN
The Chinese University of HongKong, Shenzhen

Yaoyuan Wang WANGYAOYUAN1@HUAWEI.COM
Huawei

Congliang Chen CHENCONGLIANG@SLAI.EDU.CN
Shenzhen Loop Area Institute

Ruoyu Sun SUNRUOYU@CUHK.EDU.CN
The Chinese University of HongKong, Shenzhen

Abstract

Low-bit storage of optimizer states can reduce the memory footprint of large-scale language-model training, but for states updated by exponential moving averages (EMAs), it also changes the optimizer dynamics. At every step, the stored state is dequantized, updated, requantized, and written back, so quantization errors are recursively fed into future optimizer states. We analyze the quantized EMA recursion and identify a failure mode: biased quantization errors accumulate into persistent drift, error variance is amplified, and the aggregate state error scales with state dimensionality. We propose **BReD** (**B**lock **Re**play **D**ithering), which adapts classical subtractive dithering to optimizer-state storage using deterministic seed replay and block-wise shared dither values. BReD leaves optimizer updates and hyperparameters unchanged, stores no auxiliary random tensors, and reduces random-number generation from $O(d)$ to $O(d/B)$ for state dimension d and block size B. In controlled 120M pretraining experiments across five optimizers, 4-bit BReD remains stable where nearest and stochastic rounding diverge or degrade. In AdamW and Muon pretraining at 1.1B and 3.4B parameters, 4-bit BReD closely tracks full-precision optimizer-state baselines: validation perplexity shifts by at most 0.3, average zero-shot accuracy changes by at most 0.5 points, and optimizer-state memory is reduced by up to 86.0% in the evaluated settings. These results suggest that stable low-bit EMA-state storage is a practical option for memory-constrained scaling.

© H. Zhan, Y. Huang, Y. Peng, Y. Wang, L. Kong, Z. Zhu, Q. Han, Y. Wang, C. Chen & R. Sun.

Keywords: Optimizer-state quantization; quantized EMA dynamics; memory-efficient training; subtractive dithering.

1. Introduction

Many optimizers used in large-scale model training maintain auxiliary states [6, 10, 11, 14, 27, 31], such as momentum buffers, first- and second-moment estimates, or preconditioner statistics. These states are updated throughout training and can add one or more parameter-sized tensors to the optimizer memory footprint. Since this cost grows with model size, storing optimizer states in low-bit formats is a natural way to reduce the memory required for training.

For EMA-based optimizer states, however, low-bit storage introduces an additional issue. The state is not quantized once and then treated as a fixed compressed tensor. At each optimizer step, the stored value is dequantized, used in the optimizer update, updated by the EMA rule, quantized again, and written back. Thus, the quantization error introduced by the low-bit format is fed into the next EMA update. Over many steps, small per-step errors can become accumulated state error.

We study this effect through the error recursion induced by quantized EMA storage. The analysis shows that biased quantization errors produce persistent drift, while error variance is amplified by the EMA decay. These effects are most pronounced when the decay factor is close to one, as in common momentum and second-moment states. This suggests that aggressive low-bit EMA-state storage should avoid systematic bias while keeping the error variance controlled.

We propose **BReD** (**B**lock **Re**play **D**ithering). BReD uses subtractive dithering [19] to obtain unbiased rounding under a fixed quantization grid. To make this practical for large optimizer states, it regenerates dither values by deterministic seed replay instead of storing random tensors, and shares one dither value within each block to reduce random number generation cost. BReD leaves the optimizer update and hyperparameters unchanged.

We evaluate BReD in language-model pretraining with AdamW [11, 14], Muon [10], NAdamW [6], SOAP [27], and MARS [31]. In controlled 120M experiments, nearest and stochastic rounding [1, 26] diverge or substantially degrade under the same scaling rule and quantization grid, while 4-bit BReD remains stable. At 1.1B and 3.4B parameters, 4-bit BReD closely tracks full-precision optimizer-state baselines for AdamW and Muon, with validation perplexity shifts at most 0.3 and average zero-shot accuracy changes within 0.5 points. Across the evaluated settings, packed 4-bit BReD reduces optimizer-state memory by 35.9%–86.0%.

More related work is provided in Appendix A.

2. Quantized EMA Error Accumulation

Many optimizer states are updated by exponential moving averages. Let s_t denote a full-precision EMA state,

$$s_{t+1} = \beta s_t + (1-\beta)u_{t+1}, \qquad 0 < \beta < 1, \tag{1}$$

where u_{t+1} is the incoming update term. With low-bit storage, the optimizer reads a dequantized state $\hat{s}_t$, applies the same EMA update,

$$\tilde{s}_{t+1} = \beta \hat{s}_t + (1-\beta)u_{t+1}, \tag{2}$$

and then quantizes the updated state for storage. Define the accumulated state error and the one-step quantization error as

$$e_t := \hat{s}_t - s_t, \qquad \epsilon_{t+1} := \hat{s}_{t+1} - \tilde{s}_{t+1}. \tag{3}$$

A direct subtraction gives

$$e_{t+1} = \beta e_t + \epsilon_{t+1}. \tag{4}$$

Thus, the quantization error at one step is reused through future EMA updates, rather than appearing as an independent one-step perturbation. This model isolates the error caused by low-bit state storage; it is not a convergence result for the full optimizer, where changes in the stored state may also change the parameter trajectory and the future update terms u_t.

Unrolling Eq. (4) gives the steady-state effect of the one-step quantization error. If $\mathbb{E}[\epsilon_t] = b$, then

$$\mathbb{E}[e_t] = \beta^t \mathbb{E}[e_0] + \frac{1-\beta^t}{1-\beta} b \longrightarrow \frac{b}{1-\beta}. \tag{5}$$

Thus, a biased quantization error produces persistent bias in the stored EMA state. Under the temporal-independence assumptions stated in Appendix D, if $\epsilon_t = b + \xi_t$ with $\mathbb{E}[\xi_t] = 0$ and $\mathbb{E}\|\xi_t\|_2^2 = \sigma^2$, then

$$\lim_{t\to\infty} \mathbb{E}\|e_t\|_2^2 = \frac{\|b\|_2^2}{(1-\beta)^2} + \frac{\sigma^2}{1-\beta^2}. \tag{6}$$

For a state with d entries, write the per-coordinate squared bias and residual variance as $\mu^2 := \|b\|_2^2/d$ and $\tau^2 := \sigma^2/d$. Then Eq. (6) becomes

$$\lim_{t\to\infty} \mathbb{E}\|e_t\|_2^2 = \frac{d\mu^2}{(1-\beta)^2} + \frac{d\tau^2}{1-\beta^2}. \tag{7}$$

This makes the dependence on state dimension explicit: small per-coordinate bias or variance can produce a large accumulated error for large optimizer states, especially when β is close to one.

