HADAMIZER: SPATIO-TEMPORALLY ADAPTIVE HADAMARD ROTATION FOR LOW-PRECISION DISTRIBUTED OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Transformer models have become increasingly vital across numerous applications, yet training them is fundamentally bottlenecked on high-bandwidth memory (HBM) with 32-bit optimizer states occupying a dominant share. Quantization is a key technique to alleviate this pressure, but standard coordinate-wise quantization suffers from severe clipping errors caused by heavy-tailed outliers that widen the quantization range. While orthogonal rotations effectively disperse outliers, mixing coordinates across distributed systems makes collective communication prohibitively expensive. Here, we propose **HADAMIZER**, an optimizer-native orthogonal rotation framework designed for distributed training on accelerator meshes. Spatially, it aligns rotation blocks with local shard boundaries to avoid rotation-induced collectives. Temporally, HADAMIZER adapts the randomization of Hadamard transformations to tensor dynamics: target-adaptive keying employs static keys for persistent optimizer states and dynamic keys for stochastic gradients, preserving preconditioning with minimal quantization errors. Evaluated on TPUv5e meshes, HADAMIZER achieves near-full-precision-equivalent results on various pre-training and fine-tuning tasks, while delivering up to 9.58% higher end-to-end training throughput over the current rotation-based training frameworks.

1 INTRODUCTION

The rapid advancement of Transformer models has been driven by scaling model size, training data, and computational resources (Kaplan et al., 2020; Hoffmann et al., 2022). These gains place growing demands on the memory capacity and communication bandwidth of distributed training systems (Shoeybi et al., 2019). At the core of modern large-scale training, adaptive gradient methods, such as Adam (Kingma & Ba, 2014) and its variants (Loshchilov & Hutter, 2017; Shazeer & Stern, 2018; Vyas et al., 2024), serve as the de facto standard for both pre-training and fine-tuning. However, these optimizers maintain two 32-bit state values per parameter, the first- and second-moment estimates m_t and v_t, consuming 8 bytes per parameter. These optimizer states are the dominant contributor to a substantial persistent footprint in high-bandwidth memory (HBM) (Rajbhandari et al., 2020). Even worse, updating these states generates memory traffic, while aggregating gradients introduces communication overhead across accelerator meshes.

Model compression serves as one of the most hardware-efficient methods to mitigate the aforementioned computation and memory demands of the Transformer-based language model (Dettmers et al., 2022; Xiao et al., 2023; Lin et al., 2024). Much of the progress in quantization has focused on weights and activations (Frantar et al., 2022; Liu et al., 2026), particularly for inference. Training introduces additional opportunities through the quantization of gradients (Alistarh et al., 2017; Wen et al., 2017) or optimizer states (Dettmers et al., 2021; Li et al., 2023; Xi et al., 2025; Vyas et al., 2025), but also imposes distinct numerical requirements. As gradients change at every iteration and optimizer states accumulate information over multiple updates, errors introduced into these tensors can affect subsequent optimization steps. Moreover, heavy-tailed outliers expand the quantization range, reducing resolution for smaller values or causing clipping when the range is restricted.

Orthogonal rotations provide a promising mechanism for mitigating this sensitivity by redistributing large magnitudes across coordinates and recent methods have already demonstrated their effectiveness for low-precision language model (Chee et al., 2023; Ashkboos et al., 2024; Tseng et al., 2024; Liu et al., 2025; Tian et al., 2026; Wu et al., 2026). Yet, integrating rotation into distributed training requires addressing both execution cost and temporal consistency. From a spatial perspective,

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

transforms that span a sharded system require cross-device communication. More importantly, gradients and optimizer states have different lifetimes. Communicated gradients can be reconstructed within a single iteration, whereas moment estimates must retain a consistent interpretation across iterations. This distinction is especially important for Adam's coordinate-wise second moment, which cannot be generally transported between rotation bases using the same linear transformation as the first moment. These training-time statistics are non-stationary and drift over training, requiring continuous tracking (Yang et al., 2023; Akhondzadeh et al., 2025). Moreover, a single uniform transform cannot serve the distinct dynamics of stochastic gradients and slowly varying optimizer states. Consequently, choosing the rotation basis and its update schedule is part of the optimizer design.

Contributions. We argue that the aforementioned challenges arise from applying rotation without accounting for shard boundaries or the lifetime of the rotated representation. Our key insight is to align both the scope and persistence of coordinate mixing with the tensors' spatial and temporal structure. To this end, we introduce **HADAMIZER**, an Adam-based optimizer framework that combines this spatial locality with rotation-key schedules matched to tensor lifetimes. Our contributions are threefold:

- **Spatial design with shard-aligned block rotation.** We propose shard-aligned rotation (S-A), keeping each rotation block within a local shard to avoid rotation-induced collectives. HADAMIZER sets the block size from the hardware execution tile size and the roofline ridge point rather than from a tuned hyperparameter. As a result, S-A achieves up to 2.00× rotation speedups over global rotation.
- **Temporal design with target-specific key schedules.** As gradients and moments maintain their rotated frames for different durations, HADAMIZER employs adaptivity in the rotation randomness key for each target. HADAMIZER-GRAD draws a fresh key every step to control gradient quantization distortion and uses low-precision communication to reduce transfer volume. HADAMIZER-STATE holds one key for the whole run, which keeps moments in a fixed Hadamard basis and cuts the optimizer state footprint.
- **Comprehensive evaluation of HADAMIZER as a new optimizer.** We benchmark HADAMIZER on five open-source models with both pre-training and fine-tuning. We observe that HADAMIZER closely tracks the accuracy of full-precision variants across a wide set of tasks while also improving end-to-end training throughput 9.58% over the state-of-the-art rotation-based training framework.

2 BACKGROUND & RELATED WORK

Outlier Suppression via Hadamard Rotation. Heavy-tailed outliers severely degrade low-precision model quantization by inflating scale factors (Dettmers et al., 2022; Xiao et al., 2023). Kurtosis represents how much of a distribution's mass lies in its tails. Mathematically, it is the fourth central moment of the population divided by the fourth power of its standard deviation, which equals 3 for a Gaussian distribution. In this work, we use excess kurtosis (κ), subtracting 3 from the raw kurtosis to set the normal curve's kurtosis to 0. A large positive κ indicates frequent outliers around a sharp peak. In contrast, a small value of κ indicates light tails and a flatter profile, such as in a uniform or beta distribution.

Orthogonal rotation is an attractive approach to mitigate these outliers by dispersing extreme magnitudes across coordinates without altering the loss landscape (Ashkboos et al., 2024; Liu et al., 2025). Specifically, the randomized Hadamard transform $\tilde{\mathbf{H}}_n$ modulates a recursively constructed Walsh–Hadamard matrix of size n with a diagonal sign matrix:

$$\mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{H}_n = \mathbf{H}_2 \otimes \mathbf{H}_{n/2}, \quad \tilde{\mathbf{H}}_{n,k} = \frac{1}{\sqrt{n}} \mathbf{H}_n \mathbf{D}_k, \tag{1}$$

where $\mathbf{D}_k = \mathrm{diag}(\mathbf{s}_k)$ and $\mathbf{s}_k \in \{\pm 1\}^n$ is a Rademacher sign vector generated from key k. This recursion is guaranteed only for n a power of two, and other lengths can be zero-padded to the next such size. The randomized transform $\tilde{\mathbf{H}}_n$ is orthogonal ($\tilde{\mathbf{H}}_n \tilde{\mathbf{H}}_n^\top = \mathbf{I}_n$), preserving Euclidean norms. The randomness introduced through Rademacher sign vector $\mathbf{s}$ reduces the sensitivity to input alignment with the deterministic Hadamard basis and yields tighter bounds on the maximal outlier than an unstructured random rotation of the same size (Tseng et al., 2024). These signs are generated

from a pseudorandom number generator (PRNG) key, whose update frequency controls the persistence of the rotation basis across repeated applications. A fixed key reuses the same signs throughout the sequence, whereas refreshing the key at each iteration generates a new sign vector and changes the rotation basis. Applying $\tilde{\mathbf{H}}_n$ through dense matrix multiplication requires $\mathcal{O}(n^2)$ arithmetic operations and maps naturally to matrix units. Alternatively, the fast Walsh–Hadamard transform uses $\log_2 n$ butterfly stages, requiring element-wise $\mathcal{O}(n \log n)$ additions and subtractions (Fino & Algazi, 1976).

Low-Precision Training. Recently, incorporating low-precision representations and orthogonal matrix transformations into the training loop have been proposed to improve memory efficiency, numerical stability, and update dynamics. Quantization-based approaches compress optimizer statistics and other training tensors, using block-wise scaling or distribution transformations to control representation error (Dettmers et al., 2021; Li et al., 2023; Xi et al., 2025). Rotation-based and matrix-structured methods use Hadamard transforms (Ashkboos et al., 2026), adaptive eigenbases (Gupta et al., 2018; Vyas et al., 2025), or momentum orthogonalization (Jordan et al., 2024) to mitigate quantization outliers or reshape optimization updates. Randomized rotation has also been explored for communication-efficient distributed mean estimation, including settings with heterogeneous communication budgets (Suresh et al., 2017; Vargaftik et al., 2022). Despite extensive prior work on optimizer-state quantization and rotation-based training, applying these techniques together requires considering both the shard geometry and the lifetime of accumulated statistics (Tang et al., 2021). In particular, the rotation must fit the distributed tensor layout, and its randomness must be managed according to whether the quantized representation persists across iterations. This motivates a joint design question: *How can we structure the rotation across shards and over training steps to enable efficient and stable quantization?*

3 THE HADAMIZER FRAMEWORK

HADAMIZER addresses this question through a spatio-temporal design. Spatially, it aligns Hadamard blocks with local shards to avoid additional rotation-induced communication. Temporally, it selects the rotation-key schedule according to the quantization target, using fixed keys for persistent optimizer states and per-step keys for communicated gradients in two alternative modes. This design connects the execution geometry of rotation with the lifetime of the quantized representation. Table 1 summarizes the quantization targets and rotation strategies of representative methods to our work HADAMIZER.

