OPT2025: 17th Annual Workshop on Optimization for Machine Learning

M+Adam: Stable Low-Precision Training with Combined Adam–Madam Updates

Xiaoyuan Liang XLIANG2@CALTECH.EDU
California Institute of Technology

Sebastian Loeschcke SBL@DI.KU.DK
University of Copenhagen

Mads Toftrup TOFTRUP@CS.AU.DK
Aarhus University

Anima Anandkumar ANIMA@CALTECH.EDU
California Institute of Technology

Abstract

Training large language models (LLMs) in full precision (FP32) is increasingly constrained by memory, compute, and energy demands. Low-precision formats such as BF16, which modern accelerators are optimized for, offer substantial gains, reducing memory footprint, improving throughput, and lowering energy consumption. However, when training is performed entirely in low precision, without FP32 master weights or optimizer states, it typically underperforms compared to full-precision training. Standard additive optimizers like Adam often diverge in this regime, as small updates vanish below the mantissa resolution while large ones overflow the representable range. We introduce M+ADAM, an optimizer that enables stable, fully low-precision training by jointly applying additive and multiplicative updates. Each weight is represented as a mantissa–exponent pair, where Adam refines the mantissa and Madam adjusts the exponent. This dual-path update aligns the optimizer dynamics with floating-point structure: additive updates provide fine intra-bin control, while multiplicative updates traverse quantization bins. Theoretically, we prove monotone descent under standard smoothness assumptions. Empirically, M+ADAM trains LLaMA-style models in pure BF16 (no FP32 copies) and matches the perplexity of full-precision Adam across 60M–350M parameter scales, providing a practical step toward end-to-end low-precision optimization.

1. Introduction

Training large language models (LLMs) in full precision (FP32) incurs substantial memory, compute, and energy costs. Modern accelerators are optimized for reduced-precision formats such as BF16 and FP8, which provide higher throughput and lower energy per FLOP. Consequently, most large-scale training systems adopt mixed-precision pipelines [1–3], running forward and backward passes in BF16 while maintaining full-precision (FP32) master weights. Training in pure low precision, where both arithmetic and optimizer states are stored in BF16, promises even greater efficiency gains in speed, memory, and energy consumption, but remains challenging in practice.

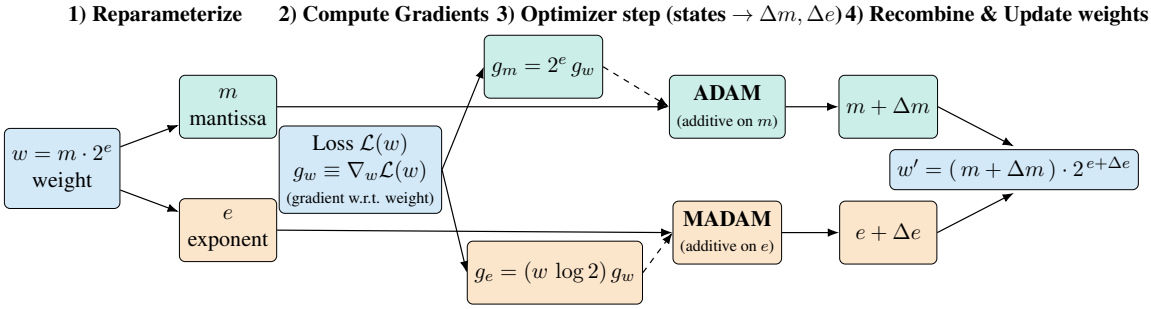

Figure 1: Overview of M+Adam.

Pure low-precision training is often unstable. Standard additive optimizers such as Adam are not well aligned with the spacing of BF16 arithmetic: small updates fall below the mantissa resolution and vanish leading to stagnation. This leads to degraded convergence and, in many cases, diverges from full-precision training [4]

Several strategies have been proposed to stabilize training under low-precision arithmetic, yet none fully resolve the problem. Mixed-precision pipelines alleviate instability by retaining FP32 master weights and optimizer states, but this reintroduces significant memory overhead and frequent FP32–BF16 casts. Multiplicative or scale-aware optimizers such as Madam [5] and its log-domain variant LNS-Madam [6] achieve robust low-precision behavior by operating directly in logarithmic space, but require arithmetic not natively supported on current hardware. Stochastic rounding offers another way to mitigate rounding bias. However, our experiments indicate that it does not fully match the performance of full-precision Adam especially in higher data to parameter regimes. Overall, existing approaches either depend on high-precision storage, specialized hardware, or incur a performance gap relative to full-precision training. Consequently, fully low-precision optimization remain an open challenge.

A floating-point parameter w can be written as $w = m\,2^e$, where the mantissa m encodes fine-grained variation within a quantization bin, and the exponent e sets the overall scale. In low-precision formats such as BF16, the mantissa has limited precision (7 bits), making small additive updates likely to round away, whereas the exponent, though more expressive (8 bits), changes too coarsely to handle small-scale refinements. This asymmetry motivates separating m and e and updating them with rules suited to their respective roles. Prior work has partially explored this direction: Madam [6] proposes a multiplicative variant of Adam that achieves stable learning rates by performing weight updates on a logarithmic scale, although arithmetic operations remain in FP32. LNS-MADAM [6] expands on this concept by fully converting weights, activations, and matrix multiplications into a specialized logarithmic number system (LNS), enabling entirely multiplicative updates to exponents. While this approach achieves stable low-bit training and FP32-level accuracy, it relies on custom log-domain arithmetic and specialized hardware support.

We introduce M+ADAM, that combines Madam and Adam updates to enable low-precision training without the need for retaining full-precision master weights (see Fig. 1). Each weight is stored directly in BF16 and decomposed into an exponent-mantissa pair $w = m\,2^e$ only temporarily and layerwise during the update step. Crucially, unlike LNS-Madam, our approach operates entirely within standard floating-point arithmetic using BF16 and thus requires no specialized hardware or

custom number system. The update applies Adam-style additive steps to the mantissa and additive updates to the exponent, which similar to Madam translates into multiplicative changes in the weight through its 2^e scaling. This preserves Madam's low-precision training property while adding Adam-like control over the mantissa within each bin, providing fine alignment where multiplicative changes alone are coarse and yielding superior validation perplexity in our experiments.

We theoretically show that M+ADAM is a valid descent method under standard smoothness assumptions. By analyzing the loss function in terms of separate exponent and mantissa parameters, we demonstrate that performing sequential exponent and mantissa updates guarantees a one-step decrease in loss under suitable step sizes. Empirically, we validate that M+ADAM enables stable and accurate training entirely in BF16, eliminating the need for FP32 master weights or optimizer states. M+ADAM matches FP32 Adam performance and consistently outperforms BF16 Adam variants in validation loss. Together, our results confirm that combining Adam-style mantissa refinement with Madam-style exponent updates yields a principled, practical solution for fully low-precision optimization, reducing memory without sacrificing accuracy.

2. Method

2.1. Mantissa–Exponent Updates

Let $w \in \mathbb{R}^d$ denote a parameter vector. We represent each element w_i in floating-point form as:

$$w_i = m_i \cdot 2^{e_i}, \tag{1}$$

where m_i is the mantissa and e_i is the exponent. Instead of updating w_i directly, we maintain and optimize the two components (m_i, e_i) separately.

In our M+Adam optimizer, the mantissa is updated using Adam [7]:

$$m_i^{(t+1)} \leftarrow \mathrm{Adam}(m_i^{(t)}, g_{m_i}^{(t)}), \tag{2}$$

while the exponent is updated using Madam [8]:

$$e_i^{(t+1)} \leftarrow \mathrm{Madam}(e_i^{(t)}, g_{e_i}^{(t)}), \tag{3}$$

where g_{m_i} and g_{e_i} are the projected gradients for the mantissa and exponent, respectively. Both updates are applied within each optimization step, and the parameters are recombined as:

$$w_i^{(t+1)} = m_i^{(t+1)} \cdot 2^{e_i^{(t+1)}}. \tag{4}$$

2.2. Mathematical Model

To analyze parallel mantissa–exponent updates, we absorb the sign into the mantissa and write

$$w \;=\; m\,2^e, \qquad m \in [-1, -\tfrac{1}{2}] \cup [\tfrac{1}{2}, 1), \quad e \in \mathbb{Z},$$

and optimize the reparameterized objective $F(e, m) = \ell(m\,2^e)$. By the chain rule, with $g_w = \partial\ell/\partial w$,

$$\nabla_m F \;=\; g_w\,2^e, \qquad \nabla_e F \;=\; g_w\,w\,\log 2, \tag{5}$$

Algorithm 1: M+ADAM

Numerical representation: weights w with mantissa–exponent form $w = ldexp(e, m)$; max weight $w_{\max}$.
Optimisation parameters: step sizes η_m, η_e; max perturbations $\eta_m^\star, \eta_e^\star$; averages β_1, β_2; small ϵ.
Initial state: $\nu_m, \mu_m, \nu_e, \mu_e \leftarrow 0$ ▷ ν: second-moment; μ: first-moment.

repeat

$g \leftarrow \text{STOCHASTICGRADIENT}()$ ▷ gradient in weight space
$(e, m) \leftarrow frexp(w)$
$g_m \leftarrow 2^e\, g; \quad g_e \leftarrow (w \log 2)\, g$
$\nu_m \leftarrow (1-\beta_2)\, g_m^2 + \beta_2\, \nu_m \quad \mu_m \leftarrow \beta_1\, \mu_m + (1-\beta_1)\, g_m$ ▷ moment updates
$\nu_e \leftarrow (1-\beta_2)\, g_e^2 + \beta_2\, \nu_e \quad \mu_e \leftarrow \beta_1\, \mu_e + (1-\beta_1)\, g_e$
`// Adam update on the mantissa (additive in m)`
$m \leftarrow m - \eta_m \operatorname{clamp}_{\eta_m^\star/\eta_m}\Big(\mu_m/(\sqrt{\nu_m} + \epsilon)\Big)$ ▷ clamp mantissa update between $\pm\eta_m^\star/\eta_m$
`// Madam update on the exponent (additive in e)`
$e \leftarrow e - \eta_e \operatorname{clamp}_{\eta_e^\star/\eta_e}\Big(\mu_e/(\sqrt{\nu_e} + \epsilon)\Big)$ ▷ clamp exponent update between $\pm\eta_e^\star/\eta_e$
$w \leftarrow \operatorname{clamp}_{w_{\max}}(ldexp(e, m))$ ▷ clamp weights between $\pm w_{\max}$

until *converged*

showing how a single weight gradient splits across the two blocks.