For a fixed scale and quantization grid, and away from clipping or saturation, the rounding rule determines the quantization error. Nearest rounding can introduce systematic bias. Stochastic rounding is locally unbiased but still leaves large variance. This motivates a rounding rule with small systematic bias and controlled variance, which we introduce in Section 3.

In-training diagnostic. We check whether this mechanism appears during language-model pre-training. In a LLaMA-120M AdamW run, we maintain measurement-only quantized copies of one second-moment state using nearest rounding (NR), stochastic rounding (SR), and subtractive dithering (SD). These copies do not affect training. For each copy, we record the accumulated state error e_t and the one-step quantization error ϵ_t. For AdamW's second-moment state, the EMA coefficient in Eq. (4) is $\beta = \beta_2$, so Eq. (5) predicts the steady mean error $\mathbb{E}[\epsilon_t]/(1-\beta_2)$ after the initial transient.

3. BReD: Block Replay Dithering

The analysis above suggests that low-bit EMA-state storage should avoid systematic quantization bias and keep the variance small. BReD applies subtractive dithering to EMA optimizer-state storage and regenerates the required dither values without storing auxiliary random tensors.

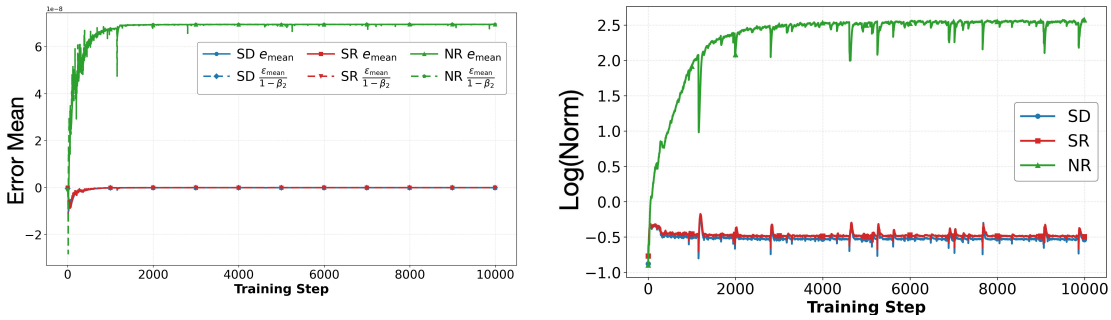

Figure 1: In-training diagnostic for quantized EMA-state storage during LLaMA-120M AdamW pretraining. Left: the measured mean state error follows $\mathbb{E}[\epsilon_t]/(1-\beta_2)$ after the transient. Right: relative state error $\log_{10}(\|e_t\|_F/\|s_t\|_F)$. NR accumulates drift and larger relative error, while SR and SD have near-zero mean error and smaller relative error in this diagnostic.

Local rounding rule. Given a quantization scaling rule and a quantization grid, and away from clipping or saturation, the local quantization error is determined by the rounding and reconstruction rule. Consider a scalar value x. Let $p_0(x) \le x \le p_1(x)$ be the neighboring grid points, and define

$$\Delta(x) := p_1(x) - p_0(x), \qquad \alpha(x) := \frac{x - p_0(x)}{\Delta(x)} \in [0,1]. \tag{8}$$

Stochastic rounding is unbiased and has conditional variance

$$\mathrm{Var}(\hat{x}_{\mathrm{SR}} - x \mid x) = \Delta(x)^2 \alpha(x)(1-\alpha(x)) \le \Delta(x)^2/4. \tag{9}$$

Definitions of nearest rounding and stochastic rounding are given in Appendix B.

Subtractive dithering draws $r \sim \mathrm{Unif}(0,1)$, rounds a dithered coordinate to a grid point, and reconstructs using the same r:

$$c_{\mathrm{SD}}(\alpha; r) := \mathbf{1}\{\alpha + r \ge 1\}, \qquad \hat{x}_{\mathrm{SD}} := p_0(x) + \Delta(x)\left(c_{\mathrm{SD}}(\alpha; r) - r + \frac{1}{2}\right). \tag{10}$$

Conditioned on x, this gives

$$\mathbb{E}[\hat{x}_{\mathrm{SD}} - x \mid x] = 0, \qquad \mathrm{Var}(\hat{x}_{\mathrm{SD}} - x \mid x) = \Delta(x)^2/12. \tag{11}$$

The advantage of SD is that the variance is independent of α, and its worst-case conditional variance is smaller: $\Delta(x)^2/12$ instead of $\Delta(x)^2/4$. SD does not dominate SR pointwise, since SR has a smaller variance near grid points. The proof is given in Appendix D.

Seed replay and block sharing. Subtractive dithering requires the same dither value during quantization and dequantization. Storing one random value per state entry would remove the memory savings. BReD instead regenerates dither values from deterministic keys. For an optimizer-state tensor with identifier $\mathrm{state_id}$, block index j, and stored step index t, BReD uses

$$r_j^{(t)} \leftarrow \mathrm{PRNG}(\mathrm{seed}, \mathrm{state_id}, t, j) \in (0,1). \tag{12}$$

(a) 120M rounding controls

Optimizer	NR	SR	BReD
AdamW	Diverged	Diverged	18.7
Muon	Degraded (31.0)	Degraded (21.7)	18.0
NAdamW	Diverged	Diverged	18.9
SOAP	Diverged	Diverged	18.1
MARS	Diverged	Diverged	18.9

(b) Scale check and optimizer-state memory

Scale	Opt.	ΔAvg.	ΔPPL	Opt. mem.
1.1B	AdamW	−0.1	+0.1	8.20→1.18GB (85.6%)
1.1B	Muon	−0.5	+0.3	4.59→1.47GB (68.0%)
3.4B	AdamW	−0.3	+0.1	25.53→3.58GB (86.0%)
3.4B	Muon	−0.5	+0.2	13.53→3.17GB (76.6%)

Table 1: Main pretraining results for 4-bit BReD. In (a), NR, SR, and BReD use the same scaling rule, quantization grid, optimizer update, and hyperparameters; numbers are validation PPL when training completes. In (b), changes are relative to the corresponding full-precision optimizer-state baseline; ΔAvg. is the change in average zero-shot accuracy. "Opt. mem." denotes optimizer-state memory only.