Table 1: Comparison with prior work.

Property	**8-bit Optimizer** (Dettmers et al., 2021)	**4-bit Optimizer** (Li et al., 2023)	**COAT** (Xi et al., 2025)	**SOAP** (Vyas et al., 2025)	**HALO** (Ashkboos et al., 2026)	**This Work HADAMIZER**
Quantization	States	States	States, activations	None	Weights, activations, errors	States, gradients
Rotation	None	None	None	Eigenbasis	Hadamard	Hadamard
Basis / key	—	—	—	Periodically updated	Operation-specific	Fixed (State), Per-step (Grad), shard-aligned

Figure 1 illustrates HADAMIZER, an Adam-based orthogonal reparameterization of the gradient and state buffers for low-precision distributed training. Designed with a spatio-temporally efficient approach, it combines a shard-aligned block-diagonal randomized Hadamard transform (Section 3.1) with a dual-mode sign key policy matched to the temporal structure of each state buffer (Section 3.2). In the sharded pipeline (Figure 1a), HADAMIZER-GRAD rotates gradients for low-precision communication and restores them within each step, whereas HADAMIZER-STATE accumulates moments in a fixed rotated basis. These different representation lifetimes motivate their respective key schedules (Figure 1b).

3.1 SPATIAL DESIGN: SHARD-ALIGNED HADAMARD ROTATION

HADAMIZER leverages quantized Fully Sharded Data Parallel (FSDP) to reduce communication overhead. FSDP (Zhao et al., 2023) is a standard strategy for large-scale language model training where the parameters are partitioned between p devices and the layer subset is gathered whenever

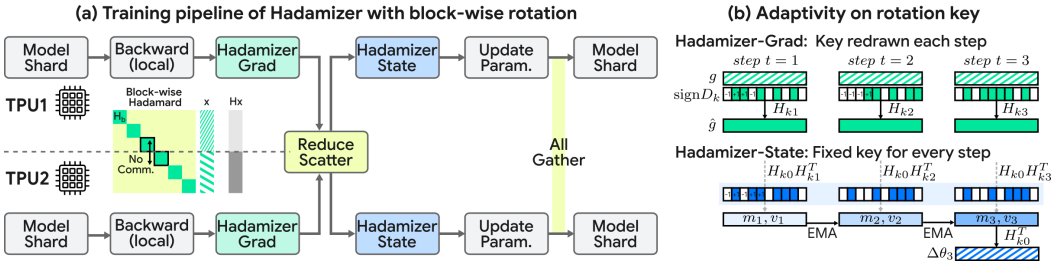


Figure 1: HADAMIZER framework. Dashed tensors are in original coordinates with high precision, and colored tensors are rotated and then quantized. Note that in (a), the block-diagonal matrix denotes the $\ell \times \ell$ transformation operator acting on the flattened local shard; all coordinates of the sharded tensor are partitioned into blocks and rotated. In (b), colored entries in the sign $\mathbf{D}_k$ denote $+1$, while white entries denote -1, and EMA refers to exponential moving averages. Gradient and state rotations differ in their key-update schedules across training steps.

necessary. The gradients are reduce-scattered after the backward pass so that every device retains the averaged gradient for its own shard. This reduce-scatter payload sets the communication volume, and the Adam moments $\mathbf{m}_t, \mathbf{v}_t$ set the optimizer memory at twice the parameter count. Both are shard-local at the moment they are written, and both carry heavy-tailed coordinate-aligned outliers that make low-bit representation lossy. HADAMIZER rotates each of them in place, which lets the reduce-scatter be issued in reduced precision and the moments be stored with minimal storage with no separate encode and decode stage.

Rotation Boundary Policy. For a weight matrix with input and output dimensions d_{in} and d_{out}, its gradient and Adam moments each contain $d_{\text{in}}d_{\text{out}}$ entries. Sharding over p devices leaves a local segment of $\ell = d_{\text{in}}d_{\text{out}}/p$ entries, which HADAMIZER partitions into $N_b = \lceil \ell/b \rceil$ contiguous blocks of size b. Stacking the blocks as rows lets us apply their shared $b \times b$ transform through one dense matrix multiplication of shape $(N_b, b) \times (b, b)$. Writing w_B for the byte width of one element, and measuring compute C for this dense product in FLOP and memory traffic V in byte,

$$C(b) = 2N_b b^2, \qquad V(b) = 2w_B N_b b, \qquad \mathrm{AI}(b) = \frac{C(b)}{V(b)} = \frac{b}{w_B} \ \text{[FLOP/byte]}, \tag{2}$$

where the arithmetic intensity $\mathrm{AI}(b)$ is independent of N_b, of p, and of the model. Shard geometry and the ridge point alone therefore fix the block size. Let π be the peak compute throughput of the accelerator in FLOP/s and β its peak memory bandwidth in byte/s. The roofline ridge point $\mathrm{AI}_{\text{ridge}} = \pi/\beta$ is the arithmetic intensity at which the compute roof and the bandwidth roof meet, so a kernel with $\mathrm{AI}(b) < \mathrm{AI}_{\text{ridge}}$ is memory bound. With M the tile size of the matrix execution unit and $B_{\max} = 2^{\lfloor \log_2 w_B \, \mathrm{AI}_{\text{ridge}} \rfloor}$ the largest power-of-two block keeping Equation 2 below the $\mathrm{AI}_{\text{ridge}}$,

$$b = \mathrm{clamp}\Big(2^{\lfloor \log_2(\ell) \rfloor},\, M,\, B_{\max}\Big). \tag{3}$$

The inner term is the largest block that fits inside a shard, which makes the rotation *shard-aligned*, so no block spans a device boundary and no collective is issued. The ceiling keeps the kernel memory-bound, where arithmetic overhead is entirely hidden by memory latency and incurs no latency penalty. HADAMIZER applies the randomized Hadamard transform of Equation 1 to each block,

$$\mathbf{y}^{(n)} = \tilde{\mathbf{H}}_{b,k}\mathbf{x}^{(n)}, \qquad \tilde{\mathbf{H}}_{b,k} = \frac{1}{\sqrt{b}}\mathbf{H}_b\mathbf{D}_k, \qquad n = 1, \ldots, N_b, \tag{4}$$

with sign key k, which is equivalent to applying the block-diagonal orthogonal matrix $\mathbf{H}_k^{\oplus N_b} = \mathrm{diag}(\mathbf{H}_k, \ldots, \mathbf{H}_k)$ to the segment. A trailing block with $\ell \bmod b \neq 0$ is completed with device-local zero-padding, which introduces neither coordinate distortion nor synthetic noise due to the isometry of $\mathbf{H}_k$, so ℓ needs not be divisible by b. In addition, the transform preserves ℓ_2 norms ($\mathbf{H}_k^\top \mathbf{H}_k = \mathbf{I}_b$) while spreading coordinate-aligned outliers across the block. Within a block, this drives the excess kurtosis into $[-2, 0]$ regardless of how heavy the input tails are, and bounds the peak-to-ℓ_2 ratio at $\mathcal{O}(\sqrt{\log b/b})$ (Appendix A.1).

Hardware-Amenable Rotation. Beyond aligning the block boundary to the system shard, the choice of execution unit matters. We issue the rotation on the hardware matrix unit rather than as an

$\mathcal{O}(b \log_2 b)$ butterfly on the vector unit, because the $b \times b$ operand maps directly onto one matrix unit tile while the vector unit is already occupied by the element-wise moment updates. Managing the randomness of the transform across a distributed mesh is the second concern, since every device must apply the identical sign vector to keep the rotation consistent across shards. HADAMIZER represents a key k as a scalar PRNG seed and regenerates $\mathbf{s}_k$ locally on each device through JAX FOLD_IN, so that the transform adds 32 bits of state per buffer without communication. Neither $\mathbf{D}_k$ nor a scale table is ever exchanged or stored. Finally, rotation is restricted to buffers of length above a threshold τ. For small state buffers like normalization scales and biases, kernel dispatch overhead outweighs the memory traffic saved by quantization, so they remain unrotated at high precision.

3.2 TEMPORAL DESIGN: DUAL-MODE KEY ADAPTIVITY

The gradient and the moments hold their rotated frame for different durations. A gradient is rotated, quantized, communicated, and returned to original coordinates within a single step, so its rotation can be resampled independently at each step. Moments, in contrast, accumulate information across the entire optimization trajectory, requiring a persistent coordinate system. HADAMIZER matches the key policy to this temporal asymmetry. The gradient path draws a fresh key every step whereas the state path holds one key for the run.

HADAMIZER-GRAD: Dynamic Keys for Gradients. Gradients $\mathbf{g}_t$ are stochastic and depend on the optimization history, including parameters updated using previous gradients. With a single static key, the resulting gradient sequence becomes statistically dependent on the fixed rotation, preventing a fresh randomized Hadamard distortion guarantee from being invoked independently at each step. HADAMIZER-GRAD instead draws a new key k_t independently of the current gradient, quantizes $\hat{\mathbf{g}}_t = \mathcal{Q}(\mathbf{H}_{k_t}\mathbf{g}_t)$ before the reduce-scatter, and restores the reduced result as $\mathbf{g}'_t = \mathbf{H}_{k_t}^\top \hat{\mathbf{g}}_t$. Conditioned on the $\mathbf{g}_t$, the fresh random transform satisfies the randomized Hadamard concentration bound at every step, controlling the peak rotated coordinate and hence the quantization distortion before communication. This argument does not rely on a bias distinction between static and dynamic keys. Under stochastic rounding, both key policies yield unbiased reconstruction error and vanishing temporal error cross-terms (Appendix A.2). The benefit of the dynamic policy is instead that the rotation randomness used to establish the per-step distortion bound remains independent of the current gradient.