Let $w = m \odot 2^e$ (elementwise). For any weight gradient g_w, define the linear maps

$$D_m(e,m)[g_w] \;=\; g_w \odot 2^e, \qquad D_e(e,m)[g_w] \;=\; g_w \odot \big(m\, 2^e \log 2\big),$$

and let $g_w^{(0)} = \nabla_w \ell(w)$ evaluated at (e, m).

Theorem 1 *Assume: (i) ℓ has an L_w-Lipschitz continuous gradient on a compact set $W \subset \mathbb{R}^d$; (ii) the reparameterization is confined to*

$$\alpha \le 2^e \le \beta, \qquad \|m \odot 2^e\|_\infty \le M,$$

so that $w = m \odot 2^e \in W$. Then there exist finite constants $L_m, L_e, L_{me} > 0$ (depending only on L_w, α, β, M) such that the one–step parallel *update*

$$m' \;=\; m - \alpha_m\, D_m(e,m)[g_w^{(0)}], \qquad e' \;=\; e - \alpha_e\, D_e(e,m)[g_w^{(0)}]$$

satisfies

$$F(e', m') \;\le\; F(e, m)$$

whenever the step sizes obey

$$\max\Big\{\alpha_m L_m,\; \alpha_e L_e,\; \alpha_m \alpha_e L_{me}\Big\} \;<\; \eta$$

for some universal $\eta \in (0, 1)$.

Full constants and proofs are deferred to App. B, where we quantify L_m, L_e, L_{me} from L_w, α, β, M and give the detailed inequality.

3. Experiments

We now evaluate M+ADAM on large-scale language modeling tasks to assess its optimization efficiency under fully BF16 training. This section outlines our experimental setup, hyperparameter tuning protocol, and fairness controls applied across all baselines. Section 3.1 describes the shared environment and datasets, while Appendix E provides full hyperparameter grids and coordinate-descent results. We follow the fair-comparison guidelines of [9] to ensure reproducible and unbiased optimizer evaluations.

3.1. General Experimental Setup

All pretraining runs use the English portion of the C4 corpus [10]. Text is tokenized using a T5-compatible SentencePiece tokenizer and packed into fixed-length sequences of $L = 512$. We train LLaMA-style decoder-only Transformers [11], initialized from scratch and report results for three model sizes that are approximately 60M, 130M, and 350M. All experiments use a global batch size of 131K tokens with warmup, and a maximum sequence length of 512 tokens. Warmup lengths vary between 5% and 15% of the total training steps. In our results we report validation loss on C4.

Data budgets and evaluation. For a model with P trainable parameters (in millions), the $1\times$ Chinchilla token budget is defined as

$$T_{1\times} \approx 20\,P \times 10^6 \text{ tokens}.$$

Full optimizer and training hyperparameters are listed in Appendix E.

Optimization and baselines. Our primary optimizer is M+ADAM, applied in pure BF16 precision for both parameters and activations. Baselines include Adam, AdamW, and their stochastic rounding (SR) variants, all trained under identical precision settings; FP32 Adam and AdamW runs are included for comparison. We apply global gradient-norm clipping, use decoupled weight decay for AdamW-style methods, and keep all other settings identical across optimizers.

3.2. Fine-grained Hyperparameter Coordinate Descent

Fixed or lightly tuned baselines can mask an optimizer's true capability and inflate apparent speedups [9]. To make optimizer comparisons fair and reproducible, we perform a systematic, one–coordinate–at–a–time search that identifies near–optimal settings in each training regime.

For each regime the effective global batch is held fixed and the training horizon is set by the target token budget; evaluation is performed on the C4 validation split at regular intervals. For every optimizer under study we define a discrete grid for each exposed hyperparameter. Across all methods, we tune the learning rate, warmup length, β_1, β_2, ϵ, gradient-norm clipping, and total batch size. For Adam and AdamW, we additionally vary the decoupled weight decay. For M+ADAM, we include mantissa and exponent learning rates, exponent scheduling, and the possibility of applying decoupled weight decay separately in mantissa and exponent spaces.

The coordinate-descent procedure follows a coarse-to-fine refinement strategy. Each search begins from a well-performing base configuration and proceeds by selecting one hyperparameter to sweep while holding the others fixed. After completing the sweep, we update the hyperparameter to the best-performing grid value only if it yields an improvement in validation loss of at least

Table 1: Validation loss at $1\times$ Chinchilla for LLaMA-60M/130M. Lower is better.

Model	Regime	Optimizer	Loss ↓
60M	BF16 (no SR)	M+Adam	3.343
		AdamW	3.399
		Adam	3.410
	BF16 + SR	Adam	3.365
		AdamW	3.401
	FP32	AdamW	3.333
		Adam	3.346
130M	BF16 (no SR)	M+Adam	3.059
		Adam	3.226
		AdamW	3.238
	BF16 + SR	Adam	3.099
		AdamW	3.168
	FP32	AdamW	3.042
		Adam	3.082

SR = stochastic rounding. All BF16 rows use BF16 parameters & activations; FP32 rows use FP32 throughout.

$\Delta_{\text{loss}} \geq 5 \times 10^{-3}$. This iterative process produces tuned configurations that are both performant and comparable across optimizers, ensuring that the observed differences in validation loss reflect the optimization algorithm itself rather than hyperparameter sensitivity. The concrete grids used for coordinate descent for each method are provided in Appendix E.

4. Conclusion

We presented M+ADAM, an optimizer that combines additive (Adam-style) and multiplicative (Madam-style) updates by operating directly on the mantissa–exponent decomposition of floating-point parameters. This formulation aligns optimizer dynamics with floating-point structure: additive updates refine values within quantization bins, while multiplicative updates adjust scale across bins. We proved monotone descent under standard smoothness assumptions and showed empirically that M+ADAM enables stable, fully low-precision training in BF16 matching the performance of FP32 Adam while outperforming BF16 Adam and AdamW across 60M–350M parameter LLaMA models. Although M+ADAM does not yet reduce optimizer-state memory, it removes the need for FP32 master weights and provides a principled foundation for future low-precision optimizers. We see this as a first step toward end-to-end low-precision training, with natural next directions including billion-scale models, adaptive mantissa–exponent coordination, and integration with compressed optimizer states such as 8-bit or low-rank Adam variants.

References

[1] NVIDIA. *Transformer Engine*. https://developer.nvidia.com/transformer-engine. 2022.

[2] A. Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[3] PyTorch. *Automatic Mixed Precision with TorchAO*. https://pytorch.org/docs/stable/amp.html. 2025.

[4] Kaan Ozkara, Tao Yu, and Youngsuk Park. *Stochastic Rounding for LLM Training: Theory and Practice*. 2025. arXiv: 2502.20566 [cs.LG]. URL: https://arxiv.org/abs/2502.20566.

[5] Jeremy Bernstein et al. "Learning Compositional Functions via Multiplicative Weight Updates". In: *NeurIPS*. 2020.

[6] J. Zhao et al. "LNS-Madam: Low-Precision Training in Logarithmic Number System Using Multiplicative Weight Update". In: *arXiv preprint arXiv:2106.13914* (2021).

[7] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*. 2015.

[8] J. Bernstein et al. "Learning compositional functions via multiplicative weight updates". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020.

[9] Kaiyue Wen et al. *Fantastic Pretraining Optimizers and Where to Find Them*. v1, Sept. 2, 2025. 2025. arXiv: 2509.02046 [cs.LG]. URL: https://arxiv.org/abs/2509.02046.

[10] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[11] Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. "LLaMA: Open and Efficient Foundation Language Models". In: *arXiv preprint arXiv:2302.13971* (2023).

[12] N. Littlestone. "Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm". In: *Machine Learning* 2.4 (1988), pp. 285–318.

[13] J. Kivinen and M. K. Warmuth. "Exponentiated gradient versus gradient descent for linear predictors". In: *Information and Computation* 132.1 (1997), pp. 1–63.

[14] Y. Freund and R. E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139.

[15] A. Beck and M. Teboulle. "Mirror descent and nonlinear projected subgradient methods for convex optimization". In: *SIAM Journal on Optimization* 14.1 (2003), pp. 257–289.

[16] Y. You, I. Gitman, and B. Ginsburg. "Large batch training of convolutional networks". In: *arXiv preprint arXiv:1708.03888* (2017).

[17] Y. You et al. "Large batch optimization for deep learning: Training BERT in 76 minutes". In: *arXiv preprint arXiv:1904.00962* (2019).

[18] B. Jacob et al. "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference". In: *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*. 2018, pp. 2704–2713.

[19] R. Banner et al. "Scalable Methods for 8-bit Training of Neural Networks". In: *arXiv preprint arXiv:1805.11046* (2018).

[20] S. Zhou et al. "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients". In: *arXiv preprint arXiv:1606.06160* (2016).

[21] D. Miyashita, E. Lee, and B. Murmann. "Convolutional Neural Networks using Logarithmic Data Representation". In: *arXiv preprint arXiv:1603.01025* (2016).
[22] P. Micikevicius et al. "Mixed Precision Training". In: *arXiv preprint arXiv:1710.03740* (2017).
[23] NVIDIA, Arm, and Intel. *FP8 Formats for Deep Learning*. https://developer.nvidia.com/blog/nvidia-arm-and-intel-publish-fp8-specification-for-standardization-as-an-interchange-format-for-ai/. 2022.
[24] NVIDIA. *Using FP8 with Transformer Engine*. https://docs.nvidia.com/deeplearning/transformer-engine/user-guide/examples/fp8_primer.html. 2023.
[25] PyTorch Team. *TorchAO Documentation*. https://pytorch.org/ao/stable/api_ref_quantization.html. 2024.
[26] DeepSeek-AI. *DeepGEMM: Clean and Efficient FP8 GEMM Kernels with Fine-Grained Scaling*. https://github.com/deepseek-ai/DeepGEMM. GitHub repository. 2025.
[27] G. G. Turrigiano et al. "Activity-dependent scaling of quantal amplitude in neocortical neurons". In: *Nature* 391.6670 (1998), pp. 892–896.
[28] Keita Ibata, Qingxiu Sun, and Gina G. Turrigiano. "Rapid Synaptic Scaling Induced by Changes in Postsynaptic Firing". In: *Neuron* 57.6 (2008), pp. 819–826.
[29] Gina G. Turrigiano. "Homeostatic synaptic plasticity: local and global mechanisms for stabilizing neuronal function". In: *Cold Spring Harbor Perspectives in Biology* 4.1 (2012), a005736. DOI: 10.1101/cshperspect.a005736.
[30] M. C. W. van Rossum, G. Q. Bi, and G. G. Turrigiano. "Stable Hebbian learning from spike timing-dependent plasticity". In: *The Journal of Neuroscience* 20.23 (2000), pp. 8812–8821.
[31] Tara Keck et al. "Integrating Hebbian and Homeostatic Plasticity". In: *Philosophical Transactions of the Royal Society B* 372.1715 (2017), p. 20160158.
[32] Friedemann Zenke, Wulfram Gerstner, and Surya Ganguli. "The Temporal Paradox of Hebbian Learning and Homeostatic Plasticity". In: *Current Opinion in Neurobiology* 43 (2017), pp. 166–176.

