HyperQuant: A Rate–Distortion-Optimal Quantization Pipeline for Large Language and Diffusion Models

Yuval Domb Hadar Sackstein Tomer Solberg
research@moonmath.ai

Abstract

We present HyperQuant (Hadamard, optimallY Packing, Entropy Rice-coding), a unified post-training quantization pipeline for the linear layers and the KV cache of large language and diffusion transformers. Across a suite of self-contained experiments (Table 1), HyperQuant outperforms the recent HIGGS quantization scheme at every operating point from 3 to 5 bits per scalar (bps), as well as beating both TurboQuant and OCTOPUS at KV quantization, down to 1.7 bps. Outside the LLM regime, we apply HyperQuant to the 19B-parameter LTX-2 DiT video model with no observable per-frame artifacts. End-to-end wall-clock speedups over bf16 are pending a fused Triton kernel benchmark that decodes HyperQuant's lattice codes directly into the MMA input format **[Pending: measured kernel speedups in progress]**.

Under the hood, HyperQuant combines four known ideas we combine together into a single construction: (i) a per-tile Walsh–Hadamard rotation that whitens weights and activations to approximate multivariate-Gaussian statistics, (ii) a low-dimensional optimal lattice (E_8, D_4, A_2, or $\mathbb{Z}$) quantization, (iii) a near-entropy-optimal variable-length Rice coding scheme over lattice indices, and (iv) various bias-correction methods for the KV cache, that keep the reconstruction unbiased under inner products, preserving attention semantics. We further integrate the pipeline with 8-bit and 4-bit Tensor-Core MMA paths (FP8-E4M3, INT8, NVFP4, MXFP4), and we find that INT8 beats FP8 on the whitened lattice output.

Table 1: Typical HyperQuant operating points across settings. Weights/KV/INT8-MMA rows are Llama-3.1-8B-Instruct on WikiText-2; the OCTOPUS row is KV-only on Qwen2.5-7B-Instruct (perplexity (PPL) Δ% at 32-token residual window); LTX-2 is the 19B DiT video model.

Setting	rate	criterion	HyperQuant	reference
Weights+KV cache, INT8 MMA	4 bps	PPL ↓	7.50 (+0.47%)	7.16 (bf16)
Weights	4 bps	ΔPPL% ↓	+**3.8**%	+6.4% (HIGGS)
Weights	3 bps	ΔPPL% ↓	+**22.1**%	+33% (HIGGS)
KV cache	2 bps	ΔPPL% ↓	+**7.4**%	+34.7% (OCTOPUS)
KV cache	2 bps	compression ↑	**6.4**×	3.0× (TurboQuant)
KV cache	1.7 bps	ΔPPL% ↓	+**26.9**%	–
LTX-2 video	4 bps	LPIPS ↓	0.20–0.21	0 (bf16)

Contents

1 Introduction

Frontier large-language and generative models [1, 18] routinely exceed tens of billions of model weight parameters and produce KV caches that dominate inference memory at modern context lengths. Both effects make autoregressive decoding overwhelmingly memory-bandwidth-bound such that each generated token must stream the entire weight set and the full KV cache from global memory while performing only a thin matrix-vector product. As a result, the arithmetic intensity stays far below the hardware's compute-to-bandwidth ratio and the Tensor Cores sit largely idle waiting on memory [31]. Post-training quantization (PTQ) turns this bandwidth bottleneck into a *rate-distortion* data compression problem.

For model weights quantization, a long line of work including GPTQ [14], AWQ [19], SmoothQuant[38], OmniQuant [36], has chipped away at the rate, typically at the cost of a calibration dataset and per-layer optimization. Data-free schemes are simpler and are generally preferred for deployment. The recent state of the art among them, HIGGS [23], identifies two specific levers: (1) Hadamard-rotate weights to make their marginal distribution approximately Gaussian, and (2) quantize to multi-dimensional codebooks that are MSE-optimal for that Gaussian. HIGGS's headline result, a linearity theorem reducing global perplexity damage to per-layer ℓ_2 error, justifies focusing the design on per-layer MSE.

But HIGGS leaves an opening. Its codebooks are finite Lloyd grids with *fixed-rate* indices. Information theory predicts that this fixed-rate overhead grows as the codebook gets finer (a.k.a. shaping gain) and that an *entropy-coded* quantizer with the same MSE will always need fewer bits [22]. On real LLM weights, our measurements (Section 6.1) confirm this gap is real. HIGGS's natural index entropy is 0.6–5.9 % below its fixed bit budget at 3-5 bps. Our way to close this gap is well known in classical lattice coding [7, 39]. Combine a *lattice* quantizer with variable-length encoder over its indices.

The quantization of the KV-cache has converged on a different rotation-plus-marginal scheme. TurboQuant [42] rotates each head's KV vector, exploits the Beta marginal of the resulting unit-norm coordinates, and Lloyd-quantizes each resulting scalar. OCTOPUS [3] extends the marginal trick to coordinate *triplets* via an octahedral parameterization, gaining nearly an extra bit at 2 bps regimes. Both methods are data-oblivious, yet they both pay the same fixed-rate overhead that HIGGS does, just on a different marginal. On the lattice side, the closest contemporary is NestQuant [34], which uses nested Gosset lattices plus a calibration-style QA-LDLQ correction. Quantizing both model weights and KV cache to 4 bps, it raises Llama-3-8B perplexity by $\sim 3.9\%$ over its bf16 baseline.

Contributions. We propose HyperQuant (Hadamard, optimallY Packing, Entropy Rice-coding), a single calibration-free, post-training pipeline that applies the rate-distortion-optimal triplet of whitening, lattice quantization, and entropy coding to both the model weights and KV cache, integrated with Hopper's fp8/int8 and Blackwell's nvfp4/mxfp4 MMA paths [25–27], and benchmarked end-to-end on Llama-3.1-8B-Instruct and LTX-2-19B (Sections 4 and 5). Our contributions are:

- **Per-tile Random Hadamard Transform (RHT)**: each linear layer's input and weight are Hadamard-rotated in blocks of size matched to the lattice dimension and to the hardware's MMA tile; the rotation is folded into the previous LayerNorm/RMSNorm where possible (no extra runtime cost) and otherwise installed as a forward hook.
- **Lattice + Rice coding**: we quantize the rotated tile with one of E_8 (8-D), D_4 (4-D), A_2 (2-D), or scalar $\mathbb{Z}$, then encode the resulting integer code with a Rice code calibrated on the per-norm Gaussian distribution, realizing any target bit-rate at $\sim$0.01-bps precision (Section 6.1).