The same key is used when the stored state is reconstructed, so the same $r_j^{(t)}$ is replayed without storing a random tensor.

BReD shares one dither value within each block. Let $\{B_j\}$ be a partition of a state tensor into blocks of size B. For all entries $i \in B_j$, BReD uses the same $r_j^{(t)}$ in Eq. (10). Since $r_j^{(t)}$ is uniform, each coordinate remains marginally unbiased under the local non-clipped model. The sharing introduces within-block correlation, but reduces the number of random draws from $O(d)$ to $O(d/B)$ for a state with d entries.

Detailed pseudocode for one EMA optimizer-state tensor is given in Appendix C.

4. Experiments

We evaluate BReD in language-model pretraining. The experiments focus on two questions: whether the rounding rule matters under a fixed low-bit format, and whether 4-bit BReD remains close to full-precision optimizer-state training at larger scales while reducing optimizer-state memory. Unless otherwise stated, all low-bit variants use the same optimizer update, learning-rate schedule, and hyperparameters as the corresponding full-precision optimizer-state baseline; only the rounding and reconstruction rule for the EMA optimizer states is changed. We report validation perplexity (PPL) and average zero-shot accuracy over standard language-model evaluation tasks. Full training setup, task-wise results, 3-bit results, and ablations are provided in Appendix E.

Rounding controls. Table 1(a) isolates the effect of the rounding rule at the 120M scale. NR, SR, and BReD use the same scale, quantization grid, optimizer update, and hyperparameters. NR and SR diverge or substantially degrade across the five optimizers, while 4-bit BReD remains stable for all of them. This controlled comparison shows that stability is not a consequence of the low-bit format alone; the rounding and reconstruction rule matters for EMA-state storage.

Scale check. We next evaluate AdamW and Muon at 1.1B and 3.4B parameters. Table 1(b) shows that 4-bit BReD closely tracks the full-precision optimizer-state baselines. Across these four settings, validation PPL changes by at most 0.3, and average zero-shot accuracy changes by at most 0.5 points. The result is consistent across both AdamW, which stores first- and second-moment estimates, and Muon, which stores momentum buffers.

Optimizer-state memory. The same table reports the memory used by optimizer states. At 3.4B parameters, 4-bit BReD reduces AdamW optimizer-state memory from 25.53GB to 3.58GB, and Muon optimizer-state memory from 13.53GB to 3.17GB. These reductions are for optimizer states only; they do not include parameters, gradients, or activations. Together with the controlled rounding results, these experiments support the main claim: BReD provides stable 4-bit EMA-state storage in the evaluated large-scale pretraining settings while substantially reducing optimizer-state memory.

5. Conclusion and Limitations

We studied how quantization errors accumulate in low-bit storage of EMA optimizer states. The analysis shows that small systematic bias can accumulate into persistent state error, while variance is amplified. BReD addresses this issue by applying subtractive dithering to EMA-state storage, using deterministic seed replay and block-wise shared dither values to avoid storing auxiliary random tensors. It keeps the optimizer updates and hyperparameters unchanged. In the evaluated pretraining settings, 4-bit BReD closely tracks full-precision optimizer-state baselines at 1.1B and 3.4B scale while substantially reducing optimizer-state memory.

The main limitation lies in the experimental scope. Our largest experiments are up to 3.4B parameters, with large-scale results focused on AdamW and Muon. Broader validation on larger models, longer runs, additional optimizers, and distributed training settings remains future work.

References

[1] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.

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

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

[4] 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*, abs/1803.05457, 2018.

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

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

[7] Aman Gupta, Rafael Celente, Abhishek Shivanna, DT 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.

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

[9] NS Jayant and LR Rabiner. The application of dither to the quantization of speech signals. *Bell System Technical Journal*, 51(6):1293–1304, 1972.

[10] Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

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

[12] Bingrui Li, Jianfei Chen, and Jun Zhu. Memory efficient optimizers with 4-bit states. *Advances in Neural Information Processing Systems*, 36:15136–15171, 2023.

[13] 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.

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

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

[16] 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*, pages 1–16. IEEE, 2020.

[17] Samyam Rajbhandari, Olatunji Ruwase, Jeff Rasley, Shaden Smith, and Yuxiong He. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–14, 2021.

[18] Jie Ren, Samyam Rajbhandari, Reza Yazdani Aminabadi, Olatunji Ruwase, Shuangyan Yang, Minjia Zhang, Dong Li, and Yuxiong He. {Zero-offload}: Democratizing {billion-scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564, 2021.

[19] Lawrence Roberts. Picture coding using pseudo-random noise. *IRE Transactions on Information Theory*, 8(2):145–154, 1962.

[20] Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023.

[21] Bita Darvish Rouhani et al. Ocp microscaling formats (mx) specification, version 1.0, 2023.

[22] 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.

[23] L. Schuchman. Dither signals and their effect on quantization noise. *IEEE Transactions on Communication Technology*, 12(4):162–165, 1964. doi: 10.1109/TCOM.1964.1088973.

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

[25] Douglas T Sherwood. Some theorems on quantization and an example using dither. In *Nineteeth Asilomar Conference on Circuits, Systems and Computers, 1985.*, pages 207–212. IEEE, 1985.

[26] John Von Neumann and Herman Heine Goldstine. Numerical inverting of matrices of high order. 1947.

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

[28] Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*, 2024.

[29] Huaijin Wu, Bingrui Li, Yebin Yang, Yi Tu, Zhanpeng Zhou, Jianfei Chen, and Junchi Yan. Achieving low-bit muon through subspace preservation and grid quantization. In *The Fourteenth International Conference on Learning Representations*.

[30] 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.

[31] Huizhuo Yuan, Yifeng Liu, Shuang Wu, Xun Zhou, and Quanquan Gu. Mars: Unleashing the power of variance reduction for training large models. *arXiv preprint arXiv:2411.10438*, 2024.