Appendix A. Background

Multiplicative weight updates Multiplicative update rules have a long history in learning and optimization, including Winnow for linear classification [12], the Exponentiated Gradient method (EG) [13], and the Hedge algorithm underlying AdaBoost [14]. These methods perform updates that are relative to the current parameter values, often working well when model layers differ by orders of magnitude. The mirror-descent viewpoint formalizes this: with a KL/relative-entropy Bregman divergence, multiplicative rules arise as mirror steps in the log domain [15]. Recent work by Bernstein *et al.* proves a descent lemma tailored to compositional networks and introduces Madam, a multiplicative variant of Adam that substantially reduces LR tuning [8]. This line motivates optimizers that control relative (rather than absolute) step sizes—exactly the knob our mantissa–exponent formulation exposes.

Adaptive relative update rules Layer-wise relative scaling is central to stabilizing large-batch and low-precision training. LARS rescales per-layer updates by the ratio $\|g\|/\|w\|$ to equalize effective step sizes across layers at scale [16]; LAMB extends this idea to Transformer-style language models [17]. FROMAGE gives a geometric rationale—"deep relative trust"—and shows that constraining relative changes in weights improves robustness across tasks [5]. Our method inherits the same philosophy but acts at a finer granularity: we set and coordinate relative steps separately

for mantissas and exponents, which empirically lowers update variance while preserving descent guarantees.

Low-precision training and number systems Quantization-aware and post-training quantization make 8–16-bit models practical with minimal accuracy loss [18–20]. However, many pipelines still keep FP32 master weights and apply additive optimizers, which can be brittle when formats change. Logarithmic number systems (LNS) offer an alternative arithmetic where multiplication/division become additions/subtractions in log space, aligning naturally with multiplicative updates [21]. LNS-MADAM co-designs LNS with a multiplicative optimizer to perform updates directly in the log domain, showing stable low-bit training and hardware benefits [6]. Our approach is complementary: without committing to a pure-LNS stack, we explicitly factor parameters into mantissa and exponent and update both with coordinated multiplicative rules; this keeps the benefits of relative control while remaining compatible with mainstream FP formats and AMP.

Mixed-precision frameworks Mixed-precision training via FP16/BF16 (AMP) is now standard, using loss scaling and FP32 master copies to maintain stability [22]. NVIDIA's Transformer Engine adds FP8 (E4M3/E5M2) kernels and runtime calibration, further shrinking memory and boosting throughput [23, 24]. PyTorch's AMP/TorchAO simplify adoption and quantization workflows in practice [3, 25]. Yet these frameworks are largely orthogonal to optimizer design: they accelerate math but do not directly solve the hyperparameter sensitivity that arises when absolute step sizes collide with reduced dynamic range. Our factorized optimizer addresses this by operating with explicit, relative controls on mantissa and exponent—plug-and-play with AMP/FP8 stacks. Concurrently, DeepSeek's open-source DeepGEMM library provides optimized FP8 GEMMs for dense and MoE/grouped layouts with SM90/SM100 backends, including a packed UE8M0 scaling path on SM100 [26].

Mantissa–Exponent parameterizations and biophysical foundations Parameter factorizations that separate direction from scale (e.g., weight normalization) can stabilize optimization; writing $w = m \odot 2^e$ likewise exposes two complementary levers for controlling relative change. The resulting composite step mixes additive and multiplicative behaviors—allowing fine-grained shape changes with coordinated gain control. This design is motivated by biophysics: neurons combine fast Hebbian plasticity with slower homeostatic mechanisms. In homeostatic synaptic scaling, prolonged activity drives near-multiplicative rescaling of all excitatory synapses while preserving relative ratios [27–29]; weight-dependent STDP uses update sizes that depend on the current weight, preventing runaway growth [30]. Theory and reviews argue that rapid Hebbian changes must be counterbalanced by slower, often multiplicative, homeostasis for stability [31, 32]. Our optimizer operationalizes this division of labor: the mantissa rule (plasticity) adjusts local directions additively with low variance, while the exponent rule (homeostasis) multiplicatively adapts effective scales to keep activity in a safe dynamic range.

Appendix B. Proof

Assumption 1 (Smoothness of F) *The reparameterized objective $F : \mathbb{R}^2 \to \mathbb{R}$ has an L-Lipschitz continuous gradient, i.e.*

$$\|\nabla F(x) - \nabla F(y)\| \;\le\; L\,\|x - y\| \quad \forall x, y \in \mathbb{R}^2.$$

Lemma 2 (Lipschitz-continuous partial gradients) *Suppose $\ell(w)$ has an L_w-Lipschitz continuous gradient on a compact set W, and reparameterize $w = m\,2^e$ with (e, m) confined so that*

$$2^e \in [\alpha, \beta], \qquad |m\,2^e| \le M.$$

Then for $F(e, m) = \ell(m\,2^e)$ with partials $g_m = \partial F/\partial m$, $g_e = \partial F/\partial e$, there exist constants $L_m, L_e > 0$ such that

$$\|g_m(e,m) - g_m(e',m')\| \le L_m\big(|m-m'| + |e-e'|\big), \quad \|g_e(e,m) - g_e(e',m')\| \le L_e\big(|m-m'| + |e-e'|\big).$$

Remark (equivalence to a separate sign bit). If one instead writes $w = s\,m\,2^e$ with $s \in \{\pm 1\}$ and $m \in [\frac{1}{2}, 1)$, the correct chain rule gives $\frac{\partial F}{\partial m} = s\,g_w\,2^e$ and $\frac{\partial F}{\partial e} = g_w\,(s\,m\,2^e)\,\log 2$. Absorbing s into m yields (5) and avoids carrying s explicitly.

Proof Let $s > 0$ be fixed and set $F(e, m) = \ell(w)$ with $w = s\,m\,2^e$. Assume $\nabla\ell$ is L_w-Lipschitz on the compact set $W = \{s\,m\,2^e :\ 2^e \in [\alpha, \beta],\ |m\,2^e| \le M\}$. By continuity of $\nabla\ell$ and compactness of W, the quantity

$$G_{\max} \;=\; \sup_{u \in W} \|\nabla\ell(u)\|$$

is finite. From $2^e \in [\alpha, \beta]$ and $|m\,2^e| \le M$ we also have

$$|m| \;\le\; M/\alpha, \qquad |m'| \;\le\; M/\alpha.$$

For later use we record two auxiliary bounds. First, by the mean-value theorem,

$$|2^e - 2^{e'}| \;=\; (\log 2)\,2^{\xi}\,|e - e'| \;\le\; \beta \log 2\,|e - e'| \quad \text{for some } \xi \text{ between } e \text{ and } e'.$$

Second, writing $w = s\,m\,2^e$ and $w' = s\,m'\,2^{e'}$,

$$\begin{aligned} \|w - w'\| &= s\,\|m\,2^e - m'\,2^{e'}\| \\ &= s\,\|(m - m')\,2^e + m'\,(2^e - 2^{e'})\| \\ &\le s\,\beta\,|m - m'| + s\,|m'|\,|2^e - 2^{e'}| \\ &\le s\,\beta\,|m - m'| + s\,(M/\alpha)\,\beta \log 2\,|e - e'|. \end{aligned} \tag{A}$$

Lipschitz bound for g_m. Since $g_m(e, m) = \partial_m F(e, m) = s\,2^e\,\nabla\ell(w)$, we obtain

$$\begin{aligned} \|g_m(e,m) - g_m(e',m')\| &= s\,\|2^e\,\nabla\ell(w) - 2^{e'}\,\nabla\ell(w')\| \\ &= s\,\|2^e\,[\nabla\ell(w) - \nabla\ell(w')] + (2^e - 2^{e'})\,\nabla\ell(w')\| \\ &\le s\,\beta\,\|\nabla\ell(w) - \nabla\ell(w')\| + s\,|2^e - 2^{e'}|\,\|\nabla\ell(w')\| \\ &\le s\,\beta\,L_w\,\|w - w'\| + s\,\beta \log 2\,G_{\max}\,|e - e'| \\ &\le s\,\beta\,L_w\big[s\,\beta\,|m - m'| + s\,(M/\alpha)\,\beta \log 2\,|e - e'|\big] + s\,\beta \log 2\,G_{\max}\,|e - e'|. \end{aligned}$$

Thus

$$\|g_m(e,m) - g_m(e',m')\| \;\le\; A_m\,|m - m'| + B_m\,|e - e'|,$$

where

$$A_m = s^2\beta^2 L_w, \qquad B_m = s^2\beta^2 L_w\,(M/\alpha) \log 2 \;+\; s\,\beta \log 2\,G_{\max}.$$

Taking $L_m = \max\{A_m, B_m\}$ yields

$$\|g_m(e,m) - g_m(e',m')\| \;\le\; L_m\big(|m - m'| + |e - e'|\big).$$

Lipschitz bound for g_e. Since $g_e(e,m) = \partial_e F(e,m) = s\,m\,(\log 2)\,2^e\,\nabla\ell(w)$, we have

$$\begin{aligned}
\|g_e(e,m) - g_e(e',m')\| &= s\log 2\,\|m\,2^e\,\nabla\ell(w) - m'\,2^{e'}\,\nabla\ell(w')\| \\
&= s\log 2\,\|(m\,2^e - m'\,2^{e'})\,\nabla\ell(w') + m\,2^e\,[\nabla\ell(w) - \nabla\ell(w')]\| \\
&\le s\log 2\,\big(|m\,2^e - m'\,2^{e'}|\,\|\nabla\ell(w')\| + |m|\,2^e\,\|\nabla\ell(w) - \nabla\ell(w')\|\big) \\
&\le s\log 2\,\big((\beta|m-m'| + (M/\alpha)\,\beta\log 2\,|e-e'|)\,G_{\max} \\
&\qquad + (M/\alpha)\,\beta\,L_w\,\|w-w'\|\big) \\
&\le s\log 2\,\big((\beta|m-m'| + (M/\alpha)\,\beta\log 2\,|e-e'|)\,G_{\max} \\
&\qquad + (M/\alpha)\,\beta\,L_w\,[s\,\beta\,|m-m'| + s\,(M/\alpha)\,\beta\log 2\,|e-e'|]\big).
\end{aligned}$$