- **A bias-correction menu** for the KV cache, a per-layer random rotation (± 1 signs or full QJL) and optional Schuchman subtractive dither [35, 40], together with a formal proof (Section A) that subtractive lattice dither is *strictly* inner-product unbiased on every cached vector, contrasted with the only distribution-average unbiasedness of QJL's 1-bit sketch [41].
- A *rate–distortion decomposition* of the gap between HyperQuant and HIGGS at matched bps (Section 6.1): at b=4 a ~ 0.75 dB piece is HIGGS's fixed-rate index redundancy (recoverable by any entropy-coded retrofit), and a larger ~ 0.91 dB piece is the structural advantage of an unbounded codebook over any finite one (enabled by, and inseparable from, variable-length coding). The latter piece grows to ~ 2.34 dB at b=5 while the index-entropy piece shrinks to ~ 0.18 dB.
- A clean two-regime picture of KV-cache quantization quality (Section 6.2): a high-quality regime (bps ≥ 2.5) where all bias-correction variants are within 0.04 PPL, and a high-compression regime (1.7–2.5 bps) where QJL-style rotation pulls ahead by up to ~ 0.5 PPL.
- An end-to-end stress test on the 19B-parameter LTX-2 DiT video model, showing that the same pipeline transfers to a non-LLM transformer architecture and delivers 3.7× weight compression with no perceptible quality loss (Section 6.5).

Outline. Section 3 reviews RHT, optimal low-dimensional lattices, Rice coding, and dithering. Section 4 outlines the HyperQuant design, and Section 5 covers its implementation. Sections 6 and 7 present benchmark comparisons against HIGGS, TurboQuant, and OCTOPUS, and per-component ablations. Section 8 concludes our work and suggests future directions.

2 Related work

Weight quantization with rotations and finite codebooks. QuIP [6] introduced *incoherence processing*, essentially a randomized rotation that whitens the weight matrix before quantization, and showed it enables 2-bit weight quantization with provable guarantees. QuIP# [37] sharpened this with an E_8-lattice codebook and a quasi-incoherent transform. QuaRot [5] and SpinQuant [20] extended the rotation idea to *learned* 4-D rotations applied online to activations, achieving end-to-end 4-bit inference. HIGGS [23] crystallized the data-free version of this paradigm: Hadamard rotation plus a multi-dimensional Lloyd codebook trained on a Gaussian prior. HyperQuant shares the rotation+codebook architecture but differs in two structural respects: the codebook is an infinite lattice (so the codeword density is set by a continuous SNR knob, not a discrete codebook size), and the index stream is *entropy-coded* via Rice coding rather than transmitted at fixed $\log_2 N$ bits per index. Both differences are explained quantitatively in Section 6.1.

Calibration-based methods. GPTQ [14], AWQ [19], OmniQuant [36], SmoothQuant [38], SpQR [10], and SliceGPT [4] use calibration data to refine per-channel scales, salvage outlier features, or solve a Hessian-aware weight allocation problem. HyperQuant is calibration-free by design and sits orthogonal to these methods; a future direction is to compose HyperQuant's bit-allocation knob with an LDLQ-style calibration update [34, 37].

KV-cache quantization. Static per-channel methods like KIVI [21] and KVQuant [17] reach 2-bit KV by using outlier-aware mixed precision. QJL [41] introduced a 1-bit data-oblivious quantizer based on a Quantized Johnson–Lindenstrauss sketch, with approximate unbiased inner products. TurboQuant [42] replaced the QJL sketch by an exact random rotation plus Lloyd-Max scalar

quantization on the Beta-distributed unit-vector coordinates, reaching 4–7× KV-cache compression with ∼ zero quality loss on Gemma and Mistral. OCTOPUS [3] jointly quantizes triplets of rotated coordinates via an octahedral parameterization, extending the rate–distortion curve to extreme ($\leq$ 2-bit) operating points. All three methods stay strictly scalar (or 3-D scalar) after rotation; HyperQuant instead uses true multi-dimensional lattices (E_8 is 8-D) addressed by a variable-length code, which jointly delivers a higher granular gain and an unbounded codebook – worth ~ 0.91 dB over even an entropy-coded HIGGS at b=4, growing to ~ 2.34 dB at b=5 (Section 6.1). On the bias side, TurboQuant inherits QJL's distribution-average unbiasedness; the only strictly per-vector-unbiased construction we know in this space is subtractive dither [11, 35, 40], which we adopt and prove correct on the full pipeline.

Nested-lattices. NestQuant [34] is the closest contemporary work in spirit: like HyperQuant it uses the E_8 Gosset lattice as the quantizer, and like HyperQuant it is motivated by information-theoretic optimality (matrix-product rate–distortion bounds). NestQuant's practical performance, however, relies on a calibration-style QA-LDLQ post-processing that performs a Cholesky-style update on the unquantized weights based on the quantized activation noise. HyperQuant keeps the data-free property and offsets the absence of that calibration step with an entropy code and a richer rotation menu (none/signs/Haar). Because the two works use different base models and eval harnesses, absolute PPL is not comparable; on a baseline-normalized basis HyperQuant's data-free weight+KV path is competitive in the matched W+KV regime: its E_8 path at 4 bps costs +4.6% on Llama-3.1-8B, within a fraction of a point of NestQuant's +3.9% W+KV at $\sim$ 4 bps, which is obtained *with* QA-LDLQ calibration. Their own ablation makes the calibration's role explicit: without QA-LDLQ, NestQuant's W+KV path PPL cost rises to +7.6%.

Diffusion and video transformers. Quantization of diffusion transformers [12, 18, 30] is less explored than LLM quantization, with most published numbers focused on image rather than video models. The LTX-2 [18] stress-test in Section 6.5 is, to our knowledge, the first end-to-end PTQ result on a billion-parameter video DiT, complementing earlier OCTOPUS results [3] on the Wan-1.3B DiT.

Numerical formats. FP8 was standardized by NVIDIA's H100 [25, 26]; the smaller FP4 formats (NVFP4 and the OCP MX-FP4) target the Blackwell generation [27, 28, 33]. The key practical difference between the two FP4 formats is the scale encoding (FP8-E4M3 in NVFP4 vs. power-of-2 E8M0 in MX-FP4) and the block size (16 vs. 32). HyperQuant targets both formats; our experiments show NVFP4 is the only one quality-viable for KV quantization (Section 6.4).

3 Preliminaries

This section reviews the four classical ingredients that the remainder of the paper builds on: the randomized Hadamard transform, optimal low-dimensional lattices as vector quantizers, Rice entropy coding, and the subtractive-dither and random-rotation bias-correction mechanisms. All material here is well-established and is included to fix notation and conventions; the design choices specific to HyperQuant are deferred to Section 4.

3.1 Randomized Hadamard transform (RHT)

For dimensions n that are a power of two, the $n \times n$ Walsh–Hadamard matrix H_n is recursively defined by $H_1 = (1)$ and $H_{2n} = \frac{1}{\sqrt{2}}\left(\begin{smallmatrix} H_n & H_n \\ H_n & -H_n \end{smallmatrix}\right)$. The columns of H_n are an orthonormal basis, so $H_n^\top H_n = I_n$. The transform admits an $O(n \log n)$ Cooley–Tukey-style butterfly implementation.