HADAMIZER-STATE: Static Keys for Optimizer States. Adam accumulates history through exponential moving averages,

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\,\mathbf{g}_t, \qquad \mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\,\mathbf{g}_t^2, \tag{5}$$

which forces the optimizer to remain in a consistent basis across steps. A step-varying key requires both accumulated moments to be re-expressed in each newly sampled frame. Although the first moment can be rotated exactly by linearity, this is not feasible for the second moment. Adam retains only the diagonal of the second-order statistic, whereas changing basis requires the off-diagonal terms of the corresponding covariance matrix. As a result, a dynamically rekeyed diagonal second moment cannot in general be reconstructed from the stored state alone.

To this end, HADAMIZER-STATE fixes a single key k_0 for the entire run and rebases each incoming gradient as $\mathbf{r}_t = \mathbf{H}_{k_0}\mathbf{g}'_t$, shown as the composition $\mathbf{H}_{k_0}\mathbf{H}_{k_t}^\top$ in Figure 1b. The moment updates run entirely on $\mathbf{r}_t$,

$$\tilde{\mathbf{m}}_t = \beta_1 \tilde{\mathbf{m}}_{t-1} + (1-\beta_1)\,\mathbf{r}_t, \qquad \tilde{\mathbf{v}}_t = \beta_2 \tilde{\mathbf{v}}_{t-1} + (1-\beta_2)\,\mathbf{r}_t \odot \mathbf{r}_t, \tag{6}$$

and only the final update $\Delta\boldsymbol{\theta}_t$ maps back to original coordinates. The first moment commutes with the rotation, $\tilde{\mathbf{m}}_t = \mathbf{H}_{k_0}\mathbf{m}_t$ for the moment accumulated from the same sequence $\mathbf{g}'_t$, by linearity in $\mathbf{r}_t$. The second moment admits no such relation ($(\mathbf{Hg})^{\odot 2} \neq \mathbf{Hg}^{\odot 2}$), which makes the resulting update as a block-diagonal preconditioner $\mathbf{H}_{k_0}^\top \operatorname{diag}(\sqrt{\tilde{\mathbf{v}}_t} + \epsilon)^{-1}\mathbf{H}_{k_0}$ rather than an exact coordinate transformation of Adam. The fixed frame preserves the accumulated optimizer state while retaining the reduced dynamic range provided by the Hadamard basis.

Justification and Resource Targets of Dual-Mode Key Adaptivity. Table 2 measures the trajectory fidelity under static and dynamic keys of each path with respect to an unrotated FP32 reference. The state path confirms the basis-consistency requirement. Under a static key, the first moment matches the reference to measurement precision at every step. Under a dynamic key, the discrepancy appears immediately because the accumulated state is no longer represented in a persistent basis.

For the gradient path, the dynamic-key policy yields substantially smaller cumulative trajectory distance. The distance remains below 2×10^{-4} and decreases with step count, whereas the static-key policy grows from 1.799×10^{-3} to 15.868×10^{-3} over the first 100 steps. We treat this as an empirical consequence of refreshing the rotation seen by each gradient rather than as evidence that stochastic-rounding errors under a static key are biased or temporally correlated. Appendix A.2 establishes zero-mean reconstruction and vanishing temporal cross-terms for both policies. Taken together, the analysis and measurements show that no single key policy is uniformly preferable. Transient gradients benefit from fresh rotation randomness, whereas persistent optimizer states require a fixed basis for accumulation.

	State Dist. to FP32 ($\mathbf{m}_t$)		Grad Cumul. Dist. to FP32 ($\mathbf{g}_t$)	
Step	**Static**	**Dynamic**	**Static**	**Dynamic**
1	$< 10^{-6}$	0.3797	1.799×10^{-3}	**0.158** $\times10^{-3}$
5	$< 10^{-6}$	0.7680	3.886×10^{-3}	**0.085** $\times10^{-3}$
10	$< 10^{-6}$	0.9428	5.972×10^{-3}	**0.042** $\times10^{-3}$
25	$< 10^{-6}$	0.7358	8.795×10^{-3}	**0.018** $\times10^{-3}$
50	$< 10^{-6}$	0.6502	11.530×10^{-3}	**0.009** $\times10^{-3}$
100	$< 10^{-6}$	0.7584	15.868×10^{-3}	**0.004** $\times10^{-3}$

Table 2: Trajectory cosine distance ($1 - \cos$) across training steps after warm-start on GEMMA3-270M. Lower is better.

Table 3: Theoretical comparison of HADAMIZER quantized into INT8 against baseline FP32 Adam with 8,192 tokens. We set $\tau = 1$M for GEMMA3-270M and $\tau = 2$M for GEMMA3-1B to rotate the same fraction (42.9%) of tensors in both models. Parentheses indicate savings relative to the baseline.

	GEMMA3-270M $\tau = 1M$ ($N_{\text{rot}} = 70.8$M)			**GEMMA3-1B** $\tau = 2M$ ($N_{\text{rot}} = 621.1$M)		
Method	Target State HBM Footprint (GB)	Target Gradient Transfer (MB)	FLOPs overhead	Target State HBM Footprint (GB)	Target Gradient Transfer (GB)	FLOPs overhead
Adam (FP32)	0.57	142	—	4.97	1.24	—
HADAMIZER-STATE	**0.21 (62.5%)**	142	$< 0.15\%$	**1.86 (62.5%)**	1.24	$< 0.35\%$
HADAMIZER-GRAD	0.57	**71 (50.0%)**	$< 0.15\%$	4.97	**0.62 (50.0%)**	$< 0.35\%$

This separation is essential as two HADAMIZER variants act on different resources. Table 3 reports the theoretical cost of each mode at INT8 against FP32 Adam. HADAMIZER-STATE reduces the optimizer state footprint by 62.5% and leaves gradient traffic unchanged, as the state never leaves the rotated basis and no inverse rotation enters the communication path. HADAMIZER-GRAD halves the gradient transfer volume and leaves the state footprint unchanged, as the rotation is undone immediately after the collective. The arithmetic overhead of the block-diagonal rotation stays below 0.15% at GEMMA3-270M and below 0.35% at GEMMA3-1B for N_{rot} rotated parameters. Memory and bandwidth are therefore addressed by separate mechanisms with independent keys and block sizes. The two modes compose without interference because they meet only in original coordinates.

4 EVALUATION

4.1 EXPERIMENTAL SETUP

Models and Tasks. We evaluate HADAMIZER on five models from two families to establish cross-architecture generality: GEMMA3-270M and GEMMA3-1B (Team et al., 2025), and QWEN2.5-0.5B, QWEN2.5-1.5B, and QWEN2.5-3B (Hui et al., 2025). Our evaluation covers pre-training on the Colossal Clean Crawled Corpus (C4/EN, Raffel et al., 2020) and fine-tuning on machine translation (MTNT/EN-FR, Michel & Neubig, 2018). Evaluating both pre-training and fine-tuning is deliberate: while pre-training tests outlier suppression and dynamic range control under large, heavy-tailed gradient spikes, fine-tuning evaluates whether low-precision representations introduce excessive quantization noise in the regime of small gradient magnitudes where subtle updates could otherwise be obscured.

All runs are deployed on an 8×8 Google TPU v5e mesh in Fully Sharded Data Parallelism manner for 10k steps. While we evaluate rotation only on TPU, our shard alignment and target-specific key schedules do not require TPU-specific features. The block size and rotation kernel can be adapted to the execution units and memory bandwidth of the target accelerator.

Baselines. Full-precision ADAM keeps both moments and the gradient path in 32-bit floating point and serves as the quality anchor. HALO (Ashkboos et al., 2026) is a state-of-the-art low-precision training framework that applies Hadamard rotation across the full dimension of the selected matrix axis and uses per-tensor quantization. MUON (Jordan et al., 2024) orthogonalizes the momentum

update using five Newton–Schulz iterations without quantization or Hadamard rotation to assess HADAMIZER against an alternative optimizer. HADAMIZER selects its rotation block size according to the sharding layout, then applies symmetric INT8 quantization using stochastic rounding with a separate scale for each block within a shard. HADAMIZER's rotation thresholds are set to rotate 42.9% of parameters of each model (e.g., 1M for GEMMA3-270M). To ensure training quality and isolate the efficiency gains of HADAMIZER for each quantization target, we evaluate HADAMIZER-GRAD and HADAMIZER-STATE separately without a combined gradient-and-state implementation.

4.2 MAIN RESULTS

Figure 2 illustrates the quality–latency trade-off on GEMMA3-270M as a representative example. We measure step time as the average wall-clock time per training step with a batch size of 512, a sequence length of 1024, and a learning rate of 4×10^{-4}. The two HADAMIZER variants occupy complementary points on the empirical Pareto frontier: HADAMIZER-STATE achieves the lowest loss, while HADAMIZER-GRAD achieves the shortest step time. Specifically, HADAMIZER-STATE reduces loss by 4.15% relative to full-precision ADAM while maintaining a comparable step time. As analyzed in Appendix A.3, accumulating second moments in a fixed Hadamard basis acts as a dense block-diagonal preconditioner, which prevents outliers in ADAM from over-dampening update steps along critical directions. Compared to HALO, it achieves a slightly lower loss with a 6.47% shorter step time, consistent with the lower rotation overhead observed in Section 4.3. These results demonstrate that maintaining optimizer states in a fixed rotated INT8 representation can preserve training quality while remaining competitive in end-to-end performance. HADAMIZER-GRAD trades higher loss for a shorter step time of 150.92 ms, increasing throughput by 2.78% over ADAM and 9.58% over HALO. On top of the shard-aligned rotation that eliminates additional cross-device collectives for the transform, INT8 quantization shrinks the gradient payload and reduces the communication costs.