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

[33] 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. *arXiv preprint arXiv:2406.16793*, 2024.

Appendix A. Related Work

Low-bit optimizer states. Optimizer states are a major memory bottleneck in large-scale training. Existing approaches reduce this cost structurally, as in Adafactor [24] and Adam-mini [33] or at the system level, as in ZeRO-style sharding/offloading [16–18]. We study a complementary axis: low-bit storage of EMA optimizer states. Prior work has shown strong results for block-wise 8-bit optimizers [5] and 4-bit AdamW [12]; we refer to the latter as Adam-LPMM. Recent methods further push low-bit AdamW states by focusing on EMA dynamics [30], or use optimizer-specific structure for Muon- [29] and Shampoo-style [7] states. In contrast, BReD targets the shared quantize–write-back path of EMA states. It changes the rounding and reconstruction rule used for state storage, while keeping the optimizer update and hyperparameters unchanged.

Unbiased rounding and dithering. Unbiased rounding reduces systematic rounding bias. Stochastic rounding is unbiased in expectation [1, 26]. Subtractive dithering is a classical technique that gives unbiased reconstruction [9, 19, 23, 25]; under the local non-clipped interval model used in 2, it has smaller worst-case conditional variance than stochastic rounding. However, subtractive dithering requires the same dither value at quantization and dequantization time, which is impractical at LLM scale if per-element dither values must be stored or generated. BReD makes subtractive dithering practical for persistent EMA optimizer states through deterministic seed replay and block-wise shared dither values.

Appendix B. Rounding

Nearest rounding (NR). NR deterministically selects the nearest endpoint:

$$c_{\mathrm{nr}}(\alpha) := 1\{\alpha \geq 1/2\}, \qquad \hat{\alpha} = c_{\mathrm{nr}}(\alpha). \tag{13}$$

NR is computationally efficient, but it can introduce biased rounding errors for asymmetric data distributions.

Stochastic rounding (SR). SR draws $u \sim \mathrm{Unif}(0,1)$ and selects the upper grid point with probability α:

$$c_{\mathrm{sr}}(\alpha; u) := 1\{u < \alpha\}, \qquad \hat{\alpha} = c_{\mathrm{sr}}(\alpha; u). \tag{14}$$

Then $\mathbb{E}[\hat{\alpha} \mid \alpha] = \alpha$, hence $\mathbb{E}[\epsilon_{\mathrm{elem}} \mid x] = 0$, and

$$\mathrm{Var}(\epsilon_{\mathrm{elem}} \mid x) = \Delta(x)^2 \alpha(1-\alpha) \leq \Delta(x)^2/4. \tag{15}$$

Although SR achieves unbiasedness, it can still leave a large variance, which is also amplified by the EMA recursion.

Appendix C. BReD Algorithm

Algorithm 1: BReD for one EMA optimizer-state tensor

Input: low-bit codes q_t and block scales a_t storing one EMA state at step t; update tensor u_{t+1}; EMA coefficient β; block partition $\{B_j\}$; global seed; state identifier $\mathrm{state_id}$

Output: low-bit storage of state $\hat{s}_{t+1}$ and block scales a_{t+1} storing the updated EMA state

Reconstruct the EMA state stored at step t.
foreach *block* B_j **do**
  $r_j^{(t)} \leftarrow \mathrm{PRNG}(\mathrm{seed}, \mathrm{state_id}, t, j)$
  Form the quantization grid for block B_j using scale $a_{t,j}$
  foreach *entry* $i \in B_j$ **do**
    $z_{t,i} \leftarrow$ the grid point represented by code $q_{t,i}$
    $\Delta_{t,i} \leftarrow$ the interval width used for the code $q_{t,i}$ under this grid
    $\hat{s}_{t,i} \leftarrow z_{t,i} - \Delta_{t,i}\left(r_j^{(t)} - \frac{1}{2}\right)$
  end
end

Apply the usual EMA update.
$\tilde{s}_{t+1} \leftarrow \beta \hat{s}_t + (1-\beta) u_{t+1}$

Quantize the updated EMA state for storage.
foreach *block* B_j **do**
  $r_j^{(t+1)} \leftarrow \mathrm{PRNG}(\mathrm{seed}, \mathrm{state_id}, t+1, j)$
  Compute the block scale $a_{t+1,j}$ from $\tilde{s}_{t+1,B_j}$
  Form the quantization grid for block B_j using scale $a_{t+1,j}$
  foreach *entry* $i \in B_j$ **do**
    $x_i \leftarrow \tilde{s}_{t+1,i}$
    if x_i *is clipped by the representable range* **then**
      $q_{t+1,i} \leftarrow$ the boundary code after clipping
    else
      Find neighboring grid points $p_0(x_i) \le x_i \le p_1(x_i)$
      $\Delta(x_i) \leftarrow p_1(x_i) - p_0(x_i)$
      $\alpha(x_i) \leftarrow \dfrac{x_i - p_0(x_i)}{\Delta(x_i)}$
      $c_i \leftarrow \mathbf{1}\{\alpha(x_i) + r_j^{(t+1)} \ge 1\}$
      if $c_i = 0$ **then**
        $q_{t+1,i} \leftarrow$ the code of $p_0(x_i)$
      else
        $q_{t+1,i} \leftarrow$ the code of $p_1(x_i)$
      end
    end
  end
end

The algorithm is written at the same local-interval level as Eq. (8)–(10). At reconstruction time, $z_{t,i}$ is the grid point represented by the stored code, and $\Delta_{t,i}$ is the corresponding local interval width under the same grid and quantization rule used when the code was written. For clipped entries, all

methods use the same clipping behavior; the unbiasedness and variance statements in the main text apply to non-clipped entries.

Appendix D. Proofs

This appendix provides the proofs for the EMA error accumulation results used in Section 2. All tensor-valued quantities can be flattened into vectors; we use $\|\cdot\|_2$ for the corresponding Euclidean norm, equivalently the Frobenius norm for tensors.

D.1. EMA error accumulation

Recall the recursion

$$e_{t+1} = \beta e_t + \epsilon_{t+1}, \qquad 0 < \beta < 1, \tag{16}$$

where e_t is the accumulated EMA-state error and ϵ_{t+1} is the one-step quantization error introduced when the updated EMA state is quantized and reconstructed.