The *randomized* Hadamard transform composes H_n with a random sign diagonal $D = \mathrm{diag}(\pm 1)$:

$$\mathrm{RHT}_n(x) \;=\; H_n D\, x, \qquad D_{ii} \in \{-1, +1\} \text{ iid uniform.} \tag{1}$$

Two facts are central. First, RHT is a fast Johnson–Lindenstrauss-style mixer: applying $H_n D$ to any deterministic vector $x \in \mathbb{R}^n$ produces a vector whose *empirical* distribution is, with high probability, close to $\mathcal{N}(0, \|x\|^2/n \cdot I)$ [2, 9, 16]. Second, because RHT_n is orthogonal, applying it before quantization and inverting it after preserves ℓ_2 error: the lattice cell volume measured in the pre-RHT space equals the cell volume measured post-RHT. RHT therefore *redistributes* the per-coordinate quantization error from being concentrated on a few outlier channels (the raw activation space) into being approximately isotropic [6, 23].

3.2 Optimal low-dimensional lattices

A lattice $\Lambda \subset \mathbb{R}^n$ is the set of integer linear combinations of n basis vectors. Two lattice invariants characterize its quality as a vector quantizer:

- the *normalized second moment*

$$G(\Lambda) := \frac{1}{n}\, \mathbb{E}_{U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))}\big[\|U\|^2\big] \cdot \det(\Lambda)^{-2/n},$$

  i.e. the per-coordinate mean-squared error of quantizing a uniformly distributed point in the Voronoi cell $\mathcal{V}(\Lambda)$ to the origin, rendered scale-invariant by the $\det(\Lambda)^{-2/n}$ normalization. $G(\Lambda)$ is bounded below by $1/(2\pi e)$, the value attained asymptotically by the n-dimensional ball, and a smaller $G(\Lambda)$ translates directly into lower granular distortion of a nearest-neighbor quantizer at fixed rate [7, 39]. Intuitively, $G(\Lambda)$ measures how "round" the Voronoi cell is.

- the *packing density* $\Delta(\Lambda)$, the fraction of $\mathbb{R}^n$ covered when non-overlapping balls of radius $r_{\mathrm{pack}}(\Lambda)$ (the inradius of the Voronoi cell, equal to half the lattice minimum distance) are placed at every lattice point. A higher $\Delta(\Lambda)$ means more codewords fit into a region of $\mathbb{R}^n$ at a fixed minimum separation, so this invariant controls how densely the codebook tiles space when the cell radius is held fixed [7].

In every dimension ≤ 8 for which the optimum is known, the densest sphere packing also achieves the smallest known $G(\Lambda)$: $\mathbb{Z} = \mathbb{Z}$ in 1-D, the hexagonal A_2 in 2-D, the Schläfli lattice D_4 in 4-D, and the Gosset lattice E_8 in 8-D [7]. Table 2 collects the classical normalized second moments and the resulting high-rate gap to the Shannon bound for these four lattices.

Nearest-neighbor decoding. For A_2, D_4, and E_8, the closest-lattice-point algorithm follows Conway–Sloane [7, Ch. 20]: round each coordinate to the nearest integer; if the result violates the lattice's parity constraint, move to the nearest point of the complementary coset (for E_8 the half-integer coset $D_8 + \frac{1}{2}\mathbf{1}$, for D_4 and A_2 the parity-flipped neighbor), and keep whichever candidate has the smaller residual norm. The total work per scalar is $O(1)$. E_8 is the highest-dimensional lattice for which nearest-neighbor decoding admits such a constant-time closed-form implementation [7].

Lattice	n	$G(\Lambda)$	High-rate gap to Shannon	Decoder $O(\cdot)$
$\mathbb{Z}$	1	$1/12 = 0.0833$	1.53 dB	$O(1)$
A_2	2	0.0802	1.36 dB	$O(1)$
D_4	4	0.0766	1.17 dB	$O(1)$
E_8	8	0.0717	0.88 dB	$O(1)$
∞-D sphere	—	$1/(2\pi e) = 0.0586$	0 dB	—

Table 2: Classical lattice quantizer constants for the four lattices used in this paper. The high-rate gap to the Shannon bound is $10\log_{10}(G(\Lambda)\cdot 2\pi e)$. The asymptotic limit $1/(2\pi e)$ is the infinite-dimensional sphere bound. These normalized second moments are scale-invariant, so they apply unchanged to the integer realizations $E_8^{\text{int}}/D_4^{\text{int}}/A_2^{\text{int}}/\mathbb{Z}_1^{\text{int}}$ used in our code.

Granular gain. The fundamental advantage of multi-dimensional vector quantization (VQ) over scalar quantization comes from the freedom to choose the shape of the quantization cell. A scalar quantizer's cell is forced to be an interval, the Voronoi cell of $\mathbb{Z}$, so its normalized second moment is fixed at $G(\mathbb{Z}) = 1/12 \approx 0.0833$. In higher dimensions the optimal Voronoi cell becomes progressively rounder, and the value of G descends toward the ball's limit $G_\infty = 1/(2\pi e) \approx 0.0586$. The ratio $10\log_{10}(G(\mathbb{Z})/G(\Lambda))$ is the *granular gain* of Λ over the scalar quantizer; it is the source-coding counterpart of the (channel-coding) shaping gain familiar from constellation design [13, 39].

In dB terms, the entire budget available to a quantizer, $\mathbb{Z}$'s gap to the Shannon bound, is 1.53 dB. The four lattices used in this paper traverse this budget unevenly: A_2 recovers 0.17 dB, D_4 recovers 0.37 dB, and E_8 recovers 0.65 dB, leaving 0.88 dB between E_8 and the Shannon bound. So E_8 closes roughly 42% of the gap, not the bulk of it: a non-trivial residual remains, but extracting it requires substantially higher-dimensional lattices with much more expensive decoders. The best lattice we know in any dimension, the 24-D Leech lattice, sits at $G \approx 0.0658$, which is only a further ~ 0.38 dB closer to Shannon than E_8 [7, 39]. E_8 is therefore not where the gap closes, but rather where the gain-per-decoder-complexity curve sharply flattens.

3.3 Entropy coding and Rice codes

Entropy coding. For a discrete source X with probability mass function p, Shannon's source coding theorem states that the average code length per symbol of any uniquely decodable code is bounded below by the entropy

$$H(X) \;=\; -\sum_x p(x)\,\log_2 p(x),$$

and that this lower bound is achievable to within a fraction of a bit per symbol by practical entropy coders such as Huffman or arithmetic coding [8]. Entropy is therefore the fundamental rate floor for *lossless* compression of a discrete source. In a *lossy* pipeline like ours the picture splits cleanly into two stages: a quantizer first maps a continuous input to a discrete index, trading distortion for the bit count of that index (the rate–distortion behavior of the quantizer), and a lossless entropy coder then represents the resulting index stream at an expected rate close to its entropy. The entropy coder neither distorts the source nor alters the quantizer's distortion characteristics; it only realizes the information-theoretic floor on the indices in actual bits.

Variable-length codes and unbounded alphabets. A fixed-length code over an alphabet of size N pays exactly $\log_2 N$ bits per symbol and is undefined when $N = \infty$. A variable-length code, in contrast, has no such limitation: it addresses an arbitrary discrete alphabet at finite expected

rate whenever the source's entropy is finite. This is the operative advantage of variable-length coding over fixed-length coding in our setting: a lattice quantizer's output is an integer vector with unbounded support, so a fixed-length code is not even well-defined, yet for Gaussian-like inputs the entropy of the integer histogram is finite and a variable-length code attains it at finite cost [8, 39]. We separate and quantify this advantage empirically in Section 6.1.