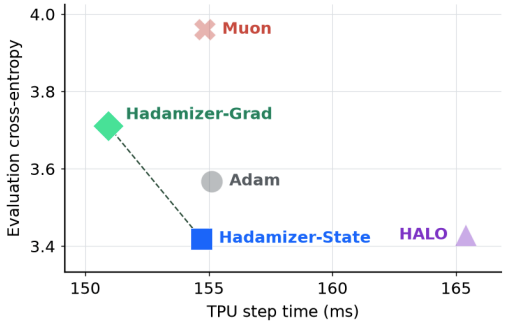


Figure 2: Evaluation cross-entropy and TPU step time for GEMMA3-270M on C4/EN. Lower is better for both axes and the dashed line connects the Pareto frontier.

Beyond low-precision baselines, we compare HADAMIZER with MUON as an alternative optimizer. HADAMIZER-STATE achieves substantially lower loss (3.4197 versus 3.9602) at a comparable step time (154.69 versus 154.82 ms), while HADAMIZER-GRAD improves both loss and latency, reducing them by 6.27% and 2.52%, respectively. This comparison reflects differences in both communication and optimizer computation. HADAMIZER-GRAD reduces gradient communication through INT8 quantization, whereas MUON performs momentum orthogonalization using five Newton–Schulz iterations per step, each involving multiple matrix multiplications.

Together, these results demonstrate that both HADAMIZER variants are non-dominated among the evaluated configurations, supporting HADAMIZER as a flexible framework for low-precision distributed training. Detailed numbers of pre-training and fine-tuning results are summarized in Appendix B.1. Notably, while HADAMIZER-STATE maintains superior quality in both regimes, HADAMIZER-GRAD incurs a modest quality penalty on fine-tuning (Table 9). This highlights that fine-tuning is far more forgiving of quantized states than quantized gradients: ADAM's moving averages naturally smooth out state quantization noise in a fixed Hadamizer basis, whereas per-step gradient quantization directly perturbs the small updates needed to adapt pre-trained weights.

4.3 ABLATION STUDY

Rotation Overhead. To assess the end-to-end performance of the rotation implementations without quantization savings, we compare the rotation-only version of HADAMIZER against HALO. HADAMIZER reduces step time by up to 19.31% on GEMMA3-270M, with additional reductions of 6.60% on GEMMA3-1B and 9.64% on QWEN2.5-3B. These results motivate considering model dimensions and shard geometry when selecting rotation blocks.

Table 4 shows the rotation kernel latency of each model's feed-forward-network when increasing the rotation block size at a fixed 4×4 TPU v5e mesh. The HADAMIZER's S-A choice is faster than global rotation on every model, up to 2.00× at 1.5B. This is because global rotation pads the hidden dimension to the next power of two, which inflates the working set by 68.42% at $d = 4864$ and by 82.86% at $d = 8960$, while block rotation requires only $b \leq \ell$ and pads none of these dimensions. Global rotation also spans the full sharded axis and spends 10.18 to 17.39 microseconds in all-to-all collectives, against none for the S-A choice, since a block contained in a shard is completed with device-local padding and issues no collective. Blocks larger than $B_{\max}$ move beyond the memory-bound regime and become increasingly compute-bound. At $b = 512$, the arithmetic intensity reaches 256 and exceeds the roofline ridge point, leading to higher latency. Compared with the S-A choice, latency increases by 35.63% for the 1B model and 73.06% for the 3B model. Additional analysis of how rotation overhead varies with system topology is provided in Appendix B.2.

Table 4: Rotation kernel latency and T_{A2A} in parentheses, in microseconds, on feed-forward-network. $\ell = d/P$. Highlighted cells are the S-A choice.

		4×4			
Model	ℓ	Global	b=128	b=256	b=512
GEMMA3-270M	128	77.10 (11.08)	71.30 (0.00)	74.65 (2.43)	98.67 (5.82)
GEMMA3-1B	432	183.19 (17.39)	107.26 (0.00)	118.43 (0.00)	145.48 (11.48)
QWEN2.5-0.5B	304	155.51 (15.98)	89.20 (0.00)	95.59 (0.00)	111.12 (9.76)
QWEN2.5-1.5B	560	277.80 (16.44)	130.67 (0.00)	138.72 (0.00)	213.27 (0.00)
QWEN2.5-3B	688	357.23 (10.18)	211.08 (0.00)	198.75 (0.00)	343.95 (0.00)

Rotation Threshold. We further examine how the rotation threshold τ affects the quality–latency trade-off of HADAMIZER variants in Table 5. We use model-specific batch sizes to accommodate the larger memory footprint of a larger model, with a 512 batch size for GEMMA3-270M and 256 for GEMMA3-1B. The learning rate is scaled proportionally with batch size, yielding 4×10^{-4} and 2×10^{-4}, respectively, while the sequence length is fixed at 1024.

On GEMMA3-270M, increasing τ from 1M to 2M reduces step time by 8.35% for HADAMIZER-STATE and 5.66% for HADAMIZER-GRAD, while lowering cross-entropy for both. On GEMMA3-1B, increasing τ from 4M to 8M also lowers cross-entropy for both variants, although step time increases slightly. Across both models, HADAMIZER-GRAD exhibits a larger cross-entropy change than HADAMIZER-STATE. These variants differ in how rotation enters the optimization process. Since the gradient path carries the round trip of rotation and inverse rotation at every step under the dynamic key, a larger τ exempts smaller tensors from this round trip and reduces both rotation overhead and the accumulated rounding. For HADAMIZER-STATE, the static key retains persistent moments in a fixed basis. Avoiding per-step changes to this representation preserves evaluation loss across the same sweep, contrasting with the behavior of HADAMIZER-GRAD.

Table 5: Rotation threshold (τ) sensitivity on evaluation cross-entropy (CE) and training step time (ms) of HADAMIZER variants.

	GEMMA3-270M			GEMMA3-1B		
Method	τ	**CE** ↓	**Time** ↓	τ	**CE** ↓	**Time** ↓
HADAMIZER-STATE	1M	3.4262	168.79	4M	3.4008	280.09
	2M	3.4197	154.69	8M	3.3855	283.85
HADAMIZER-GRAD	1M	3.8120	159.97	4M	3.9105	274.98
	2M	3.7118	150.92	8M	3.7304	275.27

In contrast to the prior approaches (Ashkboos et al., 2024; Liu et al., 2025; Ashkboos et al., 2026) that employ rotation schemes tailored only to their target computations and quantized tensors, HADAMIZER explicitly sets the rotation threshold τ to provide finer control over rotation coverage by determining which layer tensors are rotated according to their size. We underscore that rotation coverage should be selected jointly with the optimizer mode and workload to balance computational cost and training quality.

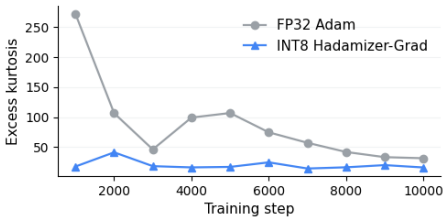
Training Stability. We assess the training stability of HADAMIZER, especially focusing on the effect of our dual-mode key adaptivity on training quality and the effect of rotation on outlier suppression. Table 6 shows the relative cross-entropy change $\Delta\%$ on fine-tuning MTNT/EN-FR, where HADAMIZER-STATE uses a static key and is compared against per-step dynamic keys, whereas HADAMIZER-GRAD uses per-step dynamic keys and is compared against a static key.

For state quantization, HADAMIZER-STATE achieves lower loss across all five models, with reductions of 4.74–41.33% relative to dynamic key for state. This supports our main novelty that preserves a consistent basis for accumulated optimizer statistics. Although the first moment can be transported linearly between bases, the coordinate-wise second moment cannot generally be reconstructed in a new basis from its diagonal statistics alone. Naive rekeying, therefore, disrupts the interpretation of

Table 6: Relative cross-entropy change of HADAMIZER and faulty mapping on key for each quantization target.

Model	HADAMIZER-STATE	HADAMIZER-GRAD
GEMMA3-270M	−7.47 %	−17.18 %
GEMMA3-1B	−41.33 %	−23.34 %
QWEN2.5-0.5B	−4.74 %	−20.64 %
QWEN2.5-1.5B	−17.79 %	−4.64 %
QWEN2.5-3B	−32.00 %	+1.00 %

Figure 3: Excess kurtosis of the GEMMA3-270M feed-forward-network gradient.



the accumulated state as discussed in Section 3.2. For gradient quantization, HADAMIZER-GRAD overall achieves lower loss with reductions of 4.64–23.34% relative to static gradient key except for QWEN2.5-3B. Since gradients are restored to original coordinates within each step, their rotation keys can change without reinterpreting persistent moments. Together, these results support target-specific key scheduling: a fixed basis preserves optimizer-state consistency, while per-step key refresh improves gradient-quantized training quality in most evaluated models.

Figure 3 evaluates how effectively each method suppresses gradient outliers during C4/EN pre-training, measured by the tensor-level excess kurtosis κ^{tot} of the first feed-forward-network layer in GEMMA3-270M. In the original basis, it peaks at 272.40 early in training and remains above 30.00 throughout the training interval. With HADAMIZER-GRAD, it lies between 14.10 and 41.10, yielding a median per-step reduction of 2.83×. The maximum-to-minimum ratio also decreases from 8.70× to 2.90×, indicating less variation in gradient tail behavior over training. The residual values stay positive because a block-diagonal rotation preserves each block's energy and therefore cannot reduce the dispersion of energy across blocks, which contributes an irreducible term to κ^{tot} (Appendix A.1). This residual variation does not directly affect quantization error, as HADAMIZER applies an independent absmax scale to each block, and the tensor-level curve provides a conservative view of the outlier suppression relevant to block quantization. Together, these results show that HADAMIZER's gradient rotation is effective at suppressing outliers, particularly during the early training phase when the unrotated gradients exhibit their highest kurtosis. Appendix B.3 further explores the potential of HADAMIZER for aggressive low-bit quantization with reconstruction SQNR.