Lemma 1 (Mean error under EMA quantization) *Assume that $\mathbb{E}[\epsilon_t] = b$ for all $t \geq 1$ and that $\mathbb{E}\|e_0\|_2 < \infty$. Then, for the recursion in Eq. (16),*

$$\mathbb{E}[e_t] = \beta^t \mathbb{E}[e_0] + \frac{1-\beta^t}{1-\beta} b. \tag{17}$$

In particular,

$$\lim_{t\to\infty} \mathbb{E}[e_t] = \frac{b}{1-\beta}. \tag{18}$$

Proof Unrolling Eq. (16) gives

$$e_t = \beta^t e_0 + \sum_{k=1}^{t} \beta^{t-k} \epsilon_k. \tag{19}$$

Taking expectation and using $\mathbb{E}[\epsilon_k] = b$ yields

$$\mathbb{E}[e_t] = \beta^t \mathbb{E}[e_0] + \sum_{k=1}^{t} \beta^{t-k} b = \beta^t \mathbb{E}[e_0] + \frac{1-\beta^t}{1-\beta} b. \tag{20}$$

Since $0 < \beta < 1$, the first term vanishes as $t \to \infty$, which proves the claim. ■

Theorem 2 (Steady-state second moment) *Assume*

$$\epsilon_t = b + \xi_t, \tag{21}$$

where $\{\xi_t\}_{t\geq 1}$ are independent over time, $\mathbb{E}[\xi_t] = 0$, and $\mathbb{E}\|\xi_t\|_2^2 = \sigma^2$ for all t. Assume also that e_0 has finite second moment and is independent of $\{\xi_t\}_{t\geq 1}$. Then

$$\lim_{t\to\infty} \mathbb{E}\|e_t\|_2^2 = \frac{\|b\|_2^2}{(1-\beta)^2} + \frac{\sigma^2}{1-\beta^2}. \tag{22}$$

Proof Using $\epsilon_t = b + \xi_t$, the unrolled recursion becomes

$$e_t = \beta^t e_0 + \sum_{k=1}^{t} \beta^{t-k} b + \sum_{k=1}^{t} \beta^{t-k} \xi_k. \tag{23}$$

Let

$$a_t := \frac{1-\beta^t}{1-\beta}. \tag{24}$$

Then

$$e_t = \beta^t e_0 + a_t b + \sum_{k=1}^{t} \beta^{t-k} \xi_k. \tag{25}$$

Because ξ_k are zero-mean and independent of e_0, the cross term between $\beta^t e_0 + a_t b$ and $\sum_{k=1}^{t} \beta^{t-k} \xi_k$ has zero expectation. Thus

$$\mathbb{E}\|e_t\|_2^2 = \mathbb{E}\|\beta^t e_0 + a_t b\|_2^2 + \mathbb{E}\left\|\sum_{k=1}^{t} \beta^{t-k} \xi_k\right\|_2^2. \tag{26}$$

For the first term, since $\mathbb{E}\|e_0\|_2^2 < \infty$ and $\beta^t \to 0$, we have $\beta^t e_0 \to 0$ in L^2, while $a_t b \to b/(1-\beta)$. Hence

$$\mathbb{E}\|\beta^t e_0 + a_t b\|_2^2 \longrightarrow \frac{\|b\|_2^2}{(1-\beta)^2}. \tag{27}$$

For the second term, independence and zero mean imply that the cross terms vanish: for $i \neq j$,

$$\mathbb{E}\langle \xi_i, \xi_j \rangle = 0. \tag{28}$$

Therefore

$$\mathbb{E}\left\|\sum_{k=1}^{t} \beta^{t-k} \xi_k\right\|_2^2 = \sum_{k=1}^{t} \beta^{2(t-k)} \mathbb{E}\|\xi_k\|_2^2 \tag{29}$$

$$= \sigma^2 \sum_{r=0}^{t-1} \beta^{2r} \tag{30}$$

$$= \sigma^2 \frac{1-\beta^{2t}}{1-\beta^2}. \tag{31}$$

Taking $t \to \infty$ in Eq. (26) gives Eq. (22). ■

Corollary 3 (Dependence on state dimension) *Consider a state with d entries. Under the assumptions of Theorem 2, define the average squared bias and average residual variance per coordinate as*

$$\mu^2 := \frac{\|b\|_2^2}{d}, \qquad \tau^2 := \frac{\sigma^2}{d}. \tag{32}$$

Then

$$\lim_{t\to\infty} \mathbb{E}\|e_t\|_2^2 = \frac{d\mu^2}{(1-\beta)^2} + \frac{d\tau^2}{1-\beta^2}. \tag{33}$$

Proof This follows by substituting $\|b\|_2^2 = d\mu^2$ and $\sigma^2 = d\tau^2$ into Eq. (22). ■

D.2. Local rounding facts

We also record the local rounding facts used in Section 3. Consider a scalar x inside a quantization interval with neighboring grid points $p_0(x) \le x \le p_1(x)$. Let

$$\Delta(x) := p_1(x) - p_0(x), \qquad \alpha(x) := \frac{x - p_0(x)}{\Delta(x)} \in [0, 1]. \tag{34}$$

For any reconstruction rule that returns

$$\hat{x} = p_0(x) + \Delta(x)\hat{\alpha}, \tag{35}$$

the local reconstruction error is

$$\epsilon_{\text{elem}}(x) := \hat{x} - x = \Delta(x)\big(\hat{\alpha} - \alpha(x)\big). \tag{36}$$

The following statements are local, i.e., they assume a fixed scale and quantization grid and do not include clipping or saturation effects.