Rice codes. Within the family of variable-length codes, the Rice code [32] is the practical near-optimal choice for sources whose integer histogram is (two-sided) geometric, i.e. Laplacian on the integer lattice; it is the power-of-two-parameter specialization of the Golomb code that is optimal in this regime. Given a Rice parameter k, a non-negative integer m is encoded as $\lfloor m/2^k \rfloor$ in unary followed by $m \bmod 2^k$ in k raw bits; signed values are handled by zig-zag interleaving or an explicit sign bit. The optimal k for a geometric distribution with parameter p is

$$k^* \;=\; \max\!\big(0,\; \lceil \log_2(-\ln(1-p)/\ln 2) \rceil\big).$$

The integer histograms we actually encounter (lattice indices of randomized-Hadamard-transformed weights and activations) are not exactly Laplacian, but they are close enough that a Rice code with empirically calibrated k stays within ~ 0.1 bps of the symbols' marginal entropy across our calibration sweep (Section B.1). We therefore adopt it as the entropy coder throughout: it has constant per-codeword cost, it is a genuine variable-length code over $\mathbb{Z}$, and the ~ 0.1 bps it concedes to an ideal marginal coder buys a stateless, table-free $O(1)$ decoder, while no *context* coder can do better, since the stripped symbols carry essentially no inter-symbol redundancy (Section B.4).

3.4 Bias correction: rotation and subtractive dither

A nearest-neighbor lattice quantizer is *deterministic* given its input and therefore biased: the reconstruction $\hat{x} = Q_\Lambda(x)$ satisfies $\hat{x} = x + e(x)$ with $e(x) \in -\mathcal{V}(\Lambda)$, which is a non-zero, x-dependent function. For weight quantization this is not a problem: biased reconstructions can be absorbed by surrounding affine parameters and disappear into the linearity theorem [23]. For KV-cache quantization, however, the attention operation

$$\text{Attention}(q,k,v) = \text{softmax}\big(\tfrac{1}{\sqrt{d}} q^\top k\big) v$$

is *linear in k and v inside the dot product*, so a deterministic bias in k accumulates across the softmax denominator and changes attention scores systematically.

Two classical mechanisms can remove this bias.

Random rotation (QJL-style). Apply a Haar-uniform random orthogonal matrix $S \sim \text{Uniform}(\mathrm{O}(n))$ to the cached vector before quantization and $S^\top$ to the reconstruction after:

$$\hat{x}_{\text{rot}}(x; S) \;=\; S^\top Q_\Lambda(S\,x). \tag{2}$$

Averaged over S, the error $e_{\text{rot}}(x) = -S^\top \pi_\Lambda(Sx)$ is zero-mean and isotropically distributed [41, 42]. But in deployment S is drawn once per layer and frozen, so the error is deterministic given (x, S_0) and biased on every individual cached vector. We formalize this in Proposition 2 and contrast it with strict unbiasedness.

Subtractive dither (Schuchman–Zamir–Feder). Draw $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$ fresh on every forward call, independently of everything else, and reconstruct

$$\hat{x}_{\mathrm{dith}}(x; U) \;=\; Q_\Lambda(x + U) - U. \tag{3}$$

The error $e_{\mathrm{dith}}(x; U) = -\pi_\Lambda(x + U)$ is then *exactly uniform* on $-\mathcal{V}(\Lambda)$, independent of x, by the Crypto Lemma [35, 40]; see the self-contained proof in Section A. In particular:

$$\mathbb{E}_U\big[\langle q,\, \hat{x}_{\mathrm{dith}}(x; U)\rangle \,\big|\, x\big] \;=\; \langle q,\, x\rangle \qquad \forall\, q, x \in \mathbb{R}^n. \tag{4}$$

We refer to this as *strict, per-vector* inner-product unbiasedness, in contrast to QJL's averaged-over-S unbiasedness.

Composing rotation and dither. The two mechanisms are statistically orthogonal: a rotation is a deterministic-given-x linear map, and dither is independent of x, so the inner-product unbiasedness of (4) survives composition with any orthogonal pre-rotation and any further deterministic-given-x linear post-processing (see Proposition 1). The composed scheme retains the isotropic-error property of the rotation and the strict unbiasedness of the dither. Standard alternatives to Haar-uniform rotations include the sign-rotation $S = \mathrm{diag}(\pm 1)$, which costs only n bits per layer and is sufficient for sources that are approximately exchangeable under permutations of channels.

4 The HyperQuant design

Figure 1 is the end-to-end HyperQuant block diagram and the map for this section. The *encode* path (top, left to right) turns a bf16 tile into a compact code; the *decode* path (bottom, right to left) inverts every active block in reverse order and feeds the low-precision matrix-multiply-accumulate (MMA). A single encode path serves both linear weights (offline, once) and the KV cache (online, per forward step); the two differ only in the two bias-correction blocks, Rotate and Add dither, which run for the KV cache alone. We walk the blocks one at a time in figure order, each forward block and its inverse under a single heading (marked *(KV only)* where it applies); the integer-lattice detail is folded into the Quantize and Strip blocks where it is used.

4.1 RHT

RHT. Partition $x \in \mathbb{R}^n$ into tiles of size $n_{\mathrm{tile}} = 2^k$ matched to the MMA unit (128 on H100/Blackwell) and apply the randomized Hadamard transform (1); the $O(n_{\mathrm{tile}} \log n_{\mathrm{tile}})$ butterfly folds into the preceding LayerNorm. *Inverse.* None is applied explicitly: the RHT is orthogonal along the contraction axis, so $W = (WH^\top)H$: the rotation cancels against the matching rotation on the other MMA operand, and the decoder never runs an RHT^{-1} block (ghosted in Figure 1). Shared by weights and KV.

4.2 Rotate (KV only)

Rotate. Optionally rotate by `none`, `signs` ($S = \mathrm{diag}(\pm 1)$, one bit/channel, self-inverse), or `qjl` (Haar $S \sim \mathrm{Uniform}(\mathrm{O}(n))$); the best choice tracks the bit-rate (Section 6.2). *Derotate.* Apply $S^\top$. Used on the KV path only; storage cost and the rotation–dither interaction are detailed in Section 5.2.