Hence

$$\|g_e(e,m) - g_e(e',m')\| \;\le\; A_e\,|m-m'| + B_e\,|e-e'|,$$

where

$$A_e = s\log 2\,\big(\beta\,G_{\max} + s\,(M/\alpha)\,\beta^2 L_w\big), \qquad B_e = s\log 2\,\big((M/\alpha)\,\beta\log 2\,G_{\max} + s\,(M/\alpha)^2\,\beta^2\log 2\,L_w\big).$$

Taking $L_e = \max\{A_e, B_e\}$ gives

$$\|g_e(e,m) - g_e(e',m')\| \;\le\; L_e\big(|m-m'| + |e-e'|\big).$$

Lemma 3 (Madam's Layerwise descent decomposition) *Let $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and partition the parameter tensor by layers $\mathbf{W} = (W_1, \ldots, W_L)$ with $g_k(\mathbf{W}) := \nabla_{W_k}\mathcal{L}(\mathbf{W})$. For any perturbation $\Delta\mathbf{W} = (\Delta W_1, \ldots, \Delta W_L)$, let θ_k be the Frobenius-angle between ΔW_k and the negative layer gradient $-g_k(\mathbf{W})$:*

$$\cos\theta_k = \frac{\langle \Delta W_k,\, -g_k(\mathbf{W})\rangle}{\|\Delta W_k\|_F\;\|g_k(\mathbf{W})\|_F} \in [-1,1].$$

Then

$$\mathcal{L}(\mathbf{W}+\Delta\mathbf{W}) - \mathcal{L}(\mathbf{W}) \;\le\; -\sum_{k=1}^{L} \|g_k(\mathbf{W})\|_F\;\|\Delta W_k\|_F \left[\cos\theta_k - \max_{t\in[0,1]} \frac{\|\,g_k(\mathbf{W}+t\Delta\mathbf{W}) - g_k(\mathbf{W})\,\|_F}{\|g_k(\mathbf{W})\|_F}\right]. \tag{6}$$

Proof Define the straight path $\mathbf{W}(t) = \mathbf{W} + t\,\Delta\mathbf{W}$, $t \in [0,1]$. By the fundamental theorem of calculus and the chain rule,

$$\begin{aligned}
\mathcal{L}(\mathbf{W}+\Delta\mathbf{W}) - \mathcal{L}(\mathbf{W}) &= \int_0^1 \langle \nabla\mathcal{L}(\mathbf{W}(t)),\, \Delta\mathbf{W}\rangle\; dt = \sum_{k=1}^{L} \int_0^1 \langle g_k(\mathbf{W}(t)),\, \Delta W_k\rangle\; dt \\
&= \sum_{k=1}^{L} \Big(\langle g_k(\mathbf{W}),\, \Delta W_k\rangle + \int_0^1 \langle\, g_k(\mathbf{W}(t)) - g_k(\mathbf{W})\,,\, \Delta W_k\rangle\; dt\Big).
\end{aligned}$$

The first term equals $-\,\|g_k(\mathbf{W})\|_F\;\|\Delta W_k\|_F \cos\theta_k$ by the cosine formula for the Frobenius inner product. For the integral term, Cauchy–Schwarz and a pointwise maximum over $t \in [0,1]$ give

$$\int_0^1 \langle\, g_k(\mathbf{W}(t)) - g_k(\mathbf{W})\,,\, \Delta W_k\rangle\; dt \;\le\; \|\Delta W_k\|_F \max_{t\in[0,1]} \|\,g_k(\mathbf{W}(t)) - g_k(\mathbf{W})\,\|_F\,.$$

Summing over k and factoring $\|g_k(\mathbf{W})\|_F$ inside the bracket yields (6). ■

Under Lemma 2 (Lipschitz-continuous partial gradients), we can show that each of our four update regimes yields a strict decrease in the objective, provided the step-sizes are chosen sufficiently small.

Theorem 4 (Unified Descent) *Let $F : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ be continuously differentiable with block-Lipschitz gradients: there exist $L_m, L_e, L_{me} > 0$ such that, for all (e, m) and $(\tilde{e}, \tilde{m})$,*

$$\|\nabla_m F(e,m) - \nabla_m F(e,\tilde{m})\| \le L_m \|m - \tilde{m}\|, \quad \|\nabla_e F(e,m) - \nabla_e F(\tilde{e},m)\| \le L_e \|e - \tilde{e}\|,$$

and $\|\nabla F(e,m) - \nabla F(\tilde{e},\tilde{m})\| \le L_{me} \|(e - \tilde{e}, m - \tilde{m})\|$. At (e, m) define $w := m \odot 2^e$ and the (stale) weight gradient

$$g_w^{(0)} := \nabla_w F(w).$$

For elementwise $w = m \odot 2^e$, the block directions obtained from any weight gradient g_w are

$$D_m(e,m)[g_w] := g_w \odot 2^e, \qquad D_e(e,m)[g_w] := g_w \odot \big(m\, 2^e \log 2\big),$$

i.e. $\nabla_m F = D_m(e,m)[g_w]$ and $\nabla_e F = D_e(e,m)[g_w]$ by the chain rule. There exists $\eta > 0$ such that for any $\alpha_m, \alpha_e > 0$ with

$$\max\{\alpha_m L_m,\ \alpha_e L_e,\ \alpha_m \alpha_e L_{me}\} < \eta,$$

each scheme below yields descent in the sense that the updated pair (e', m') satisfies $F(e', m') \le F(e, m)$.

(i) ***Simultaneous (stale g_w):***

$$m' = m - \alpha_m\, D_m(e,m)[g_w^{(0)}], \qquad e' = e - \alpha_e\, D_e(e,m)[g_w^{(0)}],$$

then $F(e', m') \le F(e, m)$.

(ii) ***Mantissa-first,* no *recompute (reuse $g_w^{(0)}$):***

$$m' = m - \alpha_m\, D_m(e,m)[g_w^{(0)}], \qquad e' = e - \alpha_e\, D_e(e,m')[g_w^{(0)}],$$

and $F(e, m') \le F(e, m)$ as well as $F(e', m') \le F(e, m')$, hence $F(e', m') \le F(e, m)$.

(iii) ***Exponent-first,* no *recompute (reuse $g_w^{(0)}$):***

$$e' = e - \alpha_e\, D_e(e,m)[g_w^{(0)}], \qquad m' = m - \alpha_m\, D_m(e',m)[g_w^{(0)}],$$

and $F(e', m) \le F(e, m)$ as well as $F(e', m') \le F(e', m)$, hence $F(e', m') \le F(e, m)$.

(iv) ***Alternating two-step with recompute for the second block:***

$$m' = m - \alpha_m\, D_m(e,m)[g_w^{(0)}], \qquad \text{let } g_w^{(1)} := \nabla_w F\big(m' \odot 2^e\big), \qquad e' = e - \alpha_e\, D_e(e,m')[g_w^{(1)}],$$

then $F(e, m') \le F(e, m)$ and $F(e', m') \le F(e, m')$, hence $F(e', m') \le F(e, m)$.

Implementation note. Items (ii)–(iii) correspond to using a single backward pass to obtain $g_w^{(0)}$ and applying it to both blocks via D_m, D_e; item (iv) performs a second backward after the first block update to obtain $g_w^{(1)}$ for the second block.

Proof Write the block gradients

$$g_m(e,m) = \nabla_m F(e,m), \tag{7}$$
$$g_e(e,m) = \nabla_e F(e,m). \tag{8}$$

Assume $m \mapsto F(e,m)$ has L_m-Lipschitz gradient (for each fixed e), $e \mapsto F(e,m)$ has L_e-Lipschitz gradient (for each fixed m), and the full gradient $\nabla F(e,m)$ is L_{me}-Lipschitz.

Scalar versions. We will use the one–dimensional descent lemma: if ϕ has L-Lipschitz derivative, then

$$\phi(x+\Delta) - \phi(x) \;\le\; \phi'(x)\,\Delta + \tfrac{L}{2}\,\Delta^2. \tag{S1}$$

(i) Simultaneous update Consider the simultaneous step

$$\Delta m = -\alpha_m\, g_m(e,m), \tag{9}$$
$$\Delta e = -\alpha_e\, g_e(e,m). \tag{10}$$

By joint L_{me}-smoothness,

$$F(e{+}\Delta e, m{+}\Delta m) - F(e,m) \le g_m(e,m)\,\Delta m + g_e(e,m)\,\Delta e + \tfrac{L_{\mathrm{me}}}{2}\big(\Delta m^2 + \Delta e^2\big). \tag{11}$$

Mantissa contribution. Using (S1) with $L = L_{\mathrm{me}}$ for the m-direction,

$$g_m(e,m)\,\Delta m + \tfrac{L_{\mathrm{me}}}{2}\Delta m^2 \;\le\; -\Big(\alpha_m - \tfrac{L_{\mathrm{me}}}{2}\alpha_m^2\Big)\, g_m(e,m)^2.$$

Exponent contribution via Madam. The exponent substep induces a multiplicative perturbation of the *weights* $w = m\,2^e$:

$$w^+ \;=\; m\,2^{\,e+\Delta e} \;=\; w\cdot 2^{\,\Delta e} \;=\; w\cdot \exp\big((\log 2)\,\Delta e\big),$$

hence *elementwise* $\delta := \frac{w^+ - w}{w} = 2^{\Delta e} - 1$ and $\rho := \|\delta\|$ controls the relative layerwise size in the Madam lemma. For $|t| \le 1$, the scalar bound $|2^t - 1| \le (\log 2)\,|t|\,e^{(\log 2)|t|}$ gives $|2^{\Delta e} - 1| \le C_e\,|\Delta e|$ with $C_e := (\log 2)\,e^{(\log 2)}$. Therefore, at each layer k,

$$\rho_k \;=\; \frac{\|\Delta W_k\|_F}{\|W_k\|_F} \;=\; \|2^{\Delta e_k} - 1\|_F \;\le\; C_e\,\|\Delta e_k\|_F \;=\; C_e\,\alpha_e\,\|g_{e,k}(e,m)\|_F.$$

Choose $\alpha_e > 0$ small enough so that $\prod_{\ell=1}^{L}(1+\rho_\ell) < 1 + \cos\theta_k$ for all k (where θ_k is the angle between ΔW_k and $-\nabla_{W_k}\mathcal{L}$ as in the Madam lemma). Then by the Madam multiplicative descent result,

$$g_e(e,m)\,\Delta e + \tfrac{L_{\mathrm{me}}}{2}\Delta e^2 \;\le\; -\,\rho_e(\alpha_e)\,g_e(e,m)^2,$$

for some $\rho_e(\alpha_e) = c_e\,\alpha_e\,(1 - O(\alpha_e)) \ge 0$ (small α_e).