Lemma 4 (Stochastic rounding) *Let $u \sim \mathrm{Unif}(0, 1)$ and define stochastic rounding by*

$$c_{\text{SR}}(\alpha; u) := \mathbf{1}\{u < \alpha\}, \qquad \hat{\alpha} := c_{\text{SR}}(\alpha; u). \tag{37}$$

Then

$$\mathbb{E}[\epsilon_{\text{elem}}(x) \mid x] = 0, \qquad \mathrm{Var}(\epsilon_{\text{elem}}(x) \mid x) = \Delta(x)^2\alpha(x)(1 - \alpha(x)) \le \frac{\Delta(x)^2}{4}. \tag{38}$$

Proof Since c_{SR} is Bernoulli with success probability α,

$$\mathbb{E}[\hat{\alpha} \mid x] = \alpha, \qquad \mathrm{Var}(\hat{\alpha} \mid x) = \alpha(1 - \alpha). \tag{39}$$

Multiplying by $\Delta(x)$ gives the mean and variance of $\epsilon_{\text{elem}}(x) = \Delta(x)(\hat{\alpha} - \alpha)$. The upper bound follows from $\alpha(1 - \alpha) \le 1/4$. ■

Lemma 5 (Subtractive dithering) *Let $u \sim \mathrm{Unif}(0, 1)$ and define subtractive dithering by*

$$c_{\text{SD}}(\alpha; u) := \mathbf{1}\{\alpha + u \ge 1\}, \qquad \hat{\alpha} := c_{\text{SD}}(\alpha; u) - u + \frac{1}{2}. \tag{40}$$

Then

$$\mathbb{E}[\epsilon_{\text{elem}}(x) \mid x] = 0, \qquad \mathrm{Var}(\epsilon_{\text{elem}}(x) \mid x) = \frac{\Delta(x)^2}{12}. \tag{41}$$

Proof Let $v := \{\alpha + u\}$ denote the fractional part of $\alpha + u$. Since u is uniform on $[0, 1]$, the shifted fractional part v is also uniform on $[0, 1]$. Moreover, because $c_{\text{SD}}(\alpha; u)$ is the integer part of $\alpha + u$ for $\alpha, u \in [0, 1]$,

$$\hat{\alpha} - \alpha = c_{\text{SD}}(\alpha; u) - u + \frac{1}{2} - \alpha \tag{42}$$

$$= \frac{1}{2} - \big(\alpha + u - c_{\text{SD}}(\alpha; u)\big) \tag{43}$$

$$= \frac{1}{2} - v. \tag{44}$$

Thus $\hat{\alpha} - \alpha$ is uniform on $[-1/2, 1/2]$. Therefore

$$\mathbb{E}[\hat{\alpha} - \alpha \mid x] = 0, \qquad \mathrm{Var}(\hat{\alpha} - \alpha \mid x) = \frac{1}{12}. \tag{45}$$

Multiplying by $\Delta(x)$ gives the stated result. ■

Lemma 6 (Coordinate-wise marginal unbiasedness under block-wise sharing) *Suppose a block B_k shares a single random variable $u^{(k)} \sim \mathrm{Unif}(0,1)$ across all coordinates in the block. If the underlying element-wise reconstruction rule is unbiased for every fixed coordinate value, then each coordinate remains marginally unbiased under block-wise sharing.*

Proof Fix a coordinate $i \in B_k$ and condition on its value x_i. Although the same $u^{(k)}$ is shared with other coordinates in the block, its marginal distribution for coordinate i is still uniform on $[0, 1]$. Therefore the same element-wise calculation applies:

$$\mathbb{E}[\hat{x}_i - x_i \mid x_i] = 0. \tag{46}$$

Block-wise sharing can introduce correlations among reconstruction errors within the same block, but it does not change the marginal unbiasedness of each coordinate. ■

Appendix E. Experimental Setup

This appendix provides the experimental details for the results in Section 4. Unless otherwise stated, for each optimizer and model scale we keep the model architecture, data recipe, training length, batch size, learning-rate schedule, optimizer hyperparameters, and evaluation protocol fixed across the full-precision optimizer-state baseline and the low-bit variants. In the controlled rounding comparisons, the scale, quantization grid, and clipping behavior are also kept fixed; only the rounding and reconstruction rule is changed.

E.1. Pretraining setup

We evaluate BReD in language-model pretraining with AdamW, Muon, NAdamW, SOAP, and MARS at the 120M scale, and with AdamW and Muon at the 1.1B and 3.4B scales. The 120M and 1.1B models use LLaMA-style decoder-only architectures following TinyLlama-style configurations, while the 3.4B model follows an OpenLLaMA-style configuration. All pretraining runs use SlimPajama as the training corpus. The standard 120M runs are trained for 10.5B tokens, and the 1.1B and 3.4B runs are trained for 31.4B tokens. The hyperparameters are given in Sec. F.

Standard pretraining results are reported as mean and standard deviation over three runs.

E.2. Quantized optimizer states

BReD is applied to EMA optimizer states, such as the first and second moments across five optimizers. For Muon-style optimizers, we quantize the momentum buffer. For SOAP, we quantize the moment estimates and the preconditioner states. Following prior low-bit optimizer-state work [5], we keep embedding-layer optimizer states in full precision during pretraining.

For 4-bit storage, we use the MXFP4 [20, 21] format with block scaling. For 3-bit storage, we use the grid $\{0, \pm 1, \pm 2, \pm 3\}$ with Microscaling [20, 21]. Unless otherwise stated, BReD uses block size $B = 32$. The block-size ablation in Appendix G.1 compares this default against element-wise dithering and larger blocks.

All rounding baselines use the same scale, quantization grid, and clipping behavior as BReD. Nearest rounding (NR), stochastic rounding (SR), and BReD therefore differ only in the rounding and reconstruction rule. This controlled setup is used to isolate the effect of the rounding rule on EMA-state storage.

E.3. Baselines and low-bit variants

The full-precision optimizer-state baseline uses the same optimizer and training recipe, but stores the selected optimizer states in full precision. The NR and SR baselines replace BReD's subtractive dithering with nearest rounding or stochastic rounding while keeping the same low-bit format.

We also compare with optimizer-specific low-bit methods where applicable, including AdamW-LPMM for AdamW and Muon-Grasp for Muon. When BReD is combined with an optimizer-specific low-bit method, the optimizer-specific method is kept fixed and BReD is used as the rounding and reconstruction rule for the selected EMA states.

E.4. Evaluation protocol

For pretraining, we report validation perplexity and zero-shot language-model evaluation accuracy. The zero-shot tasks are HellaSwag [32], OpenBookQA [15], WinoGrande [22], ARC [4], BoolQ [3], and PIQA [2]. The average zero-shot accuracy is the arithmetic mean over these tasks.