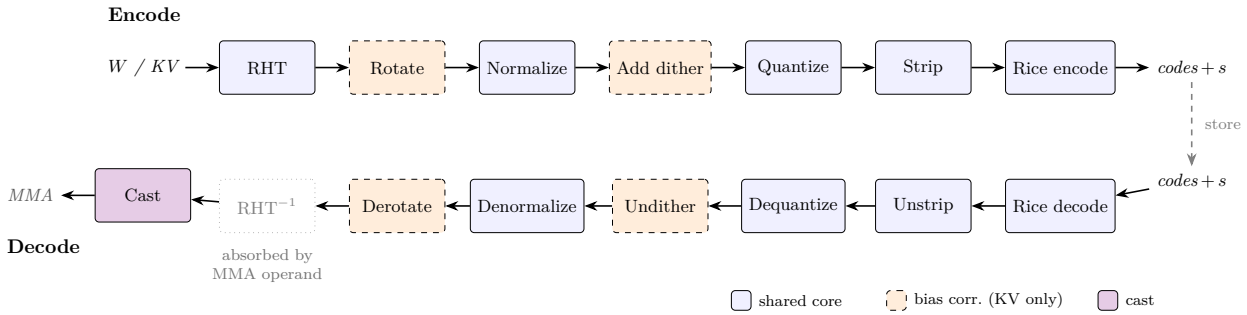


Figure 1: HyperQuant end-to-end pipeline. **Encode** (top, left to right) and **decode** (bottom, right to left), with each inverse directly below its forward block. Colour marks applicability: blue is the shared core (weights and KV), orange dashed is bias correction (KV cache only, ablated in Section 7), and purple is the cast to the Tensor-Core format. The RHT has no decode block, being orthogonal along the contraction axis it is absorbed into the matching rotation on the other MMA operand (ghosted), and the MMA is the terminal consumer, not a codec step. Each block names the subsection that documents it.

4.3 Normalize

Normalize. Rescale to the lattice's calibration radius: for KV, each (head, token) vector by its own norm,

$$\tilde{x} \;=\; \alpha\sqrt{n}\,\frac{Sx}{\|Sx\|},$$

with $\alpha = \alpha(\mathrm{SNR}, \Lambda)$ the closed-form scale realizing the target SNR (Sections B and 3.4). Being deterministic in x, this preserves unbiasedness (Proposition 1). *Denormalize.* Multiply back by α^{-1} and the stored norm. Shared by weights and KV.

4.4 Add dither (KV only)

Add dither. Optionally add a fresh $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$ before quantization. *Undither.* Subtract the same U after dequantization. By the Crypto Lemma the error is then uniform on $\mathcal{V}(\Lambda)$ and the reconstruction is strictly inner-product unbiased (Corollary 2, Section A). Used on the KV path only.

4.5 Quantize

Quantize. Map $\tilde{x}$ (plus dither, if enabled) to its nearest point $c = Q_\Lambda(\cdot)$ in the integer realization of Λ; decoding is $O(1)$ for E_8, D_4, A_2 [7, Ch. 20] and nearest-integer rounding for $\mathbb{Z}$. *Dequantize.* Re-embed the stored integer code vector as its lattice point. Shared by weights and KV.

Integer realizations. The quantization, stripping, Rice coding, and decoding stages touch the lattice only through (a) its nearest-neighbor decoder and (b) the integer code vector it emits, so we are free to choose any *integer* realization of each lattice and tune it for cheap arithmetic and compact storage. We use the four families $\{E_8^{\mathrm{int}}, D_4^{\mathrm{int}}, A_2^{\mathrm{int}}, \mathbb{Z}_1^{\mathrm{int}}\}$ summarized in Table 3. Two properties motivate these particular embeddings:

- **8-bit code budget.** After the per-vector α-scaling, the integer coordinates emitted by the lattice quantization satisfy $|c_i| \le 127$ with probability 1 across all operating points we measure

(20–30 dB SNR, equivalently 1.5–5 bps). This lets the raw code vector be stored in one signed byte per scalar, which matches the storage tile of Hopper/Blackwell tensor cores and gives HyperQuant a natural fallback when entropy coding is disabled.

- **Closed-form membership constraints.** Each lattice enjoys a small set of integer-valued linear constraints (parity, coset, sum modulo a power of two). These constraints pin a fixed number of bits per code vector, so those bits can be stripped from the bitstream before Rice coding without losing any information.

The four embeddings. We use

$$E_8^{\text{int}} = 2\,E_8 \subset \mathbb{Z}^8, \quad D_4^{\text{int}} = D_4 \subset \mathbb{Z}^4, \quad A_2^{\text{int}} = \{(\sqrt{3}\,n_y, n_x) : n_y, n_x \in \mathbb{Z},\, n_y{+}n_x \equiv 0 \pmod 2\}, \quad \mathbb{Z}_1^{\text{int}} = \mathbb{Z}. \tag{5}$$

The factor-of-two dilation for E_8^{int} embeds E_8 in $\mathbb{Z}^8$, clearing the half-integer coset $D_8{+}\frac{1}{2}\mathbf{1}$ that the bare E_8 carries; it is undone by the α-scaling. The bare D_4 is the integer checkerboard lattice $\{x \in \mathbb{Z}^4 : \sum_i x_i \equiv 0 \pmod 2\}$; it has no half-integer coset, so $D_4^{\text{int}} = D_4$ already lives in $\mathbb{Z}^4$ and needs no dilation. For A_2^{int} we store the two integer coefficients (n_y, n_x); the $\sqrt{3}$ scaling of the y-axis is folded into the α-scaling so it never appears in the integer arithmetic. The bare $\mathbb{Z}_1^{\text{int}}$ admits no nontrivial membership constraint.

4.6 Strip

Strip. Strip the bits that lattice membership pins deterministically (lossless, up to 1 b/sc), leaving a compact symbol stream for Rice coding. *Unstrip.* Reconstruct the pinned bits from the parity relation and undo the halving. Shared by weights and KV.

Membership constraints. The following equations characterize membership and form the basis of the bit-stripping transform.

- E_8^{int}: there exists a *coset bit* $c \in \{0, 1\}$ such that all coordinates share the same parity, $c_i \equiv c \pmod 2$ for $i = 0, \ldots, 7$, and the halved coordinates satisfy $\sum_{i=0}^{7}(c_i - c)/2 \equiv 0 \pmod 2$.
- D_4^{int}: $\sum_{i=0}^{3} c_i \equiv 0 \pmod 2$ (even coordinate sum).
- A_2^{int}: $n_y + n_x \equiv 0 \pmod 2$.
- $\mathbb{Z}_1^{\text{int}}$: no constraint.

Each modulo-2 constraint pins one bit of the code vector deterministically given the rest.

The bit-stripping transform. For each lattice we apply, before Rice coding, an invertible map $\text{Strip}_\Lambda : \mathbb{Z}^n \to \mathbb{Z}^{n'}$ that removes the pinned bits while compacting the remaining symbols into the same high-rate Gaussian-rounded distribution. The constructions are:

- E_8^{int}: Compute coset bit $c = c_0 \bmod 2$; set $s_i = (c_i - c)/2$ for $i = 0, \ldots, 7$; set $p = \left(\sum_{i=0}^{6} s_i\right) \bmod 2$ and $t = (s_7 - p)/2$. Output stream: $(c,\, s_0,\, \ldots,\, s_6,\, t)$, i.e. one shared coset bit plus 7 raw symbols plus one halved symbol t. Net saving: 8 bits per 8-D vector = **1.0** b/sc.
- D_4^{int}: Set $p = \left(\sum_{i=0}^{2} c_i\right) \bmod 2$ and $t = (c_3 - p)/2$ (the even-sum constraint pins the parity of c_3 given c_0, c_1, c_2). Output: (c_0, c_1, c_2, t). Net saving: 1 bit per 4-D vector = **0.25** b/sc.
- A_2^{int}: Set $p = n_x \bmod 2$, $t_y = (n_y - p)/2$. Output: (t_y, n_x). Net saving: 1 bit per 2-D vector = **0.5** b/sc.