5 CONCLUSION

Recently, quantization and outlier suppression through rotation for large Transformer models have been extensively explored, yet their rotation schemes overlook both the deployment cost on a distributed system and the temporal consistency of persistent optimizer states. To this end, we propose HADAMIZER, an Adam-based optimizer framework that jointly adapts the spatial scope and temporal consistency of Hadamard rotation for low-precision distributed training. Our shard-aligned transform eliminates rotation-induced cross-device communication, achieving up to 2.00× speedups over global rotation. Temporally, HADAMIZER incorporates adaptivity on randomness of Hadamard by adjusting key update frequency for gradient and momentum states. Our temporal adaptivity stabilizes training by reducing the final loss by up to 41.33% compared to a faulty-mapped key, highlighting that the representation of optimizer statistics is an explicit algorithmic design choice. Extensive experiments on open-source models with various training tasks demonstrate that HADAMIZER delivers near-full-precision training quality. Alongside the training accuracy, HADAMIZER also improves training throughput, outperforming the state-of-the-art rotation-based framework by up to 9.58% in end-to-end training.

While HADAMIZER establishes the effectiveness of adapting the spatial scope and temporal consistency of rotation, both axes leave open questions. Spatially, tying the rotation scope to the sharding layout constrains outlier suppression to the per-device shard width, which can be substantially smaller than the dimension available to a global rotation. Recovering this loss without additional cross-device communication is left to future work. Temporally, the key update frequency remains empirically tuned, motivating a principled criterion that refreshes the Hadamard key from observed drift in optimizer statistics to make the schedule transferable and self-regulating. Ultimately, HADAMIZER points toward training systems that jointly adapt numerical representation, communication, and update schedule for efficient large-scale foundation model training.

REFERENCES

Mohammad Sadegh Akhondzadeh, Aleksandar Bojchevski, Evangelos Eleftheriou, and Martino Dazzi. Kurtail: Kurtosis-based llm quantization. In *EMNLP (Findings)*, pp. 17404–17419, 2025.

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.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

Saleh Ashkboos, Mahdi Nikdan, Rush Tabesh, Roberto Castro, Torsten Hoefler, and Dan Alistarh. Halo: Hadamard-assisted lower-precision optimization for llms. *Advances in Neural Information Processing Systems*, 38:118871–118896, 2026.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in neural information processing systems*, 36: 4396–4429, 2023.

Matteo Croci, Massimiliano Fasi, Nicholas J Higham, Theo Mary, and Mantas Mikaitis. Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9(3): 211631, 2022.

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in neural information processing systems*, 35: 30318–30332, 2022.

Fino and Algazi. Unified matrix treatment of the fast walsh-hadamard transform. *IEEE Transactions on Computers*, 100(11):1142–1146, 1976.

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

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

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

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

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/`.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

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

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

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

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In *International Conference on Learning Representations*, volume 2025, pp. 92009–92032, 2025.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, et al. Paretoq: Improving scaling laws in extremely low-bit llm quantization. *Advances in Neural Information Processing Systems*, 38:91311–91336, 2026.

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

Paul Michel and Graham Neubig. Mtnt: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 543–553, 2018.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pp. 1–16. IEEE, 2020.

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

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

Ananda Theertha Suresh, X Yu Felix, Sanjiv Kumar, and H Brendan McMahan. Distributed mean estimation with limited communication. In *International conference on machine learning*, pp. 3329–3337. PMLR, 2017.

Hanlin Tang, Shaoduo Gan, Ammar Ahmad Awan, Samyam Rajbhandari, Conglong Li, Xiangru Lian, Ji Liu, Ce Zhang, and Yuxiong He. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *International Conference on Machine Learning*, pp. 10118–10129. PMLR, 2021.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Zhen Tian, Xin Zhao, and Ji-Rong Wen. Irrational complex rotations empower low-bit optimizers. *Advances in Neural Information Processing Systems*, 38:170660–170682, 2026.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *Proceedings of machine learning research*, 235:48630, 2024.

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

Nikhil Vyas, Depen Morwani, and Sham M Kakade. Adamem: Memory efficient momentum for adafactor. In *2nd Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ ICML 2024)*, 2024.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam for language modeling. In *International Conference on Learning Representations*, volume 2025, pp. 93423–93444, 2025.

Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in neural information processing systems*, 30, 2017.

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*, 2026.

Haocheng Xi, Han Cai, Ligeng Zhu, Yao Lu, Kurt Keutzer, Jianfei Chen, and Song Han. Coat: Compressing optimizer states and activations for memory-efficient fp8 training. In *International Conference on Learning Representations*, volume 2025, pp. 42989–43011, 2025.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Je Yang, Sukbin Lim, Sukjin Lee, Jae-Young Kim, and Joo-Young Kim. Jnpu: A 1.04 tflops joint-dnn training processor with speculative cyclic quantization and triple heterogeneity on microarchitecture/precision/dataflow. In *ESSCIRC 2023-IEEE 49th European Solid State Circuits Conference (ESSCIRC)*, pp. 349–352. IEEE, 2023.

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

A Technical Appendices and Proof

Throughout the appendix, we use the notation in Table 7.

Table 7: Notation used throughout the appendix.

Symbol	Meaning
b, N_b	Block size, the dimension one rotation acts on, and the total number of blocks
p, P, t, T	Device index and count, step index and total number of training steps
$\mathbf{H}_b$	Unnormalized $b \times b$ Walsh–Hadamard matrix
$\mathbf{D}_k = \mathrm{diag}(\mathbf{s}_k)$	Rademacher sign flip whose entries $s_{k,i} \in \{-1, +1\}$ generated from key k
$\tilde{\mathbf{H}}_{b,k} = \frac{1}{\sqrt{b}}\mathbf{H}_b\mathbf{D}_k$	Block transform, orthogonal
$\mathbf{R}_t = \tilde{\mathbf{H}}_{b,k_t}$	Transform applied at step t
$\mathbf{g}_t^{(p)}$, $\mathbf{g}_t$	Local gradient block on device p, and its aggregate
$\mathcal{Q}$, w, $q_w = 2^{w-1} - 1$	Absmax quantizer, bit width, code range
$\mathbf{g}'_t$, $\mathbf{e}_t = \mathbf{g}'_t - \mathbf{g}_t$	Restored gradient and its error
$\mathbf{E}_T = \sum_{t \le T} \mathbf{e}_t$	Accumulated error
$\mathcal{G}_t$, $\mathcal{G}_t^-$	Conditioning at step t with and without the key k_t
$\mathbf{m}_t$, $\mathbf{v}_t$	Adam moments in the original coordinates
$\tilde{\mathbf{m}}_t$, $\tilde{\mathbf{v}}_t$	Moments maintained in the rotated basis
$\beta_1, \beta_2, \eta, \epsilon$	Adam hyperparameters

A.1 Kurtosis Reduction Analysis

We distinguish three excess-kurtosis quantities that are easily conflated. For a fixed nonzero block $\mathbf{x} \in \mathbb{R}^b$, define the raw coordinate moments

$$\mu_2(\mathbf{x}) = \frac{1}{b}\|\mathbf{x}\|_2^2, \qquad \mu_4(\mathbf{x}) = \frac{1}{b}\sum_i x_i^4, \qquad \kappa(\mathbf{x}) = \frac{\mu_4(\mathbf{x})}{\mu_2(\mathbf{x})^2} - 3. \tag{7}$$

First, $\kappa(\mathbf{x})$ is the *within-block* raw-moment excess kurtosis, which coincides with the conventional empirical excess kurtosis when the coordinate mean is zero. Second, for a transformed coordinate y_j, we have $\kappa_k(y_j) = \mathbb{E}_k[y_j^4]/\mathbb{E}_k[y_j^2]^2 - 3$ for the *rotated* excess kurtosis taken over key randomness with $\mathbf{x}$ held fixed. Third, for a full parameter tensor partitioned into N_b blocks of size b, we have κ^{tot} for the excess kurtosis computed over all $N_b b$ coordinates jointly. Theorem 1 characterizes $\kappa_k(y_j)$, and Proposition 1 connects the two.