Combine. Substituting both bounds into (11) yields

$$F(e+\Delta e, m+\Delta m) - F(e,m) \le -\Big(\alpha_m - \tfrac{L_{\mathrm{me}}}{2}\alpha_m^2\Big) g_m(e,m)^2 \;-\; \rho_e(\alpha_e)\, g_e(e,m)^2. \tag{12}$$

Hence $F(e+\Delta e, m+\Delta m) \le F(e,m)$ whenever $\alpha_m \in (0, 2/L_{\mathrm{me}})$ and α_e is chosen small enough to satisfy the Madam condition (3). Strict inequality holds if at least one of g_m, g_e is nonzero.

(ii) Mantissa-first sequential update Set the two substeps

$$m' = m - \alpha_m\, g_m(e,m), \tag{13}$$
$$e' = e - \alpha_e\, g_e(e,m'). \tag{14}$$

Mantissa contribution (Euclidean smoothness). Applying (S1) in the m–direction with $L = L_m$ gives

$$F(e,m') - F(e,m) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big)\, g_m(e,m)^2. \tag{15}$$

Exponent contribution (Madam multiplicative descent, fix m'). Write the elementwise weights $w' = m'\,2^e$. The exponent move induces

$$w' \;\mapsto\; w' \odot 2^{\,e'-e} \;=\; w' \odot \exp\big((\log 2)\,(e'-e)\big),$$

so $\delta := 2^{\,e'-e} - 1$ is the multiplicative change and $\rho_k := \|\Delta W_k'\|_F / \|W_k'\|_F = \|2^{\,e'-e} - 1\|_F$ is the relative size at layer k. Using $|2^t - 1| \le (\log 2)\,|t|\, e^{(\log 2)|t|} \le C_e|t|$ for $|t| \le 1$ with $C_e := (\log 2) e^{(\log 2)}$, we obtain

$$\rho_k \;\le\; C_e\, \|e' - e\|_{F,k} \;=\; C_e\, \alpha_e\, \|g_{e,k}(e,m')\|_F. \tag{16}$$

Let θ_k be the angle between $\Delta W_k'$ and $-\nabla_{W_k}\mathcal{L}$, and define $\eta^\star(\theta) := (1+\cos\theta)^{1/L} - 1$. A sufficient global stepsize condition ensuring the Madam descent condition at all layers is

$$C_e\, \alpha_e\, \max_k \|g_{e,k}(e,m')\|_F \;\le\; \eta^\star(\bar\theta'), \qquad \bar\theta' := \max_k \theta_k. \tag{17}$$

Under (17), Lemma 3 yields

$$F(e',m') - F(e,m') \;\le\; -\,\rho_e(\alpha_e)\, g_e(e,m')^2, \qquad \rho_e(\alpha_e) = c_e\,\alpha_e\,(1 - O(\alpha_e)) > 0 \text{ for small } \alpha_e. \tag{18}$$

Combine. Adding (15) and (18),

$$F(e',m') - F(e,m) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big)\, g_m(e,m)^2 \;-\; \rho_e(\alpha_e)\, g_e(e,m')^2. \tag{19}$$

Therefore $F(e',m') \le F(e,m)$ whenever $\alpha_m \in (0, 2/L_m)$ and α_e satisfies (17); strict if at least one of $g_m(e,m), g_e(e,m')$ is nonzero.

(iii) Exponent-first sequential update Set

$$e' = e - \alpha_e \, g_e(e, m), \tag{20}$$
$$m' = m - \alpha_m \, g_m(e', m). \tag{21}$$

Exponent contribution (Madam multiplicative descent, fix m). With $w = m\,2^e$, the exponent move is $w \mapsto w \odot 2^{\,e'-e}$, so $\rho_k = \|2^{\,e'-e} - 1\|_F \le C_e\alpha_e \|g_{e,k}(e,m)\|_F$ as above. Let θ_k be the angle between ΔW_k and $-\nabla_{W_k}\mathcal{L}$ and define $\eta^\star$ as before. A sufficient stepsize condition is

$$C_e \, \alpha_e \, \max_k \|g_{e,k}(e,m)\|_F \;\le\; \eta^\star(\bar\theta), \qquad \bar\theta := \max_k \theta_k. \tag{22}$$

Under (22), Lemma 3 implies

$$F(e', m) - F(e, m) \;\le\; -\,\rho_e(\alpha_e)\, g_e(e,m)^2, \qquad \rho_e(\alpha_e) = c_e\,\alpha_e\,(1 - O(\alpha_e)) > 0. \tag{23}$$

Mantissa contribution (Euclidean smoothness, fix e'). Applying (S1) at (e', m) with $L = L_m$,

$$F(e', m') - F(e', m) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big)\, g_m(e', m)^2. \tag{24}$$

Combine. Summing (23) and (24),

$$F(e', m') - F(e, m) \;\le\; -\,\rho_e(\alpha_e)\, g_e(e,m)^2 \;-\; \Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big)\, g_m(e', m)^2. \tag{25}$$

Thus $F(e', m') \le F(e, m)$ for α_e obeying (22) and $\alpha_m \in (0, 2/L_m)$, provided the respective block gradients at their evaluation points are nonzero.

(iv) Alternating two-step update Set

$$m' = m - \alpha_m \, g_m(e, m), \tag{26}$$
$$e'' = e - \alpha_e \, g_e(e, m'). \tag{27}$$

First substep (mantissa, Euclidean smoothness).

$$F(e, m') - F(e, m) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big)\, g_m(e, m)^2. \tag{28}$$

Second substep (exponent via Madam, fix m').
As in (16), the relative sizes satisfy $\rho_k \le C_e\,\alpha_e\,\|g_{e,k}(e,m')\|_F$. A sufficient stepsize condition is

$$C_e \, \alpha_e \, \max_k \|g_{e,k}(e,m')\|_F \;\le\; \eta^\star(\bar\theta''), \qquad \bar\theta'' := \max_k \theta_k, \tag{29}$$

which ensures the Madam condition at all layers for this substep. Then

$$F(e'', m') - F(e, m') \;\le\; -\,\rho_e(\alpha_e)\, g_e(e,m')^2, \qquad \rho_e(\alpha_e) = c_e\,\alpha_e\,(1 - O(\alpha_e)) > 0. \tag{30}$$

Combine. From (28)–(30),

$$F(e'', m') - F(e, m) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big)\, g_m(e, m)^2 \;-\; \rho_e(\alpha_e)\, g_e(e, m')^2. \tag{31}$$

Therefore $F(e'', m') \le F(e, m)$ whenever $\alpha_m \in (0, 2/L_m)$ and α_e satisfies (29); strict if the corresponding block gradients are nonzero.

Vector/matrix versions. Let $\boldsymbol{m} \in \mathbb{R}^p$ and $\boldsymbol{e} \in \mathbb{R}^q$, and write

$$\boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m}) = \nabla_{\boldsymbol{m}} F(\boldsymbol{e}, \boldsymbol{m}), \tag{32}$$

$$\boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m}) = \nabla_{\boldsymbol{e}} F(\boldsymbol{e}, \boldsymbol{m}). \tag{33}$$

If the parameterization uses elementwise structure (e.g. $\boldsymbol{w} = \boldsymbol{m} \odot 2^{\boldsymbol{e}}$), all componentwise multiplications are written with $\odot$ and the power $2^{\boldsymbol{e}}$ is taken elementwise. Block-smoothness yields, for any increments $\Delta\boldsymbol{m}, \Delta\boldsymbol{e}$,

$$F(\boldsymbol{e}, \boldsymbol{m} + \Delta\boldsymbol{m}) \le F(\boldsymbol{e}, \boldsymbol{m}) + \langle \boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m})\rangle \Delta\boldsymbol{m} + \tfrac{L_m}{2}\, \|\Delta\boldsymbol{m}\|^2 \tag{V1}$$

$$F(\boldsymbol{e} + \Delta\boldsymbol{e}, \boldsymbol{m}) \le F(\boldsymbol{e}, \boldsymbol{m}) + \langle \boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m})\rangle \Delta\boldsymbol{e} + \tfrac{L_e}{2}\, \|\Delta\boldsymbol{e}\|^2 \tag{V2}$$

$$F(\boldsymbol{e} + \Delta\boldsymbol{e}, \boldsymbol{m} + \Delta\boldsymbol{m}) \le F(\boldsymbol{e}, \boldsymbol{m}) + \langle \nabla F(\boldsymbol{e}, \boldsymbol{m})\rangle (\Delta\boldsymbol{e}, \Delta\boldsymbol{m}) + \tfrac{L_{\mathrm{me}}}{2}\big(\|\Delta\boldsymbol{m}\|^2 + \|\Delta\boldsymbol{e}\|^2\big). \tag{V3}$$

(i) Simultaneous update Set

$$\Delta\boldsymbol{m} = -\alpha_m\, \boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m}), \tag{34}$$

$$\Delta\boldsymbol{e} = -\alpha_e\, \boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m}). \tag{35}$$

Using (V3),

$$F(\boldsymbol{e}{+}\Delta\boldsymbol{e}, \boldsymbol{m}{+}\Delta\boldsymbol{m}) - F(\boldsymbol{e}, \boldsymbol{m}) \le \langle \boldsymbol{g}_m\rangle \Delta\boldsymbol{m} + \langle \boldsymbol{g}_e\rangle \Delta\boldsymbol{e} + \tfrac{L_{\mathrm{me}}}{2}\big(\|\Delta\boldsymbol{m}\|^2 + \|\Delta\boldsymbol{e}\|^2\big). \tag{36}$$

Mantissa contribution (Euclidean smoothness).