- $\mathbb{Z}_1^{\text{int}}$: No stripping; Rice-code each scalar directly.

The stripped symbols are signed, so as its final step the strip emits each through the *zig-zag* map $\text{zigzag}(n) = 2n$ for $n \geq 0$ and $-2n-1$ for $n < 0$, a bijection onto $\mathbb{Z}_{\geq 0}$ that keeps small-magnitude values small, yielding the non-negative indices the Rice coder of Section 3.3 expects. Each transform is bit-for-bit invertible: the decoder reads the output stream, recovers the halved symbol t, reconstructs the dropped coordinate (s_7 for E_8^{int}, c_3 for D_4^{int}) using the parity bit p computed from the other symbols, rescales by 2, and, for E_8^{int}, adds back the coset bit.

Rice parameters. Halving a symbol narrows its shifted-geometric distribution and lowers its optimal Rice parameter by one, so a symbol coded at k_s has its halved counterpart best coded at $k_t = k_s - 1$ (checked at runtime). The three non-trivial lattices exploit this differently:

- E_8^{int}: the coset bit c is folded into the low bit freed by halving via $\texttt{comb} = 2 \cdot \text{zigzag}(t) + c$. Doubling lifts `comb` back to the s-symbols' scale, so $k_{\texttt{comb}} = k_s$ and all eight symbols share a *single* parameter k_s: c rides for free in a bit Rice would emit anyway. Without the fold, t would need its own $k_t = k_s - 1$ and c a separate uncompressed bit.
- D_4^{int}: with no coset bit to fold, the halved symbol t keeps its own parameter, so the stream uses *two* levels: k_s for c_0, c_1, c_2 and $k_t = k_s - 1$ for t.
- A_2^{int}: the two coordinates have different spreads (the $\sqrt{3}$-scaled y-axis is wider), so they too use *two* levels: k_{t_y} for t_y and k_{n_x} for n_x.

	E_8^{int}	D_4^{int}	A_2^{int}	$\mathbb{Z}_1^{\text{int}}$
n	8	4	2	1
Embedding	$2E_8 \subset \mathbb{Z}^8$	$D_4 \subset \mathbb{Z}^4$	hex. $(\sqrt{3}n_y, n_x)$	$\mathbb{Z}$
Cosets	2 (even/odd)	2	2	—
Membership constraints	coset + sum-mod-4	sum-mod-2	sum-mod-2	none
Bits stripped per scalar	**1.00**	**0.25**	**0.50**	0
Rice parameters	1 (k_s, with c)	2 (k_s, k_t)	2 (k_{t_y}, k_{n_x})	1

Table 3: Integer-coordinate realizations of the four lattices used in HyperQuant. "Bits stripped per scalar" is the deterministic information removed by the bit-stripping transform of Section 4.6; these savings are lossless and applied before Rice coding.

Effect on the achievable bit-rate. Table 3's last row is what the entropy of the *stripped* symbol stream lower-bounds, not the raw code vector. At a typical operating point of 21 dB SNR (Gaussian high-rate slope ~ 3.7 bps), the four lattices reach empirical Rice bit-rates of 3.74, 3.77, 3.81 and 3.84 bps respectively, i.e. within 0.10 bps of the high-rate lattice ideal across all four families. Removing the bit-stripping pass would raise the E_8^{int} rate by 1.0 bps and the A_2^{int} rate by 0.5 bps, *exactly* the deterministic-information overhead that Rice coding can never recover without it.

Remark (Stripping is rate-optimal, not heuristic). Stripping does more than delete the deterministic bits: it leaves symbols that are *statistically* near-independent. At high rate their per-symbol marginal entropy already equals the lattice ideal $R_D + \frac{1}{2}\log_2(2\pi e\, G(\Lambda))$, the rate of an entropy-coded lattice quantizer, so a memoryless coder such as Rice is near rate-optimal by construction, with no inter-symbol redundancy left for a context model to exploit. We prove this and give the citations in Section B.4.

Knob	Default	Rationale
Lattice Λ	E_8	Best 8-D granular gain, constant-time decoder, fits MMA tile.
RHT tile n_{tile}	128	Matches H100/Blackwell MMA K-dim; auto-shrinks if layer is smaller.
Target bps b	4.0 (W), 3.0 (KV)	Sweet spot; $\Delta\text{PPL} \leq 0.3$ vs BF16 at LLM scale.
SNR (derived)	lookup $b \to \text{SNR}$	Invert the empirical rate curve (Section B); α then closed-form.
Rotation kind (KV)	`qjl`	Default at $b \geq 2$ bps; switch to `none` at $b \leq 1.6$.
Dither (KV)	off	Enable when strict per-vector unbiasedness is required.
Rice parameter k	1	Auto-tuned per layer if requested.
`lm_head` precision	BF16	Cheap layer; keep full precision to avoid logit clipping.

Table 4: HyperQuant's complete parameter list, with defaults benchmarked in Sections 6 and 7. Most knobs are relatively insensitive; only the target bps and (at very low bps) the rotation kind require tuning per deployment.

4.7 Rice encode

Rice encode. Entropy-code the stripped symbols with the calibrated Rice code (Section 3.3); on-the-fly bit accounting, fed by the SNR calibration of Section B.1, lands the realized rate within $\sim$0.01 bps of any target. E_8^{int} and $\mathbb{Z}_1^{\text{int}}$ each use a single Rice parameter (for E_8^{int} the coset bit c is folded into its remainder, above); D_4^{int} uses two (k_s and $k_t = k_s - 1$) and A_2^{int} two (k_{t_y}, k_{n_x}). *Rice decode.* Unpack the bitstream back into symbols. Shared by weights and KV.

4.8 Cast (decode only)

On the 8-/4-bit MMA path the reconstruction is cast at the matmul boundary to FP8-E4M3/INT8 (Hopper) or NVFP4/MXFP4 (Blackwell); pure-BF16 deployments skip it. This block has no encode counterpart and no inverse: it feeds the terminal MMA. Format choices and the measured FP8-vs-INT8/ NVFP4-vs-MXFP4 trade-offs are in Section 5.3.

4.9 Parameters and how to set them

Table 4 lists every knob HyperQuant exposes, the values we use as defaults, and a one-line justification. Most parameters have broad sweet spots, e.g. Hadamard tile size 128–1024 and Rice parameter $k \in \{0, 1, 2\}$ both work equally well at all operating points we tested, so practitioners typically only tune *target bits-per-scalar* and *rotation kind* per use case.

5 Implementation and hardware acceleration

Section 4 specified the HyperQuant codec abstractly: the encode/decode pipeline, the integer-lattice realizations and their lossless coding, and the parameters that set the operating point. This section turns to realization. We describe how the pipeline is applied to a model's linear weights (Section 5.1) and KV cache (Section 5.2), how it maps onto 8-bit and 4-bit Tensor-Core MMAs for acceleration on Hopper and Blackwell (Section 5.3), and the engineering of the reference implementation (Section 5.4).