E.5. Packed low-bit storage and memory accounting

All reported BReD low-bit runs use physically packed optimizer-state tensors rather than high-precision emulation. The implementation stores encoded optimizer states in packed integer tensors, reconstructs them during the optimizer step, performs the EMA update in higher precision, and then requantizes and packs the updated states back to memory.

The memory footprint reported in the paper refers to the optimizer-state memory only. For full-precision optimizer-state baselines, this counts the selected optimizer-state tensors in full precision. For BReD, this counts the packed low-bit codes and the associated scale information. The reported optimizer-state memory does not include model parameters, gradients, activations. For settings where embedding-layer optimizer states are kept in full precision, their memory is included in the corresponding optimizer-state memory total.

E.6. Randomness and reproducibility

BReD uses a stateless keyed pseudo-random number generator to generate dither values from the global seed, state identifier, training step, and block index. This allows the same dither value to be regenerated during reconstruction without storing an auxiliary random tensor. The keyed construction also makes the generated dither values independent of kernel scheduling or block execution order.

When multiple runs are reported, we report mean and standard deviation over completed runs. All low-bit variants are evaluated with the same training recipe as their corresponding full-precision optimizer-state baseline.

Appendix F. Pretraining Optimizer Hyperparameters

This appendix reports the complete optimizer hyperparameters used in all experiments. Unless otherwise stated, we keep **all training settings identical** across methods (model architecture, data recipe, training length, batch size, learning-rate schedule, and evaluation), and only vary the **storage precision** of optimizer states.

F.1. Global settings

All runs share the following optimizer-related settings:

- **Gradient clipping:** global norm clip at 1.0 (disabled if set to 0).
- **Epsilon:** $\epsilon = 1 \times 10^{-8}$.
- **LR scheduler:** WSD [8, 28].
- **Warmup ratio:** 0.1 (warmup steps = $0.1\times$ total steps).
- **Min LR ratio:** 0.1 (minimum LR = $0.1\times$ peak LR).

F.2. Hyperparameters for 120M pretraining

Table 2 summarizes the optimizer hyperparameters used for pretraining the 120M model. Unless otherwise noted, all unspecified entries are not applicable to the corresponding optimizer.

Table 2: Optimizer hyperparameters for the 120M pretraining runs.

	AdamW	Muon	NAdamW	SOAP	MARS
Peak LR	1×10^{-3}	1×10^{-3}	1×10^{-3}	2×10^{-3}	1×10^{-3}
(β_1, β_2)	(0.9, 0.95)	—	(0.9, 0.95)	(0.95, 0.95)	(0.9, 0.99)
Weight decay	0.1	0.1	0.1	0.01	0
Weight decay (1D params)	—	—	—	—	0.01
Muon momentum β	—	0.95	—	—	—
NS steps	—	5	—	—	—
Momentum decay	—	—	0.004	—	—
Shampoo β	—	—	—	0.95	—

F.3. Settings for larger models

Following Liu et al. [13], we use the same learning rate and weight decay for Muon and AdamW. For the 1.1B and 3.4B models, we use:

- **Peak LR for 1.1B model:** 6×10^{-4}.

- **Peak LR for 3.4B model:** 6×10^{-4}.
- **AdamW betas:** $(\beta_1, \beta_2) = (0.9, 0.95)$.
- **Muon momentum:** $\beta = 0.95$.
- **NS steps:** 5.

Appendix G. Full pretraining results

Table 3: Zero-shot LM evaluation accuracies for the 120M model. Subscripts denote standard deviations over three runs. Parentheses report the difference relative to the full-precision optimizer-state baseline of the same optimizer.

Optimizer	HS	OBQA	WG	Arc_e	Arc_c	BoolQ	PIQA	Avg.↑
AdamW	$29.41_{0.4}$	$27.60_{0.7}$	$51.20_{0.4}$	$36.36_{0.1}$	$21.79_{1.1}$	$56.12_{1.9}$	$60.58_{0.6}$	40.4(+0.0)
AdamW-LPMM	$29.15_{0.2}$	$27.73_{0.8}$	$50.64_{0.7}$	$35.66_{1.0}$	$22.58_{0.9}$	$57.44_{1.9}$	$58.94_{1.0}$	40.3(-0.1)
AdamW-LPMM+**BReD**	$28.91_{0.1}$	$26.33_{1.3}$	$51.88_{0.4}$	$35.80_{0.3}$	$22.18_{0.8}$	$59.35_{0.4}$	$59.21_{0.6}$	40.5(+0.1)
AdamW-4bit+**BReD**	$29.12_{0.2}$	$27.07_{0.6}$	$52.04_{0.7}$	$35.84_{1.0}$	$21.82_{0.7}$	$59.35_{1.0}$	$59.58_{0.5}$	40.7(+0.3)
AdamW-3bit+**BReD**	$28.68_{0.2}$	$27.27_{0.5}$	$51.93_{0.2}$	$36.10_{0.9}$	$22.33_{0.6}$	$59.63_{1.3}$	$58.62_{0.6}$	40.6(+0.2)
Muon	$29.85_{0.1}$	$26.73_{0.4}$	$51.30_{0.5}$	$36.60_{0.3}$	$22.50_{0.3}$	$59.16_{2.6}$	$60.19_{0.3}$	40.9(+0.0)
Muon-4bit+**BReD**	$29.44_{0.2}$	$27.13_{1.5}$	$52.57_{1.1}$	$36.67_{1.3}$	$22.55_{0.3}$	$60.15_{1.7}$	$60.08_{0.9}$	41.2(+0.3)
Muon-3bit+**BReD**	$29.02_{0.1}$	$27.47_{0.6}$	$51.49_{0.8}$	$36.74_{0.4}$	$22.24_{0.5}$	$59.67_{0.6}$	$59.99_{0.4}$	40.9(+0.0)
NAdamW	$28.91_{0.5}$	$26.47_{0.4}$	$50.85_{0.9}$	$36.08_{0.4}$	$21.96_{0.9}$	$56.71_{2.3}$	$59.39_{0.4}$	40.1(+0.0)
NAdamW-4bit+**BReD**	$29.98_{0.3}$	$26.93_{1.2}$	$49.59_{1.1}$	$36.00_{0.6}$	$22.04_{0.8}$	$60.44_{2.4}$	$60.01_{0.1}$	40.6(+0.5)
NAdamW-3bit+**BReD**	$28.91_{0.2}$	$25.93_{1.4}$	$51.12_{1.6}$	$35.64_{0.7}$	$21.96_{0.5}$	$60.20_{2.4}$	$59.67_{0.6}$	40.5(+0.4)
SOAP	$29.69_{0.1}$	$24.93_{0.3}$	$50.12_{1.2}$	$36.60_{0.7}$	$22.47_{0.5}$	$54.08_{8.8}$	$59.43_{1.1}$	39.6(+0.0)
SOAP-4bit+**BReD**	$28.97_{0.1}$	$27.20_{0.2}$	$50.54_{0.9}$	$36.02_{0.7}$	$21.73_{0.4}$	$58.03_{5.1}$	$59.36_{0.6}$	40.3(+0.7)
SOAP-3bit+**BReD**	$28.86_{0.1}$	$26.47_{0.8}$	$50.70_{0.5}$	$35.62_{0.8}$	$22.47_{0.8}$	$61.04_{1.7}$	$59.01_{1.2}$	40.6(+1.0)
MARS	$29.03_{0.1}$	$27.20_{0.5}$	$51.65_{1.1}$	$36.00_{1.0}$	$21.56_{0.5}$	$60.89_{2.3}$	$59.25_{1.0}$	40.8(+0.0)
MARS-4bit+**BReD**	$28.88_{0.1}$	$25.20_{0.7}$	$50.54_{0.3}$	$35.69_{0.3}$	$22.58_{0.7}$	$59.67_{1.0}$	$59.33_{0.3}$	40.4(-0.4)
MARS-3bit+**BReD**	$28.59_{0.3}$	$27.47_{1.2}$	$51.93_{0.5}$	$35.91_{0.6}$	$22.55_{1.2}$	$61.22_{1.0}$	$59.49_{1.0}$	41.0(+0.2)