$$\langle \boldsymbol{g}_m\rangle \Delta\boldsymbol{m} + \tfrac{L_{\mathrm{me}}}{2}\|\Delta\boldsymbol{m}\|^2 \;\le\; -\Big(\alpha_m - \tfrac{L_{\mathrm{me}}}{2}\alpha_m^2\Big)\, \|\boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m})\|^2. \tag{37}$$

Exponent contribution (Madam multiplicative descent). For each layer k, the exponent move produces the multiplicative perturbation $W_k \mapsto W_k \odot 2^{\,\Delta \boldsymbol{e}_k}$. Let θ_k be the angle between ΔW_k and $-\nabla_{W_k}\mathcal{L}$ (as in Lemma 3), and let $\rho_k := \|2^{\,\Delta \boldsymbol{e}_k} - 1\|_F$ denote the relative size. Using $|2^t - 1| \le (\log 2)\, |t|\, e^{(\log 2)|t|} \le C_e |t|$ for $|t| \le 1$ with $C_e := (\log 2) e^{(\log 2)}$,

$$\rho_k \;\le\; C_e\, \|\Delta \boldsymbol{e}_k\|_F \;=\; C_e\, \alpha_e\, \|\boldsymbol{g}_{e,k}(\boldsymbol{e}, \boldsymbol{m})\|_F. \tag{38}$$

A sufficient global condition to ensure the Madam descent condition for all layers is

$$C_e\, \alpha_e\, \max_k \|\boldsymbol{g}_{e,k}(\boldsymbol{e}, \boldsymbol{m})\|_F \;\le\; (1 + \cos\bar{\theta})^{1/L} - 1, \qquad \bar{\theta} := \max_k \theta_k. \tag{39}$$

Under (39), Lemma 3 yields

$$\langle \boldsymbol{g}_e\rangle \Delta\boldsymbol{e} + \tfrac{L_{\mathrm{me}}}{2}\|\Delta\boldsymbol{e}\|^2 \;\le\; -\,\rho_e(\alpha_e)\, \|\boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m})\|^2, \qquad \rho_e(\alpha_e) = c_e\, \alpha_e\, (1 - O(\alpha_e)) > 0. \tag{40}$$

Combine. Substituting (37) and (40) into (36),

$$F(\boldsymbol{e}{+}\Delta\boldsymbol{e}, \boldsymbol{m}{+}\Delta\boldsymbol{m}) - F(\boldsymbol{e}, \boldsymbol{m}) \le -\Big(\alpha_m - \tfrac{L_{\mathrm{me}}}{2}\alpha_m^2\Big)\, \|\boldsymbol{g}_m\|^2 \;-\; \rho_e(\alpha_e)\, \|\boldsymbol{g}_e\|^2. \tag{41}$$

Thus $F(\boldsymbol{e}{+}\Delta\boldsymbol{e}, \boldsymbol{m}{+}\Delta\boldsymbol{m}) \le F(\boldsymbol{e}, \boldsymbol{m})$ whenever $\alpha_m \in (0, 2/L_{\mathrm{me}})$ and α_e satisfies (39), with strict descent if $(\boldsymbol{g}_m, \boldsymbol{g}_e) \neq (\boldsymbol{0}, \boldsymbol{0})$.

(ii) Mantissa-first sequential update Set

$$\boldsymbol{m}' = \boldsymbol{m} - \alpha_m \, \boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m}), \tag{42}$$
$$\boldsymbol{e}' = \boldsymbol{e} - \alpha_e \, \boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m}'). \tag{43}$$

Mantissa contribution (Euclidean smoothness). By (V1),

$$F(\boldsymbol{e}, \boldsymbol{m}') - F(\boldsymbol{e}, \boldsymbol{m}) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big) \, \|\boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m})\|^2. \tag{44}$$

Exponent contribution (Madam, fix $\boldsymbol{m}'$). As above, the step $\boldsymbol{e} \mapsto \boldsymbol{e}'$ induces $W_k \mapsto W_k \odot 2^{\boldsymbol{e}'-\boldsymbol{e}}$. With $\rho_k \le C_e \alpha_e \|\boldsymbol{g}_{e,k}(\boldsymbol{e}, \boldsymbol{m}')\|_F$, a sufficient condition is

$$C_e \, \alpha_e \, \max_k \|\boldsymbol{g}_{e,k}(\boldsymbol{e}, \boldsymbol{m}')\|_F \;\le\; (1 + \cos\bar{\theta}')^{1/L} - 1, \qquad \bar{\theta}' := \max_k \theta_k, \tag{45}$$

which guarantees (by Lemma 3)

$$F(\boldsymbol{e}', \boldsymbol{m}') - F(\boldsymbol{e}, \boldsymbol{m}') \;\le\; -\rho_e(\alpha_e) \, \|\boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m}')\|^2. \tag{46}$$

Combine. Adding (44) and (46),

$$F(\boldsymbol{e}', \boldsymbol{m}') - F(\boldsymbol{e}, \boldsymbol{m}) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big) \, \|\boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m})\|^2 \;-\; \rho_e(\alpha_e) \, \|\boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m}')\|^2.$$

Hence $F(\boldsymbol{e}', \boldsymbol{m}') \le F(\boldsymbol{e}, \boldsymbol{m})$ whenever $\alpha_m \in (0, 2/L_m)$ and α_e satisfies (45).

(iii) Exponent-first sequential update Set

$$\boldsymbol{e}' = \boldsymbol{e} - \alpha_e \, \boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m}), \tag{47}$$
$$\boldsymbol{m}' = \boldsymbol{m} - \alpha_m \, \boldsymbol{g}_m(\boldsymbol{e}', \boldsymbol{m}). \tag{48}$$

Exponent contribution (Madam, fix $\boldsymbol{m}$). With $\rho_k \le C_e \alpha_e \|\boldsymbol{g}_{e,k}(\boldsymbol{e}, \boldsymbol{m})\|_F$, a sufficient condition is

$$C_e \, \alpha_e \, \max_k \|\boldsymbol{g}_{e,k}(\boldsymbol{e}, \boldsymbol{m})\|_F \;\le\; (1 + \cos\bar{\theta})^{1/L} - 1, \qquad \bar{\theta} := \max_k \theta_k, \tag{49}$$

under which

$$F(\boldsymbol{e}', \boldsymbol{m}) - F(\boldsymbol{e}, \boldsymbol{m}) \;\le\; -\rho_e(\alpha_e) \, \|\boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m})\|^2. \tag{50}$$

Mantissa contribution (Euclidean smoothness, fix $\boldsymbol{e}'$). By (V1),

$$F(\boldsymbol{e}', \boldsymbol{m}') - F(\boldsymbol{e}', \boldsymbol{m}) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big) \, \|\boldsymbol{g}_m(\boldsymbol{e}', \boldsymbol{m})\|^2. \tag{51}$$

Combine. Summing (50) and (51),

$$F(\boldsymbol{e}', \boldsymbol{m}') - F(\boldsymbol{e}, \boldsymbol{m}) \;\le\; -\rho_e(\alpha_e) \, \|\boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m})\|^2 \;-\; \Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big) \, \|\boldsymbol{g}_m(\boldsymbol{e}', \boldsymbol{m})\|^2.$$

Thus $F(\boldsymbol{e}', \boldsymbol{m}') \le F(\boldsymbol{e}, \boldsymbol{m})$ whenever $\alpha_m \in (0, 2/L_m)$ and α_e satisfies (49).

(iv) Alternating two-step update Set

$$\boldsymbol{m}' = \boldsymbol{m} - \alpha_m \, \boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m}), \tag{52}$$

$$\boldsymbol{e}'' = \boldsymbol{e} - \alpha_e \, \boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m}'). \tag{53}$$

First substep (mantissa, Euclidean smoothness). By (V1),

$$F(\boldsymbol{e}, \boldsymbol{m}') - F(\boldsymbol{e}, \boldsymbol{m}) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big) \, \|\boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m})\|^2. \tag{54}$$

Second substep (exponent via Madam, fix $\boldsymbol{m}'$). With $\rho_k \le C_e \alpha_e \|\boldsymbol{g}_{e,k}(\boldsymbol{e}, \boldsymbol{m}')\|_F$, a sufficient condition is

$$C_e \, \alpha_e \, \max_k \|\boldsymbol{g}_{e,k}(\boldsymbol{e}, \boldsymbol{m}')\|_F \;\le\; (1 + \cos\bar{\theta}'')^{1/L} - 1, \qquad \bar{\theta}'' := \max_k \theta_k, \tag{55}$$

which guarantees

$$F(\boldsymbol{e}'', \boldsymbol{m}') - F(\boldsymbol{e}, \boldsymbol{m}') \;\le\; -\,\rho_e(\alpha_e) \, \|\boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m}')\|^2. \tag{56}$$

Combine. From (54)–(56),

$$F(\boldsymbol{e}'', \boldsymbol{m}') - F(\boldsymbol{e}, \boldsymbol{m}) \;\le\; -\Big(\alpha_m - \tfrac{L_m}{2}\alpha_m^2\Big) \, \|\boldsymbol{g}_m(\boldsymbol{e}, \boldsymbol{m})\|^2 \;-\; \rho_e(\alpha_e) \, \|\boldsymbol{g}_e(\boldsymbol{e}, \boldsymbol{m}')\|^2.$$

Therefore $F(\boldsymbol{e}'', \boldsymbol{m}') \le F(\boldsymbol{e}, \boldsymbol{m})$ whenever $\alpha_m \in (0, 2/L_m)$ and α_e satisfies (55); strict if the corresponding block gradients are nonzero.

■

Appendix C. Algorithmic Variants for the M+Adam Optimizer

C.1. Subroutines: AdamMantissa and MadamExponent

Algorithm 2: Subroutines shared by all strategies (state $\mu_m, \nu_m, \mu_e, \nu_e$ is persistent)

Input: time step t; current (e, m); gradients (g_m, g_e)

AdamMantissa(m, g_m, t) $\mu_m \leftarrow \beta_1 \mu_m + (1-\beta_1) g_m$;
$\nu_m \leftarrow \beta_2 \nu_m + (1-\beta_2) g_m^2$;
$\hat{\mu}_m \leftarrow \mu_m / (1-\beta_1^t)$; $\hat{\nu}_m \leftarrow \nu_m / (1-\beta_2^t)$;
$\Delta m \leftarrow \hat{\mu}_m / (\sqrt{\hat{\nu}_m} + \epsilon)$;
$m \leftarrow m - \eta_m \operatorname{clamp}_{\eta_m^\star / \eta_m}(\Delta m)$
return m

MadamExponent(e, g_e, t) $\mu_e \leftarrow \beta_1 \mu_e + (1-\beta_1) g_e$;
$\nu_e \leftarrow \beta_2 \nu_e + (1-\beta_2) g_e^2$;
$\hat{\mu}_e \leftarrow \mu_e / (1-\beta_1^t)$; $\hat{\nu}_e \leftarrow \nu_e / (1-\beta_2^t)$;
$\Delta e \leftarrow \hat{\mu}_e / (\sqrt{\hat{\nu}_e} + \epsilon)$;
$e \leftarrow e - \eta_e \operatorname{clamp}_{\eta_e^\star / \eta_e}(\Delta e)$ ▷ $w \propto 2^{-\eta_e \operatorname{clamp}(\Delta e)}$ (multiplicative effect)
return e

C.2. Schedules for non-parallel strategies

Let $\pi(\text{strategy}, t)$ be the sequence of substeps at iteration t, where M denotes a call to AdamMantissa and E denotes a call to MadamExponent. As in the main text, g_m and g_e are computed *once per iteration* from (e, m) before any substep.