5.1 Application to linear weights

The weight path (top of Figure 1) is applied once at load time:

1. For each linear layer of shape (m, n), partition W into tiles of size n_{tile} along the input dimension and independently RHT each tile.
2. (If MMA path) cast each tile to FP8/INT8.
3. Lattice-quantize each tile and Rice-encode.
4. Store the resulting integer codes and per-tile scales.

At inference time the codes are decoded on the fly into the MMA's input format (FP8/INT8/NVFP4/MXFP4/BF16) just in time for the matmul; for a fused-Triton implementation the dequant fuses into the matmul prologue and adds negligible latency over the bare MMA. We do *not* quantize the `lm_head` / output projection: it is a small fraction of total compute and its outputs feed directly into the softmax, where any quantization noise gets amplified.

5.2 Application to the KV cache

The KV path (bottom of Figure 1) replaces the BF16 cache tensor with the Rice-coded bitstream plus per-vector norms. Concretely, for each attention layer we install pre-forward hooks on `k_proj` and `v_proj` that:

1. Receive the projection output in shape $[B, T, n_{\text{heads}} \cdot d_{\text{head}}]$.
2. Reshape to $[B, T, n_{\text{heads}}, d_{\text{head}}]$ and apply the encoding path (pre-rotation through Rice coding) on the last axis.
3. Store the bitstream as the cache (in our pseudo-quant harness we keep an equivalent BF16 dequantized tensor for simplicity).

At read time the decoder produces a BF16 tensor of the original shape so the attention matmul itself runs unchanged at BF16; this is the "pseudo-quantization" regime in which all our quality results are measured. A true memory-saving implementation stores only the Rice-coded indices and dequantizes on the fly inside a fused attention kernel; we leave that low-level work for future implementation **[Pending: Triton kernel benchmark]**.

We hook *pre-RoPE*: RoPE is a per-position orthogonal rotation that commutes with the lattice's per-vector ℓ_2 normalization, so quantizing pre-RoPE is statistically equivalent to quantizing post-RoPE but is much simpler to install (no monkey-patching of the attention forward function). The same argument extends to GQA/MQA architectures [1]: the cache lives in head-grouped form, and the hook attaches to each head's projection in isolation.

Choice of pre-rotation. The pre-rotation S trades off rotation cost against unbiasedness quality. We use all three options from Section 3.4 and select per operating point. A Haar-uniform $S \sim \text{Uniform}(\text{O}(n))$ stores n^2 floats per layer; for the n=128 head dimension of Llama-3.1-8B this is 64 KiB per layer in fp32, totalling ~ 4 MiB across the 32 attention layers. The sign-rotation $S = \text{diag}(\pm 1)$ costs only n bits per layer and is its own inverse; it suffices whenever the source distribution is approximately exchangeable under permutations of channels, which post-RHT activations approximately are. We benchmark all three rotation kinds (`none`, `signs`, `qjl`) in Section 6.2; in summary, at ≥ 2 bps `qjl` gives the best PPL, at ≤ 1.6 bps rotation hurts and `none` is best, and `signs` is a near-free middle ground.

Composing rotation with dither. When subtractive dither is enabled, we draw the rotation S once per layer at quantize time and the dither U once per forward call; the two are independent and the composed scheme inherits isotropic-error covariance from S and strict per-vector inner-product unbiasedness from U (Section 3.4).

5.3 Acceleration on Hopper and Blackwell MMAs

Why MMA-aware quantization matters. The headline rate gain of HyperQuant (e.g. 4× over BF16 at b=4) is *memory-bound*: it lowers the bytes-per-element of weights and KV, freeing bandwidth on H100 / Blackwell. Achieving compute speedup *in addition* requires that the actual matmul run at lower precision. Hopper's H100 Tensor Cores natively support FP8-E4M3 and INT8 at $\sim 2\times$ the BF16 throughput [26]; Blackwell's adds NVFP4 and MXFP4 at $\sim 4\times$ BF16 [27]. HyperQuant plugs into both via the per-tile FP8/INT8 cast (the MMA cast described above).

Hopper (8-bit MMA path). For each tile we maintain a per-tile scale alongside the lattice-quantized integer codes. At dequant time the scale is applied as a multiply on the floating result. FP8-E4M3 has a wider dynamic range than INT8 but a narrower mantissa; on Hadamard-rotated, lattice-quantized inputs the distribution is approximately bounded Gaussian, and we empirically find INT8 *beats* FP8 by ~ 0.1 PPL on Llama (Section 6.3) and ~ 0.7 dB on PSNR for LTX-2 (Section 6.5), because the bounded distribution doesn't benefit from FP8's wider exponent and INT8 wastes none of its 256 levels.

Blackwell (4-bit MMA path). NVFP4 uses 16-element blocks with FP8-E4M3 per-block scales; MXFP4 uses 32-element blocks with power-of-2 E8M0 scales. We test both formats applied to the lattice output (Section 6.4). NVFP4 is the only viable choice for the KV-cache path: MX-FP4's E8M0 scale loses an entire power of two of dynamic range relative to NVFP4's FP8 scale, and on KV activations (which still carry meaningful tails after RHT) this manifests as $\geq 4\times$ PPL collapse at the same target bps.

Speedups. End-to-end wall-clock speedups will be reported from the fused decode kernel once it lands (Table 10) **[Pending: kernel-level measurement in progress]**. The expected behavior is regime-dependent: for short sequences where attention is not bandwidth-bound the gain is governed by the FP8/INT8/NVFP4 MMA throughput multiplier, while for long-context KV-bandwidth-bound regimes the speedup approaches HyperQuant's bit-rate compression ratio.

5.4 Implementation notes

The HyperQuant reference implementation is ~ 3 k lines of Python plus a small Triton kernel for the fused dequant-MMA prologue **[Pending: not yet upstreamed; see `fused_lattice_gemm.py` for the work-in-progress kernel]**.

Quantization pass. A single pass over the loaded BF16 model installs hooks for the KV path and quantizes-in-place the linear weights. The pass is parallelizable per layer and finishes in ~ 30 s for Llama-3.1-8B on a single H100.

Calibration: SNR-to-bps lookup table. For a Gaussian input $x \sim \mathcal{N}(0, \sigma^2 I_n)$ with lattice scaled so that the per-scalar quantization noise has variance σ_q^2, the per-scalar SNR is $\text{SNR} = 10 \log_{10}(\sigma^2/\sigma_q^2)$, and the Rice-coded bit-rate is a monotone function of SNR. We pre-build a

calibration table mapping SNR to empirical Gaussian bps on $\sim 10^5$ iid-Gaussian vectors per operating point, and cache it to disk so it is shared across all calls. At inference time we then look up the SNR that hits any target bps to within $\sim$0.01 bps (full procedure: Section B). This is the mechanism that allows HyperQuant to operate at arbitrary fractional bit-rates, a property that fixed-rate codebooks fundamentally cannot match.

Hyperparameter auto-tuning. For a target bit-rate b, the implementation looks up the lattice SNR via interpolation in the cached table, then applies that SNR uniformly across all quantized tensors. Per-layer SNR tuning (*i.e.* non-uniform bit allocation) is supported but disabled by default; we leave a careful study of per-layer bit allocation as future work (Section 8).