Theorem 1 (Kurtosis Suppression). *Let $\mathbf{y} = \tilde{\mathbf{H}}_{b,k}\mathbf{x}$ with $\mathbf{x} \in \mathbb{R}^b$ fixed. For every coordinate j, $\mathbb{E}_k[y_j] = 0$, $\mathbb{E}_k[y_j^2] = \mu_2(\mathbf{x})$, and*

$$\kappa_k(y_j) = -\frac{2}{b}\big(\kappa(\mathbf{x}) + 3\big) \in [-2, 0]. \tag{8}$$

Moreover, for any $\delta \in (0,1)$, with probability at least $1-\delta$ over k,

$$\|\mathbf{y}\|_\infty \le \sqrt{\frac{2\log(2b/\delta)}{b}}\,\|\mathbf{x}\|_2. \tag{9}$$

Proof. Every entry of $\mathbf{H}_b$ is ± 1, so $y_j = \frac{1}{\sqrt{b}}\sum_{i=1}^{b} h_{ji}s_{k,i}x_i = \frac{1}{\sqrt{b}}\sum_i s'_{k,i}x_i$ where $s'_{k,i} = h_{ji}s_{k,i}$ are again i.i.d. Rademacher. Odd moments vanish by symmetry and $\mathbb{E}_k[y_j^2] = \frac{1}{b}\sum_i x_i^2 = \mu_2(\mathbf{x})$. For the fourth moment, only terms with even multiplicity survive,

$$\mathbb{E}_k\Big[\big(\textstyle\sum_i s'_{k,i}x_i\big)^4\Big] = \sum_i x_i^4 + 3\sum_{i\neq l} x_i^2 x_l^2 = 3\Big(\sum_i x_i^2\Big)^2 - 2\sum_i x_i^4, \tag{10}$$

hence $\mathbb{E}_k[y_j^4] = 3\mu_2^2 - \frac{2}{b}\mu_4$ and $\kappa_k(y_j) = -\frac{2}{b}\,\mu_4/\mu_2^2 = -\frac{2}{b}(\kappa(\mathbf{x}) + 3)$. The range follows from $\mu_2^2 \le \mu_4 \le b\,\mu_2^2$. For Equation 9, Hoeffding's inequality (Hoeffding, 1963) applied to $\sum_i s'_{k,i}x_i$ with ranges $[-|x_i|, |x_i|]$ gives $\Pr(|y_j| \ge t) \le 2\exp(-bt^2/2\|\mathbf{x}\|_2^2)$, and a union bound over the b coordinates with the right-hand side set to δ yields the claim. □

The bound is exact rather than asymptotic. Heavier tails in $\mathbf{x}$ produce a more negative rotated kurtosis, i.e., a more sub-Gaussian coordinate. In the extreme case of a single spike, $\kappa(\mathbf{x}) = b - 3$ and $\kappa_k(y_j) = -2$, the minimum attainable by any distribution, corresponding to the spike energy being spread as $\pm|x|/\sqrt{b}$ across all b coordinates. The next result lifts this blockwise statement to the tensor level reported in Figure 3.

Proposition 1 (Tensor-level kurtosis under block rotation). *Partition a tensor into N_b blocks $\mathbf{x}^{(n)} \in \mathbb{R}^b$ for $n = 1, \ldots, N_b$, and let $\mathbf{y}^{(n)} = \tilde{\mathbf{H}}_{b,k}\mathbf{x}^{(n)}$. Write*

$$\bar{\mu}_2 = \frac{1}{N_b}\sum_{n=1}^{N_b}\mu_2(\mathbf{x}^{(n)}), \qquad \rho = \frac{1}{\bar{\mu}_2^2}\cdot\frac{1}{N_b}\sum_{n=1}^{N_b}\mu_2(\mathbf{x}^{(n)})^2 \;=\; 1 + \frac{\mathrm{Var}_n\big[\mu_2(\mathbf{x}^{(n)})\big]}{\bar{\mu}_2^2} \;\ge\; 1. \tag{11}$$

Then $\mu_2^{\text{tot}}(\mathbf{y}) = \bar{\mu}_2$ and

$$\mathbb{E}_k\big[\kappa^{\text{tot}}(\mathbf{y})\big] = \underbrace{3(\rho-1)}_{\textit{across-block dispersion}} - \underbrace{\frac{2}{b}\big(\kappa^{\text{tot}}(\mathbf{x}) + 3\big)}_{\textit{within-block suppression}}. \tag{12}$$

Proof. Since each $\tilde{\mathbf{H}}_{b,k}$ is orthogonal, $\|\mathbf{y}^{(n)}\|_2 = \|\mathbf{x}^{(n)}\|_2$ and the second raw moment is preserved blockwise. Deterministically, $\mu_2^{\text{tot}}(\mathbf{y}) = C$. By the fourth-moment computation in Theorem 1, $\mathbb{E}_k[\mu_4(\mathbf{y}^{(n)})] = 3\mu_2(\mathbf{x}^{(n)})^2 - \frac{2}{b}\mu_4(\mathbf{x}^{(n)})$. Averaging over n gives $\mathbb{E}_k[\mu_4^{\text{tot}}(\mathbf{y})] = 3B - \frac{2}{b}\mu_4^{\text{tot}}(\mathbf{x})$. Dividing by C^2 and subtracting 3 yields Equation 12, using $\mu_4^{\text{tot}}(\mathbf{x})/C^2 = \kappa^{\text{tot}}(\mathbf{x}) + 3$. The bound $\rho \ge 1$ is Jensen's inequality. □

Setting $N_b = 1$ gives $\rho = 1$ and recovers Equation 8. For $N_b > 1$, the quantity ρ depends only on the block energy profile, which any block-diagonal orthogonal map leaves invariant. The term $3(\rho - 1)$ is therefore a floor that rotation cannot remove, and it accounts for the positive tensor-level values in Figure 3. What rotation removes is the within-block contribution. The following theorem bounds the largest transformed coordinate relative to the block's ℓ_2 norm.

Theorem 2 (Peak Suppression). *Under the assumptions of Theorem 1, for any $\delta \in (0,1)$, Equation 9 bounds the largest transformed coordinate relative to the block's ℓ_2 norm with probability at least $1-\delta$ over k. For a nonzero block and fixed δ, the resulting peak-to-ℓ_2 ratio is $\mathcal{O}(\sqrt{\log b/b})$.*

A.2 HADAMIZER-GRAD: GRADIENT INVARIANCE AND QUANTIZATION BOUNDS

At step t, all devices apply the same orthogonal transform $\mathbf{R}_t = \tilde{\mathbf{H}}_{b,k_t}$ and quantize their local rotated gradients as $\hat{\mathbf{y}}_t^{(p)} = \mathcal{Q}(\mathbf{R}_t \mathbf{g}_t^{(p)})$, where $\mathbf{g}_t^{(p)} \in \mathbb{R}^b$ is the block of the local gradient on device p. Let $\mathbf{g}_t = \sum_p \mathbf{g}_t^{(p)}$, $\mathbf{y}_t = \mathbf{R}_t \mathbf{g}_t$, and $\hat{\mathbf{y}}_t = \sum_p \hat{\mathbf{y}}_t^{(p)}$. Since the transform is shared and linear,

$$\sum_p \mathbf{R}_t \mathbf{g}_t^{(p)} = \mathbf{R}_t \mathbf{g}_t.$$

Under the shard-aligned layout, this identity also applies to the corresponding output shards of reduce-scatter. Thus, rotation and inverse rotation preserve the aggregated gradient in the absence of quantization. With quantization, the restored gradient $\mathbf{g}'_t = \mathbf{R}_t^\top \hat{\mathbf{y}}_t$ has error

$$\mathbf{e}_t = \mathbf{g}'_t - \mathbf{g}_t = \mathbf{R}_t^\top (\hat{\mathbf{y}}_t - \mathbf{y}_t) = \sum_{p=1}^{P} \mathbf{e}_t^{(p)}, \qquad \mathbf{e}_t^{(p)} = \mathbf{R}_t^\top \left(\hat{\mathbf{y}}_t^{(p)} - \mathbf{R}_t \mathbf{g}_t^{(p)} \right). \tag{13}$$

Orthogonality gives $\|\mathbf{e}_t\|_2 = \|\hat{\mathbf{y}}_t - \mathbf{y}_t\|_2$. These identities hold for both static and dynamic keys. Throughout, $\mathcal{G}_t$ denotes the history through step $t-1$ including the stochastic rounding randomness at those steps, together with all local gradients at step t and the key k_t, before rounding at step t.

Quantization Distortion per Step. Let $\mathbf{y}_t^{(p)} = \mathbf{R}_t \mathbf{g}_t^{(p)}$ and $q_w = 2^{w-1} - 1$. For a symmetric per-block absmax quantizer with stochastic rounding, the quantization step is $\Delta_t^{(p)} = \|\mathbf{y}_t^{(p)}\|_\infty / q_w$. Conditioned on $\mathbf{y}_t^{(p)}$, stochastic rounding is unbiased and its coordinate-wise error variance is at most $(\Delta_t^{(p)})^2/4$ (Croci et al., 2022), so

$$\mathbb{E}_{\mathrm{SR}} \left[\|\mathbf{e}_t^{(p)}\|_2^2 \mid \mathcal{G}_t \right] \le \frac{b}{4q_w^2} \|\mathbf{y}_t^{(p)}\|_\infty^2. \tag{14}$$

where $\mathcal{G}_t^-$ for the history through step $t-1$ together with all local gradients. For a fixed $\mathbf{g}_t^{(p)}$ and an independently sampled key, Equation 9 implies that, with probability at least $1-\delta'$ over the key,

$$\mathbb{E}_{\mathrm{SR}} \left[\|\mathbf{e}_t^{(p)}\|_2^2 \mid \mathcal{G}_t \right] \le \frac{\log(2N_b/\delta')}{2q_w^2} \|\mathbf{g}_t^{(p)}\|_2^2. \tag{15}$$

Without rotation, the corresponding generic upper bound is $b\|\mathbf{g}_t^{(p)}\|_2^2/(4q_w^2)$, which follows from $\|\mathbf{g}_t^{(p)}\|_\infty \le \|\mathbf{g}_t^{(p)}\|_2$ and is attained only for a one-sparse block. For fixed δ', rotation therefore replaces the linear dependence on b in this upper bound with a logarithmic one. This comparison concerns upper bounds and does not guarantee a corresponding reduction in the realized quantization error.

Rounding randomness is independent across devices, so on the event that Equation 9 holds for every device, every block and every step,

$$\mathbb{E}_{\mathrm{SR}} \left[\|\mathbf{e}_t\|_2^2 \mid \mathcal{G}_t \right] = \sum_{p=1}^{P} \mathbb{E}_{\mathrm{SR}} \left[\|\mathbf{e}_t^{(p)}\|_2^2 \mid \mathcal{G}_t \right] \le \frac{\log(2bN_bPT/\delta)}{2q_w^2} \sum_{p=1}^{P} \|\mathbf{g}_t^{(p)}\|_2^2, \tag{16}$$

where a union bound over the P devices, the N_b blocks of the model and the T steps gives this event probability at least $1-\delta$ with $\delta' = \delta/(N_bPT)$. Note that $\sum_p \|\mathbf{g}_t^{(p)}\|_2^2$ is in general not equal to $\|\mathbf{g}_t\|_2^2$, since per-device gradients may partially cancel under aggregation.