Table 4: Zero-shot LM evaluation accuracies and validation PPL on 1.1B and 3.4B models. Subscripts denote standard deviations over three runs. Parentheses report the difference relative to the full-precision optimizer-state baseline of the same optimizer.

#PARAM	OPTIMIZER	HS	OBQA	WG	ARC_E	ARC_C	BOOLQ	PIQA	AVG. ↑	PPL ↓
1.1B	ADAMW	$43.45_{0.1}$	$30.80_{0.4}$	$52.62_{0.5}$	$45.38_{0.4}$	$25.66_{0.7}$	$59.66_{0.1}$	$68.05_{0.7}$	46.5(+0.0)	10.8(+0.0)
	ADAMW-4BIT+**BREED**	$43.05_{0.3}$	$30.67_{1.3}$	$52.12_{0.2}$	$45.50_{2.0}$	$25.51_{0.5}$	$59.63_{1.8}$	$68.03_{0.2}$	46.4(-0.1)	10.9(+0.1)
	MUON	$46.01_{0.5}$	$31.07_{0.6}$	$52.75_{1.1}$	$48.23_{1.1}$	$26.37_{0.5}$	$59.57_{2.7}$	$69.10_{0.5}$	47.6(+0.0)	10.3(+0.0)
	MUON-4BIT+**BREED**	$44.46_{0.4}$	$30.87_{1.6}$	$53.57_{0.3}$	$47.28_{1.1}$	$25.57_{0.6}$	$58.96_{4.2}$	$68.93_{0.3}$	47.1(-0.5)	10.6(+0.3)
3.4B	ADAMW	$48.01_{0.3}$	$31.33_{0.3}$	$53.99_{1.8}$	$48.45_{0.4}$	$26.93_{0.4}$	$56.47_{4.7}$	$69.98_{0.6}$	47.9(+0.0)	9.7(+0.0)
	ADAMW-4BIT+**BREED**	$47.08_{0.7}$	$31.47_{0.7}$	$52.80_{1.8}$	$47.53_{1.2}$	$26.08_{0.2}$	$59.25_{1.0}$	$69.24_{0.2}$	47.6(-0.3)	9.8(+0.1)
	MUON	$48.37_{0.2}$	$31.53_{0.3}$	$54.91_{1.3}$	$49.17_{0.4}$	$26.48_{0.1}$	$57.50_{2.8}$	$69.68_{0.5}$	48.2(+0.0)	9.6(+0.0)
	MUON-4BIT+**BREED**	$46.31_{0.0}$	$30.47_{0.3}$	$54.96_{0.9}$	$47.11_{1.2}$	$26.23_{0.4}$	$59.77_{1.0}$	$68.74_{0.4}$	47.7(-0.5)	9.8(+0.2)

G.1. 3-bit results and block-size ablation

The main results use 4-bit BReD as the default operating point. We also evaluate 3-bit BReD as a more aggressive compression setting. As summarized in Table 5(a), 3-bit BReD remains convergent in all evaluated pretraining and fine-tuning settings, but the quality shifts are larger and less uniform than in the 4-bit case. We therefore treat 3-bit BReD as an alternative rather than the default setting.

We further ablate the block-wise randomization component of BReD. Block-wise randomization (BWR) shares one dither value within each block, reducing PRNG calls from $O(d)$ to $O(d/B)$ while introducing within-block correlation. Table 5(b) shows that the default $B = 32$ remains close to element-wise dithering in quality, while reducing the number of generated random values by $32\times$ and lowering generation time from 486.32 ± 29 ms to 340.88 ± 41 ms.

Table 5: Aggressive compression and BWR ablation. (a) Summary of 3-bit BReD results. ΔAvg. denotes the range of changes relative to the corresponding full-precision optimizer-state baseline. (b) Block-wise randomization ablation for 4-bit AdamW at the 120M scale.

(a) 3-bit BReD summary

Setting	Architecture / size	Optimizers	ΔAvg.	ΔPPL
Pretrain	LLaMA-style / 120M	5 opt.	$[+0.0, +1.0]$	$[+0.2, +1.2]$
Pretrain	LLaMA-style / 1.1B	AdamW, Muon	$[-0.9, -0.7]$	$[+0.1, +0.6]$

(b) BWR ablation

Block size B	Avg. acc. $\uparrow$	PPL $\downarrow$	Random ratio	Generation time
1	40.9	18.8	1	486.32 ± 29 ms
32	40.7	18.7	$1/32$	340.88 ± 41 ms
256	40.6	18.8	$1/256$	335.47 ± 35 ms