Strategy	$\pi(\text{strategy}, t)$
MantissaFirst	[M, E]
ExponentFirst	[E, M]
Alternating	[M] if t even; [E] if t odd

Table 2: Schedules for the three non-parallel M+Adam variants.

Notes. (1) With shared gradients per iteration, MantissaFirst and ExponentFirst differ only by substep order; numerical differences are at roundoff level.
(2) Alternating halves the frequency of each sub-update and is useful for stress-testing stability.
(3) A "recompute" mode may refresh g_m or g_e after the first substep to make the first two schedules behaviorally distinct; our experiments use the shared-gradient default.

C.3. Wrapper for the three non-parallel variants

Algorithm 3: Schedule-driven wrapper for M+Adam variants (reuses Alg. 2)

Input: w, step t, strategy $\in$ {MantissaFirst, ExponentFirst, Alternating}; STOCHASTICGRADIENT()
Output: updated w

$g \leftarrow$ STOCHASTICGRADIENT() ▷ gradient in weight space
$(e, m) \leftarrow (w)$
$g_m \leftarrow 2^e\, g$; $\quad g_e \leftarrow (w \ln 2)\, g$
foreach $s \in \pi(\textit{strategy}, t)$ **do**
  if $s = M$ **then**
    $m \leftarrow$ **AdamMantissa**(m, g_m, t)
  else
    $e \leftarrow$ **MadamExponent**(e, g_e, t)
  end
end
$w \leftarrow \text{clamp}_{w_{\max}}\big((e, m)\big)$ ▷ clamp weights between $\pm w_{\max}$
return w

C.4. Variant-specific descriptions (delta to Simultaneous)

- **MantissaFirst**: sequentially apply M then E within the iteration (Table 2). State and hyperparameters match Alg. 1.
- **ExponentFirst**: sequentially apply E then M. Identical state/hyperparameters; only the order differs.
- **Alternating**: apply exactly one substep per iteration, alternating by the parity of t; this reduces the effective update frequency per block by $1/2$ and often exposes stability differences (see ablations).

Appendix D. Model architectures

Model	Params	Seq Len	Hidden Dim	Inter Dim	# Layers	# Heads
LLaMA-60M	60M	512	512	1376	8	8
LLaMA-130M	130M	512	768	2048	12	12
LLaMA-350M	350M	512	1024	2736	24	16

Table 3: Detailed architecture hyperparameters for each model size we studied.

Appendix E. Hyperparameter Ablation

We reported the results for the optimizers we swept main text. The result is formulated as follows: the first row shows the approximately best configuration found and the following rows show the results for the 1-dimensional ablations centered around the found configuration. The loss presented here is the final loss on the C4/EN validation set. We employ two different schemes: parameters in bf16 and activations in bf16 and parameters in fp32 and activations in fp32.

Table 4: Hyperparameter ablation for M+Adam on LLaMA-60M ($1\times$ Chinchilla data). Parameters in bf16 and activations in bf16.

ϵ	η_m	η_e	$warmup_m$	$warmup_e$	α_e	λ_m	λ_e	EXP schedule	β_1	β_2	Loss
1e-9	0.0040	1.2e-4	800	800	3.0	0.100	0.010	logcosine	0.87	0.99	3.343
1e-8	—	—	—	—	—	—	—	—	—	—	3.347
1e-10	—	—	—	—	—	—	—	—	—	—	3.347
—	0.0036	—	—	—	—	—	—	—	—	—	3.349
—	0.0046	—	—	—	—	—	—	—	—	—	3.347
—	—	7.5e-5	—	—	—	—	—	—	—	—	3.347
—	—	1.9e-4	—	—	—	—	—	—	—	—	3.347
—	—	—	600	—	—	—	—	—	—	—	3.350
—	—	—	1200	—	—	—	—	—	—	—	3.347
—	—	—	—	600	—	—	—	—	—	—	3.346
—	—	—	—	1200	—	—	—	—	—	—	3.348
—	—	—	—	—	2.0	—	—	—	—	—	3.347
—	—	—	—	—	4.0	—	—	—	—	—	3.347
—	—	—	—	—	—	0.050	—	—	—	—	3.350
—	—	—	—	—	—	0.150	—	—	—	—	3.350
—	—	—	—	—	—	—	0.000	—	—	—	3.347
—	—	—	—	—	—	—	0.020	—	—	—	3.348
—	—	—	—	—	—	—	—	cosine	—	—	3.347
—	—	—	—	—	—	—	—	linear	—	—	3.347
—	—	—	—	—	—	—	—	—	0.85	—	3.347
—	—	—	—	—	—	—	—	—	0.92	—	3.362
—	—	—	—	—	—	—	—	—	—	0.96	3.350
—	—	—	—	—	—	—	—	—	—	0.97	3.343

Table 5: Hyperparameter ablation for M+Adam on LLaMA-130M (1× Chinchilla data). Parameters in bf16 and activations in bf16.

ϵ	η_m	η_e	$warmup_m$	$warmup_e$	α_e	λ_m	λ_e	EXP schedule	β_1	β_2	Loss
1e-9	0.0040	1.2e-4	800	800	3.0	0.100	0.010	logcosine	0.87	0.99	3.059
1e-8	—	—	—	—	—	—	—	—	—	—	3.060
1e-10	—	—	—	—	—	—	—	—	—	—	3.060
—	0.0036	—	—	—	—	—	—	—	—	—	3.063
—	0.0046	—	—	—	—	—	—	—	—	—	3.061
—	—	7.5e-5	—	—	—	—	—	—	—	—	3.059
—	—	1.9e-4	—	—	—	—	—	—	—	—	3.062
—	—	—	600	—	—	—	—	—	—	—	3.064
—	—	—	1200	—	—	—	—	—	—	—	3.058
—	—	—	—	600	—	—	—	—	—	—	3.060
—	—	—	—	1200	—	—	—	—	—	—	3.060
—	—	—	—	—	2.0	—	—	—	—	—	3.059
—	—	—	—	—	4.0	—	—	—	—	—	3.059
—	—	—	—	—	—	0.050	—	—	—	—	3.066
—	—	—	—	—	—	0.150	—	—	—	—	3.064
—	—	—	—	—	—	—	0.000	—	—	—	3.058
—	—	—	—	—	—	—	0.020	—	—	—	3.058
—	—	—	—	—	—	—	—	cosine	—	—	3.059
—	—	—	—	—	—	—	—	linear	—	—	3.059
—	—	—	—	—	—	—	—	—	0.85	—	3.062
—	—	—	—	—	—	—	—	—	0.92	—	3.080
—	—	—	—	—	—	—	—	—	—	0.96	3.071
—	—	—	—	—	—	—	—	—	—	0.97	3.065

Table 6: Hyperparameter ablation for M+Adam on LLaMA-350M (1× Chinchilla data). Parameters in bf16 and activations in bf16.

ϵ	η_m	η_e	$warmup_m$	$warmup_e$	α_e	λ_m	λ_e	`EXP schedule`	β_1	β_2	Loss
1e-9	0.0036	1.2e-4	5000	5000	3.0	0.100	0.010	logcosine	0.87	0.99	2.771
1e-8	—	—	—	—	—	—	—	—	—	—	2.776
1e-10	—	—	—	—	—	—	—	—	—	—	2.785
—	0.004	—	—	—	—	—	—	—	—	—	2.778
—	0.0046	—	—	—	—	—	—	—	—	—	2.791
—	—	7.5e-5	—	—	—	—	—	—	—	—	2.781
—	—	1.9e-4	—	—	—	—	—	—	—	—	2.776
—	—	—	4000	—	—	—	—	—	—	—	2.779
—	—	—	6000	—	—	—	—	—	—	—	2.775
—	—	—	—	4000	—	—	—	—	—	—	2.774
—	—	—	—	6000	—	—	—	—	—	—	2.782
—	—	—	—	—	2.0	—	—	—	—	—	2.777
—	—	—	—	—	4.0	—	—	—	—	—	2.780
—	—	—	—	—	—	0.050	—	—	—	—	3.213
—	—	—	—	—	—	0.150	—	—	—	—	2.773
—	—	—	—	—	—	—	0.000	—	—	—	2.780
—	—	—	—	—	—	—	0.020	—	—	—	2.780
—	—	—	—	—	—	—	—	cosine	—	—	2.779
—	—	—	—	—	—	—	—	linear	—	—	2.778
—	—	—	—	—	—	—	—	—	0.85	—	2.772
—	—	—	—	—	—	—	—	—	0.92	—	2.803
—	—	—	—	—	—	—	—	—	—	0.96	2.779
—	—	—	—	—	—	—	—	—	—	0.97	2.778

Table 7: Hyperparameter ablation for AdamW on 60M (1× Chinchilla Data). Parameters in bf16 and activations in bf16.

η	β_1	β_2	ϵ	warmup	max_grad_norm	weight_decay	Final loss
0.0050	0.90	0.98	1e-9	1000	1.0	0.10	3.399
0.0035	—	—	—	—	—	—	3.413
0.0075	—	—	—	—	—	—	3.467
—	0.95	—	—	—	—	—	3.443
—	0.98	—	—	—	—	—	3.643
—	—	0.95	—	—	—	—	3.455
—	—	0.999	—	—	—	—	3.456
—	—	—	1e-10	—	—	—	3.444
—	—	—	1e-8	—	—	—	3.441
—	—	—	—	500	—	—	3.502
—	—	—	—	2000	—	—	3.410
—	—	—	—	—	0.0	—	3.438
—	—	—	—	—	2.0	—	3.442
—	—	—	—	—	—	0.00	3.443
—	—	—	—	—	—	0.20	3.443

Table 8: Hyperparameter ablation for AdamW SR on 60M (1× Chinchilla Data). Parameters in bf16 and activations in bf16.