Error Accumulation across Steps. Assume that rounding uses fresh randomness and is conditionally unbiased given $\mathcal{G}_t$, and that aggregation of the w-bit codes is exact in the accumulator used by the collective. Then

$$\mathbb{E}[\mathbf{e}_t \mid \mathcal{G}_t] = \mathbf{R}_t^\top \sum_p \mathbb{E} \left[\hat{\mathbf{y}}_t^{(p)} - \mathbf{R}_t \mathbf{g}_t^{(p)} \mid \mathcal{G}_t \right] = \mathbf{0}. \tag{17}$$

This property follows from stochastic rounding and holds for both static and dynamic keys. For $s < t$, $\mathbf{e}_s$ is measurable with respect to $\mathcal{G}_t$, so

$$\mathbb{E}\langle \mathbf{e}_s, \mathbf{e}_t \rangle = \mathbb{E}[\langle \mathbf{e}_s, \mathbb{E}[\mathbf{e}_t \mid \mathcal{G}_t] \rangle] = 0. \tag{18}$$

Consequently, assuming finite second moments, the accumulated error $\mathbf{E}_T = \sum_{t=1}^{T} \mathbf{e}_t$ satisfies

$$\mathbb{E}\|\mathbf{E}_T\|_2^2 = \sum_{t=1}^{T} \mathbb{E}\|\mathbf{e}_t\|_2^2. \tag{19}$$

If $\mathbb{E}\|\mathbf{e}_t\|_2^2 \le \sigma^2$ uniformly in t, then $\left(\mathbb{E}\|\mathbf{E}_T\|_2^2\right)^{1/2} \le \sigma\sqrt{T}$ for either key policy. Equation 16 supplies such a σ^2 whenever $\sum_p \|\mathbf{g}_t^{(p)}\|_2^2$ is bounded over training. This result concerns the sum of gradient quantization errors. It does not directly bound the deviation of optimizer trajectories.

Role of Fresh Rotation Keys. The theoretical distinction between the key policies concerns the applicability of the randomized distortion bound. When k_t is sampled independently of the history and of the current local gradients, conditioned on those gradients, Equation 15 applies to each local block at step t. With a static key, subsequent gradients may depend on that key through previous optimization steps, so the same fixed-input concentration argument does not directly apply. Under a static key the deterministic bound $\|\mathbf{y}_t^{(p)}\|_\infty \le \|\mathbf{g}_t^{(p)}\|_2$ continues to hold at every step, and the randomized bound applies at the step where the key is drawn, so the guarantee interpolates between the logarithmic and the linear dependence on b rather than being lost. This distinction does not imply that static keys produce biased errors or nonzero temporal error cross-covariances.

A.3 HADAMIZER-STATE: MOMENTUM DYNAMICS UNDER STATIC AND DYNAMIC ROTATION

As in the preceding sections, all statements are made for a single block and extend to the full model block-wise. Where a statement is required to hold simultaneously for every block, a union bound over the N_b blocks replaces δ by δ/N_b.

Let $\mathbf{m}_t, \mathbf{v}_t$ be the Adam moments of $\mathbf{g}'_t$ in the original coordinates, initialized at $\mathbf{m}_0 = \mathbf{v}_0 = \mathbf{0}$. HADAMIZER-STATE fixes a single key k_0 and forms the rotated gradient $\mathbf{r}_t = \tilde{\mathbf{H}}_{b,k_0}\mathbf{g}'_t$. The rotated moments $\tilde{\mathbf{m}}_t, \tilde{\mathbf{v}}_t$ follow Equation 6 and are likewise initialized at zero. The update is normalized element-wise in the rotated coordinates and then mapped back to the original coordinates,

$$\Delta\boldsymbol{\theta}_t = -\eta\, \tilde{\mathbf{H}}_{b,k_0}^{\top}\mathbf{u}_t, \qquad u_{t,j} = \frac{\tilde{m}_{t,j}}{\sqrt{\tilde{v}_{t,j}} + \epsilon}. \tag{20}$$

Here $\tilde{m}_{t,j}$ and $\tilde{v}_{t,j}$ are the first- and second-moment estimates in the rotated basis, η is the learning rate, and $\epsilon > 0$ stabilizes the denominator. The inverse rotation $\tilde{\mathbf{H}}_{b,k_0}^{\top}$ maps the normalized update back to the original parameter coordinates. Bias correction and decoupled weight decay are omitted for clarity.

Linearity of the First Moment under a Static Key. By induction, $\tilde{\mathbf{m}}_0 = \tilde{\mathbf{H}}_{b,k_0}\mathbf{m}_0$ and

$$\tilde{\mathbf{m}}_t = \beta_1\tilde{\mathbf{H}}_{b,k_0}\mathbf{m}_{t-1} + (1-\beta_1)\tilde{\mathbf{H}}_{b,k_0}\mathbf{g}'_t = \tilde{\mathbf{H}}_{b,k_0}\mathbf{m}_t, \tag{21}$$

so $\tilde{\mathbf{H}}_{b,k_0}^{\top}\tilde{\mathbf{m}}_t = \mathbf{m}_t$ exactly at every step, consistent with the near-zero state cosine distance in Table 2. The argument uses only linearity of the EMA and a fixed orthogonal map.

Loss of Momentum under Dynamic Keys. Suppose instead $\mathbf{r}_t = \tilde{\mathbf{H}}_{b,k_t}\mathbf{g}'_t$ with a fresh key k_t at every step, and define the read-back $\mathbf{q}_t = \tilde{\mathbf{H}}_{b,k_t}^{\top}\tilde{\mathbf{m}}_t$. Then $\tilde{\mathbf{m}}_t = \beta_1\tilde{\mathbf{m}}_{t-1} + (1-\beta_1)\mathbf{r}_t$ and, using $\tilde{\mathbf{H}}_{b,k_t}^{\top}\tilde{\mathbf{H}}_{b,k_t} = \mathbf{I}$,

$$\mathbf{q}_t = (1-\beta_1)\,\mathbf{g}'_t + \beta_1\,\tilde{\mathbf{H}}_{b,k_t}^{\top}\tilde{\mathbf{m}}_{t-1}. \tag{22}$$

Since $\tilde{\mathbf{m}}_{t-1}$ is independent of k_t and $\mathbb{E}[\tilde{\mathbf{H}}_{b,k_t}] = \frac{1}{\sqrt{b}}\mathbf{H}_b\,\mathbb{E}[\mathbf{D}_{k_t}] = \mathbf{0}$, we have $\mathbb{E}[\mathbf{q}_t \mid \mathcal{G}_t^{-}] = (1-\beta_1)\mathbf{g}'_t$, where $\mathcal{G}_t^{-}$ is the conditioning of Section A.2 with the key at step t excluded, and the history term is zero-mean noise of norm $\beta_1\|\tilde{\mathbf{m}}_{t-1}\|_2$. Only the $(1-\beta_1)$ fraction carried by the current gradient survives in expectation, consistent with the larger state cosine distance under dynamic keys in Table 2.

Second Moment as a Block-Diagonal Preconditioner. Due to the nonlinearity of $\odot$, $\tilde{\mathbf{v}}_t \neq \tilde{\mathbf{H}}_{b,k_0}\mathbf{v}_t$. Substituting the first-moment identity into Equation 20 gives

$$\Delta\boldsymbol{\theta}_t = -\eta\,\mathbf{P}_t\,\mathbf{m}_t, \qquad \mathbf{P}_t = \tilde{\mathbf{H}}_{b,k_0}^{\top}\,\mathrm{diag}\left(\sqrt{\tilde{\mathbf{v}}_t}+\epsilon\right)^{-1}\tilde{\mathbf{H}}_{b,k_0}, \tag{23}$$

where $\mathbf{P}_t$ is block-diagonal with $b \times b$ symmetric positive definite blocks whose eigenvectors are the columns of $\tilde{\mathbf{H}}_{b,k_0}^{\top}$ and whose eigenvalues are $(\sqrt{\tilde{v}_{t,j}}+\epsilon)^{-1}$. ADAM is the special case $\tilde{\mathbf{H}}_{b,k_0} = \mathbf{I}$. Two properties of $\tilde{\mathbf{v}}_t$ follow. First, total mass per block is conserved, $\mathbf{1}^{\top}\tilde{\mathbf{v}}_t = \mathbf{1}^{\top}\mathbf{v}_t$, since $\|\mathbf{r}_s\|_2^2 = \|\mathbf{g}'_s\|_2^2$ for every s and the EMA is linear in $\mathbf{r}_s \odot \mathbf{r}_s$. Writing $\bar{v}_t = \frac{1}{b}\mathbf{1}^{\top}\tilde{\mathbf{v}}_t$, which by conservation equals $\frac{1}{b}\mathbf{1}^{\top}\mathbf{v}_t$, the rotation redistributes the second-moment mass within each block without changing its total. Second, the within-block maximum-to-mean ratio is bounded. For a fixed gradient sequence independent of the random key k_0, applying Equation 9 to every $\mathbf{g}'_s$ with a union bound over the t steps gives

$$\tilde{v}_{t,j} = (1-\beta_2)\sum_{s\le t}\beta_2^{t-s}r_{s,j}^2 \le 2\log(2bt/\delta)\,(1-\beta_2)\sum_{s\le t}\beta_2^{t-s}\,\mu_2(\mathbf{g}'_s) = 2\log(2bt/\delta)\;\bar{v}_t \tag{24}$$

with probability at least $1-\delta$ for this block, and with $\log(2bN_bt/\delta)$ in place of $\log(2bt/\delta)$ if the statement is required simultaneously for all N_b blocks. By comparison, an unrotated second-moment vector concentrated on a single coordinate, $\mathbf{v}_t = (a, 0, \ldots, 0)$ with $a > 0$, has mean a/b and hence $\max_j v_{t,j}\big/\left(\frac{1}{b}\mathbf{1}^{\top}\mathbf{v}_t\right) = b$. This logarithmic rather than linear maximum-to-mean bound limits the influence of concentrated second-moment values on the quantization scale. In the isotropic limit $\tilde{\mathbf{v}}_t = \bar{v}_t\mathbf{1}$, the preconditioner reduces to $\mathbf{P}_t = (\sqrt{\bar{v}_t}+\epsilon)^{-1}\mathbf{I}_b$, and the update becomes momentum SGD with a per-block adaptive scalar step. More generally, the anisotropy of each block is characterized by

$$\mathrm{cond}_2(\mathbf{P}_t) = \frac{\sqrt{\max_j \tilde{v}_{t,j}}+\epsilon}{\sqrt{\min_j \tilde{v}_{t,j}}+\epsilon}. \tag{25}$$