η	β_1	β_2	ϵ	warmup	max_grad_norm	weight_decay	Final loss
0.0050	0.95	0.98	1e-8	2000	1.0	0.10	3.401
0.0035	—	—	—	—	—	—	3.420
0.0075	—	—	—	—	—	—	3.448
—	0.90	—	—	—	—	—	3.403
—	0.98	—	—	—	—	—	3.547
—	—	0.95	—	—	—	—	3.445
—	—	0.999	—	—	—	—	3.419
—	—	—	1e-10	—	—	—	3.426
—	—	—	1e-9	—	—	—	3.425
—	—	—	—	500	—	—	3.470
—	—	—	—	1000	—	—	3.425
—	—	—	—	—	0.0	—	3.429
—	—	—	—	—	2.0	—	3.427
—	—	—	—	—	—	0.00	3.425
—	—	—	—	—	—	0.20	3.425

Table 9: Hyperparameter ablation for Adam with Stochastic Rounding on 60M (1× Chinchilla Data). Parameters in bf16 and activations in bf16.

η	β_1	β_2	ϵ	warmup	max_grad_norm	weight_decay	Final loss
0.0050	0.90	0.95	1e-8	1000	1.0	0.10	3.365
0.0035	—	—	—	—	—	—	3.385
0.0075	—	—	—	—	—	—	3.389
—	0.95	—	—	—	—	—	3.383
—	0.98	—	—	—	—	—	3.458
—	—	0.98	—	—	—	—	3.383
—	—	0.999	—	—	—	—	3.384
—	—	—	1e-10	—	—	—	3.381
—	—	—	1e-9	—	—	—	3.383
—	—	—	—	500	—	—	3.409
—	—	—	—	2000	—	—	3.369
—	—	—	—	—	0.0	—	3.384
—	—	—	—	—	2.0	—	3.384
—	—	—	—	—	—	0.00	3.425
—	—	—	—	—	—	0.20	3.383

Table 10: Hyperparameter ablation for Adam on 60M (1× Chinchilla Data). Parameters in bf16 and activations in bf16.

η	β_1	β_2	ϵ	warmup	max_grad_norm	Final loss
0.0050	0.90	0.95	1e-8	2000	1.0	3.410
0.0035	—	—	—	—	—	3.413
0.0075	—	—	—	—	—	3.467
—	0.95	—	—	—	—	3.443
—	0.98	—	—	—	—	3.643
—	—	0.98	—	—	—	3.443
—	—	0.999	—	—	—	3.456
—	—	—	1e-10	—	—	3.444
—	—	—	1e-9	—	—	3.443
—	—	—	—	500	—	3.502
—	—	—	—	1000	—	3.443
—	—	—	—	—	0.0	3.438
—	—	—	—	—	2.0	3.442

Table 11: Hyperparameter ablation for AdamW on 60M (1× Chinchilla Data). Parameters in fp32 and activations in fp32.

η	β_1	β_2	ϵ	warmup	max_grad_norm	weight_decay	Final loss
0.0050	0.90	0.95	1e-9	2000	1.0	0.10	3.333
0.0035	—	—	—	—	—	—	3.340
0.0075	—	—	—	—	—	—	3.358
—	0.95	—	—	—	—	—	3.344
—	0.98	—	—	—	—	—	3.426
—	—	0.98	—	—	—	—	3.344
—	—	0.999	—	—	—	—	3.349
—	—	—	1e-10	—	—	—	3.345
—	—	—	1e-8	—	—	—	3.345
—	—	—	—	500	—	—	3.371
—	—	—	—	1000	—	—	3.344
—	—	—	—	—	0.0	—	3.348
—	—	—	—	—	2.0	—	3.348
—	—	—	—	—	—	0.00	3.367
—	—	—	—	—	—	0.20	3.351

Table 12: Hyperparameter ablation for Adam on 60M (1× Chinchilla Data). Parameters in fp32 and activations in fp32.

η	β_1	β_2	ϵ	warmup	max_grad_norm	Final loss
0.0050	0.90	0.95	1e-8	2000	1.0	3.346
0.0035	—	—	—	—	—	3.358
0.0075	—	—	—	—	—	3.395
—	0.95	—	—	—	—	3.368
—	0.98	—	—	—	—	3.493
—	—	0.98	—	—	—	3.368
—	—	0.999	—	—	—	3.367
—	—	—	1e-10	—	—	3.370
—	—	—	1e-9	—	—	3.368
—	—	—	—	500	—	3.414
—	—	—	—	1000	—	3.368
—	—	—	—	—	0.0	3.374
—	—	—	—	—	2.0	3.348

Table 13: Hyperparameter ablation for AdamW on 130M (1× Chinchilla Data). Parameters in fp32 and activations in fp32.

η	β_1	β_2	ϵ	warmup	max_grad_norm	weight_decay	Final loss
0.0035	0.90	0.95	1e-9	2000	1.0	0.10	3.042
0.0050	—	—	—	—	—	—	3.091
0.0075	—	—	—	—	—	—	4.992
—	0.95	—	—	—	—	—	3.159
—	0.98	—	—	—	—	—	3.134
—	—	0.98	—	—	—	—	3.098
—	—	0.999	—	—	—	—	3.222
—	—	—	1e-10	—	—	—	3.091
—	—	—	1e-8	—	—	—	3.090
—	—	—	—	500	—	—	3.137
—	—	—	—	1000	—	—	3.159
—	—	—	—	—	0.0	—	3.092
—	—	—	—	—	2.0	—	3.097
—	—	—	—	—	—	0.00	3.134
—	—	—	—	—	—	0.20	3.099

Table 14: Hyperparameter ablation for Adam on 130M (1× Chinchilla Data). Parameters in fp32 and activations in fp32.

η	β_1	β_2	ϵ	warmup	max_grad_norm	Final loss
0.0050	0.95	0.98	1e-9	2000	1.0	3.082
0.0035	—	—	—	—	—	3.106
0.0075	—	—	—	—	—	5.020
—	0.90	—	—	—	—	4.753
—	0.98	—	—	—	—	4.732
—	—	0.95	—	—	—	3.138
—	—	0.999	—	—	—	5.408
—	—	—	1e-10	—	—	3.136
—	—	—	1e-8	—	—	3.134
—	—	—	—	500	—	4.627
—	—	—	—	1000	—	3.135
—	—	—	—	—	0.0	3.138
—	—	—	—	—	2.0	3.144

Table 15: Hyperparameter ablation for AdamW on 130M (1× Chinchilla Data). Parameters in bf16 and activations in bf16.

η	β_1	β_2	ϵ	warmup	max_grad_norm	weight_decay	Final loss
0.0035	0.95	0.98	1e-9	1000	1.0	0.10	3.238
0.0035	—	—	—	—	—	—	3.238
0.0075	—	—	—	—	—	—	5.144
—	0.90	—	—	—	—	—	5.019
—	0.98	—	—	—	—	—	4.697
—	—	0.95	—	—	—	—	3.305
—	—	0.999	—	—	—	—	4.400
—	—	—	1e-10	—	—	—	3.272
—	—	—	1e-8	—	—	—	3.280
—	—	—	—	500	—	—	4.618
—	—	—	—	2000	—	—	4.010
—	—	—	—	—	0.0	—	3.309
—	—	—	—	—	2.0	—	3.302
—	—	—	—	—	—	0.00	3.275
—	—	—	—	—	—	0.20	3.275

Table 16: Hyperparameter ablation for AdamW SR on 130M (1× Chinchilla Data). Parameters in bf16 and activations in bf16.

η	β_1	β_2	ϵ	warmup	max_grad_norm	weight_decay	Final loss
0.0050	0.95	0.98	1e-9	2000	1.0	0.10	3.168
0.0035	—	—	—	—	—	—	3.194
0.0075	—	—	—	—	—	—	5.060
—	0.90	—	—	—	—	—	3.213
—	0.98	—	—	—	—	—	4.775
—	—	0.95	—	—	—	—	3.226
—	—	0.999	—	—	—	—	5.236
—	—	—	1e-10	—	—	—	3.214
—	—	—	1e-8	—	—	—	3.213
—	—	—	—	500	—	—	4.607
—	—	—	—	1000	—	—	3.213
—	—	—	—	—	0.0	—	3.220
—	—	—	—	—	2.0	—	3.227
—	—	—	—	—	—	0.00	3.213
—	—	—	—	—	—	0.20	3.213

Table 17: Hyperparameter ablation for Adam with Stochastic Rounding on 130M (1× Chinchilla Data). Parameters in bf16 and activations in bf16.

η	β_1	β_2	ϵ	warmup	max_grad_norm	weight_decay	Final loss
0.0050	0.90	0.95	1e-8	2000	1.0	0.10	3.099
0.0035	—	—	—	—	—	—	3.114
0.0075	—	—	—	—	—	—	3.148
—	0.95	—	—	—	—	—	3.120
—	0.98	—	—	—	—	—	3.182
—	—	0.98	—	—	—	—	3.120
—	—	0.999	—	—	—	—	3.143
—	—	—	1e-10	—	—	—	3.120
—	—	—	1e-9	—	—	—	3.120
—	—	—	—	500	—	—	3.177
—	—	—	—	1000	—	—	3.120
—	—	—	—	—	0.0	—	3.122
—	—	—	—	—	2.0	—	3.130
—	—	—	—	—	—	0.00	3.215
—	—	—	—	—	—	0.20	3.123

Table 18: Hyperparameter ablation for Adam on 130M (1× Chinchilla Data). Parameters in bf16 and activations in bf16.

η	β_1	β_2	ϵ	warmup	max_grad_norm	Final loss
0.0050	0.95	0.95	1e-8	1000	1.0	3.226
0.0035	—	—	—	—	—	3.238
0.0075	—	—	—	—	—	5.144
—	0.90	—	—	—	—	5.019
—	0.98	—	—	—	—	4.697
—	—	0.98	—	—	—	3.275
—	—	0.999	—	—	—	4.400
—	—	—	1e-10	—	—	3.272
—	—	—	1e-9	—	—	3.275
—	—	—	—	500	—	4.618
—	—	—	—	1000	—	4.010
—	—	—	—	—	0.0	3.305
—	—	—	—	—	2.0	3.302