Momentum SGD has condition number one, while ADAM has the analogous expression with $\mathbf{v}_t$ in place of $\tilde{\mathbf{v}}_t$. The maximum-to-mean bound above controls the concentration of the rotated second moments, but does not by itself establish a condition number close to one or an ordering relative to ADAM. Thus, HADAMIZER-STATE defines a block-diagonal preconditioned optimizer whose eigenvectors are fixed by the Hadamard basis and whose anisotropy is determined by the rotated second moments, rather than a reparameterization of ADAM.

B SUPPLEMENTARY RESULTS

B.1 HYPER-PARAMETERS AND FINE-TUNING RESULTS

For all experiments, we use a sequence length of 1,024 tokens. Batch sizes are chosen to accommodate model-specific memory requirements: 512 for GEMMA3-270M, 256 for GEMMA3-1B, QWEN2.5-0.5B, and QWEN2.5-1.5B, and 128 for QWEN2.5-3B. The learning rates are proportional to batch size, with model-specific variation around $\eta = 2 \times 10^{-4}(B/256)$, where B denotes the batch size. This choice accounts for the more frequent optimizer updates per training token at smaller batch sizes by using correspondingly smaller learning rates. Specifically, we use learning rates of 4×10^{-4}, 2×10^{-4}, 2.1×10^{-4}, 1.5×10^{-4}, and 10^{-4} for GEMMA3-270M, GEMMA3-1B, QWEN2.5-0.5B, QWEN2.5-1.5B, and QWEN2.5-3B, respectively. Within each model, all evaluated methods use the same batch size and learning rate. All methods follow a cosine learning-rate decay schedule with 5% warmup.

Table 8 summarizes the final cross-entropy on C4/EN pre-training. HADAMIZER-STATE consistently improves upon FP32 ADAM across all five models, reducing cross-entropy by 3.97–5.44%. These quality gains do not necessarily come at the expense of execution efficiency. On QWEN2.5-3B, HADAMIZER-STATE reduces TPU step time from 591.03 ms for HALO to 546.31 ms while also lowering cross-entropy from 3.6786 to 3.6569, corresponding to an 8.19% throughput improvement. HADAMIZER-GRAD provides a complementary operating point that favors execution efficiency, enhancing throughput over HALO on four of the five models and reaching up to a 9.58% improvement across the evaluated configurations. Together, the two variants demonstrate the target-specific design of HADAMIZER: the static-state path prioritizes optimization quality while remaining competitive

Table 8: Final cross-entropy on the C4/EN pre-training workloads. Parentheses report the relative change from FP32 ADAM for each model, with negative values indicating improvement.

	GEMMA3-270M	GEMMA3-1B	QWEN2.5-0.5B	QWEN2.5-1.5B	QWEN2.5-3B
ADAM (FP32)	3.5679	3.5964	3.7150	3.5876	3.8628
HALO	3.4284 (−3.91%)	3.3898 (−5.74%)	3.5370 (−4.79%)	3.3898 (−5.51%)	3.6786 (−4.77%)
Muon	3.9602 (+11.00%)	3.9786 (+10.63%)	4.4158 (+18.86%)	4.7212 (+31.60%)	5.1288 (+32.77%)
HADAMIZER-GRAD	3.8120 (+6.84%)	3.9105 (+8.73%)	4.0973 (+10.29%)	3.9742 (+10.78%)	4.2985 (+11.28%)
HADAMIZER-STATE	3.4262 (−3.97%)	3.4008 (−5.44%)	3.5239 (−5.14%)	3.4316 (−4.35%)	3.6569 (−5.33%)

Table 9: Final cross-entropy on the MTNT/EN-FR fine-tuning workloads. Parentheses report the relative change from FP32 ADAM for each model, with negative values indicating improvement.

	GEMMA3-270M	GEMMA3-1B	QWEN2.5-0.5B	QWEN2.5-1.5B	QWEN2.5-3B
ADAM (FP32)	7.0838	7.2096	7.2905	7.2905	7.7130
HALO	7.0928 (+0.13%)	7.1647 (−0.62%)	7.1557 (−1.85%)	7.1467 (−1.97%)	7.5377 (−2.27%)
Muon	7.4973 (+5.84%)	7.5243 (+4.37%)	8.0996 (+11.10%)	8.3153 (+14.06%)	8.9131 (+15.56%)
HADAMIZER-GRAD	7.3085 (+3.17%)	7.5153 (+4.24%)	7.5692 (+3.82%)	7.7310 (+6.04%)	8.0232 (+4.02%)
HADAMIZER-STATE	7.0568 (−0.38%)	7.1017 (−1.50%)	7.2006 (−1.23%)	7.1827 (−1.48%)	7.5647 (−1.92%)

in throughput, whereas the dynamic-gradient path provides a more performance-oriented operating point.

Table 9 summarizes the final cross-entropy on fine-tuning MTNT/EN-FR. HADAMIZER-STATE improves upon FP32 ADAM across all five models, reducing cross-entropy by 0.38–1.92% with the largest relative reduction on QWEN2.5-3B. In contrast, HADAMIZER-GRAD increases cross-entropy relative to FP32 ADAM by 3.17–6.04%, still outperforming MUON across all five models. These results show that HADAMIZER-STATE consistently improves fine-tuning cross-entropy over FP32 ADAM, whereas HADAMIZER-GRAD incurs a quality penalty.

B.2 ROTATION SCALABILITY

Table 10 shows the extended rotation overhead results when sweeping the training system size. The S-A choice is faster than global rotation across all evaluated configurations, achieving up to a 2.11× speedup for the 1.5B model on an 8×8 mesh. Because S-A rotation operates entirely within each device-local shard, its cost does not include collective communication and is determined primarily by the local shard width ℓ and block size b. A few highly sharded configurations fall outside this regime: for GEMMA3-270M at 32 and 64 devices, and for GEMMA3-1B and QWEN2.5-0.5B at 64 devices, ℓ decreases to 32–108, causing the rotation block to exceed the local shard boundary. In the target regime of $d/p \geq 128$, however, the rotation fits entirely within each shard and covers all remaining configurations. This accentuates that shard-aligned rotation avoids the mesh-dependent collective overhead of global rotation, while its cost is instead determined by the resulting per-device shard size.

Table 10: Rotation kernel latency in microseconds with respect to platform scalability on the feed-forward-network.

	4×4		4×8		8×8	
Model	Glob.	S-A	Glob.	S-A	Glob.	S-A
GEMMA3-270M	77.10	**71.30**	87.96	**76.96**	78.89	**72.76**
GEMMA3-1B	183.19	**107.26**	185.78	**110.10**	201.76	**112.21**
QWEN2.5-0.5B	155.51	**89.20**	159.68	**85.76**	155.79	**86.84**
QWEN2.5-1.5B	277.80	**138.72**	278.30	**142.03**	281.98	**133.75**
QWEN2.5-3B	357.23	**198.75**	353.94	**201.36**	361.61	**202.30**

B.3 QUANTIZATION FIDELITY

Table 11 characterizes the reconstruction fidelity of the parameter update under Hadamard-rotated quantization at INT8 and INT4 precision across five models on C4/EN and MTNT/EN-FR. We use signal-to-quantization-noise ratio (SQNR) to quantify the magnitude of the perturbation introduced by quantization relative to the full-precision signal. Specifically, we measure SQNR between the FP32 parameter update and the update reconstructed from the quantized gradient and optimizer states.

INT8 SQNR ranges from 42.50 to 42.63 dB, while INT4 SQNR ranges from 18.60 to 18.74 dB. The narrow variation at each precision indicates similar update fidelity across the evaluated model–workload pairs. Reducing precision to INT4 lowers SQNR by approximately 24 dB, quantifying the additional distortion introduced into the parameter update by more aggressive quantization. Unlike prior work that evaluates SQNR on quantized weights or layer outputs, our measurement is performed on the reconstructed optimizer update, after both the gradient and optimizer states have undergone the Hadamard-rotated quantization path. These measurements provide an initial characterization of extending HADAMIZER to INT4 and indicate a clear opportunity for precision-specific rotation and quantization schemes that further improve update fidelity under more aggressive compression.

Table 11: SQNR (dB) of Hadamard-rotated quantization at INT8 and INT4 precision. Higher values indicate greater reconstruction fidelity.

	C4/EN		MTNT/EN-FR	
Model	**INT8**	**INT4**	**INT8**	**INT4**
GEMMA3-270M	42.5038	18.6083	42.5726	18.7384
GEMMA3-1B	42.502	18.6107	42.4976	18.671
QWEN2.5-0.5B	42.6322	18.6459	42.6299	18.5999
QWEN2.5-1.5B	42.5029	18.6094	42.5031	18.6094
QWEN2.5-3B	42.5017	18.6094	42.5021	18.6096