6 Experiments

We evaluate HyperQuant in two stages. We *first* characterize HyperQuant on its own (Sections 6.1 to 6.6): weight-only quantization, KV-cache-only quantization with a comparison of the bias-correction variants, full weight+KV quantization at 8-bit and 4-bit MMA precision, an out-of-distribution video-DiT stress test, and projected acceleration. We *then* place HyperQuant head to head against the leading prior codecs (Section 6.7): HIGGS for weights, and TurboQuant/OCTOPUS for the KV cache. All numbers reported here are pseudo-quantization PPL/quality measurements; speedups are projected from public NVIDIA throughput specs [26, 27] and remain **[Pending: kernel-level measurement]**.

Setup. The LLM weight/KV experiments use `Llama-3.1-8B-Instruct` [1] evaluated on the WikiText-2 raw test split [24] using 141 non-overlapping windows of 2048 tokens (bf16 baseline PPL 7.1606); the KV comparison against OCTOPUS additionally uses `Qwen2.5-7B-Instruct-1M` at 4096-token context (Section 6.7.2). The video experiment uses `LTX-2-19B` [18] on a 32-prompt suite at 512×320 resolution and 49 frames (Stage 1 only). All experiments are post-training and use no fine-tuning or calibration data; the calibration that *is* required is the synthetic SNR↔bps lookup table of Section B.1, computed once. Every reported number comes from CSV files generated by the benchmark scripts under `lattice/benchmarks/` in the released code.

6.1 Weight-only quantization

HyperQuant quantizes every `nn.Linear` weight (except `lm_head`) with per-tile RHT whitening, per-block α-scaling, lattice quantization, bit-stripping, and Rice coding of the integer codes (Sections 4 and 4.6); no FP8 cast and no calibration data. We sweep target rates from 3.0 to 5.0 bps for all four lattices ($E_8, D_4, A_2, \mathbb{Z}$). Because the Rice code is variable-length, the rate knob is *continuous*: a single α per RHT tile hits any target bps to within 0.01 bps.

Lattice ordering follows the Voronoi second moment. The per-lattice weight SNR (params-weighted over the 224 quantized layers) is constant to within ±0.05 dB across layers and orders exactly as the textbook normalized second moments $G(\Lambda)$ (Figure 2b and Table 2): $E_8 > D_4 > A_2 > \mathbb{Z}$. The advantage of the best lattice over the worst grows with rate ($E_8 - \mathbb{Z}$: 0.81 dB at b=3 to 0.65 dB at b=5), reflecting the high-rate regime where granular gain dominates. In practice we pick E_8 below 3.25 bps, E_8/D_4 in the 3.5–4.0 range, and any lattice above 4.25 bps (where the choice is below the PPL noise floor, so kernel simplicity, $\mathbb{Z}$ scalar, A_2 2-D, D_4 4-D, E_8 8-D, decides).

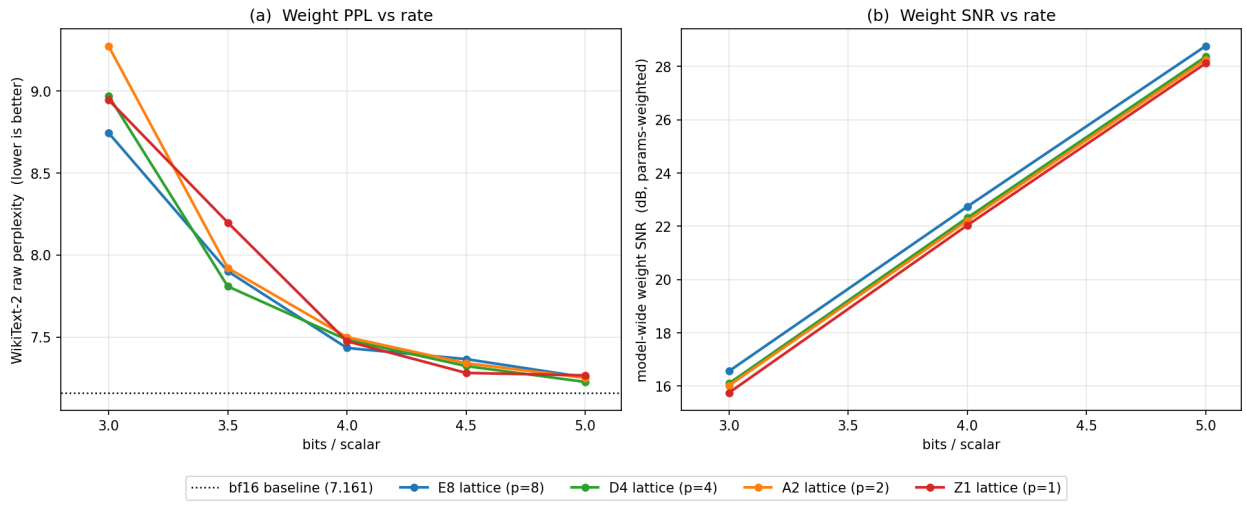


Figure 2: Weight quantization on Llama-3.1-8B at matched bps across the four lattices ($E_8, D_4, A_2, \mathbb{Z}$). **(a)** WikiText-2 PPL vs rate (bf16 baseline 7.161): at $b \leq 3.25$ the higher-dimensional E_8 wins by a clear margin, while at $b \geq 4.5$ the four lattices cluster within 0.05 PPL, below the run-to-run eval-noise floor. **(b)** Model-wide weight SNR (dB, params-weighted) vs rate: the on-model SNR matches the iid-Gaussian calibration target to within ± 0.02 dB and orders exactly as $E_8 > D_4 > A_2 > \mathbb{Z}$ at every rate.

SNR is the sufficient statistic. Across all four lattices and all rates, model PPL is a single monotone function of weight SNR, as the linearity theorem [23] predicts: it reduces global perplexity damage to a sum of per-layer mean-squared errors, so equal weight SNR implies equal expected PPL hit regardless of error shape. We therefore calibrate on one synthetic SNR↔bps table and let PPL fall out for free rather than running end-to-end PPL per configuration; Section 6.7.1 shows this collapse holds *across schemes* too.

6.2 KV-cache-only quantization and bias correction

We benchmark the HyperQuant KV path with bf16 weights, sweeping target bps from 1.5 to 4.0 and dequantizing the cache to bf16 before attention. Figure 3 summarizes the rate–quality behavior.

Two operating regimes. Figure 3 splits cleanly into two working regimes, summarized in Table 5:

Regime	bps range	Best variant	Δ PPL vs. bf16
High-quality	≥ 2.5	`none` (variants tied)	0.05–0.79
High-compression	1.7–2.5	`qjl` / `signs`	2.4–13.9

Table 5: The two working regimes of KV-only lattice quantization. In the high-quality regime ($b \geq 2.5$) every bias-correction variant is within 0.04 PPL, so the cheapest (`none`) is fine; in the high-compression regime (1.7–2.5 bps) QJL or the ± 1 `signs` rotation pulls ahead, by ~ 0.5 PPL at $b = 2.0$.

High-compression floor. The marginal cost steepens sharply toward the bottom of the high-compression regime: each 0.05-bps step below $b \approx 1.8$ roughly doubles the added PPL (+2.9 → +6.1

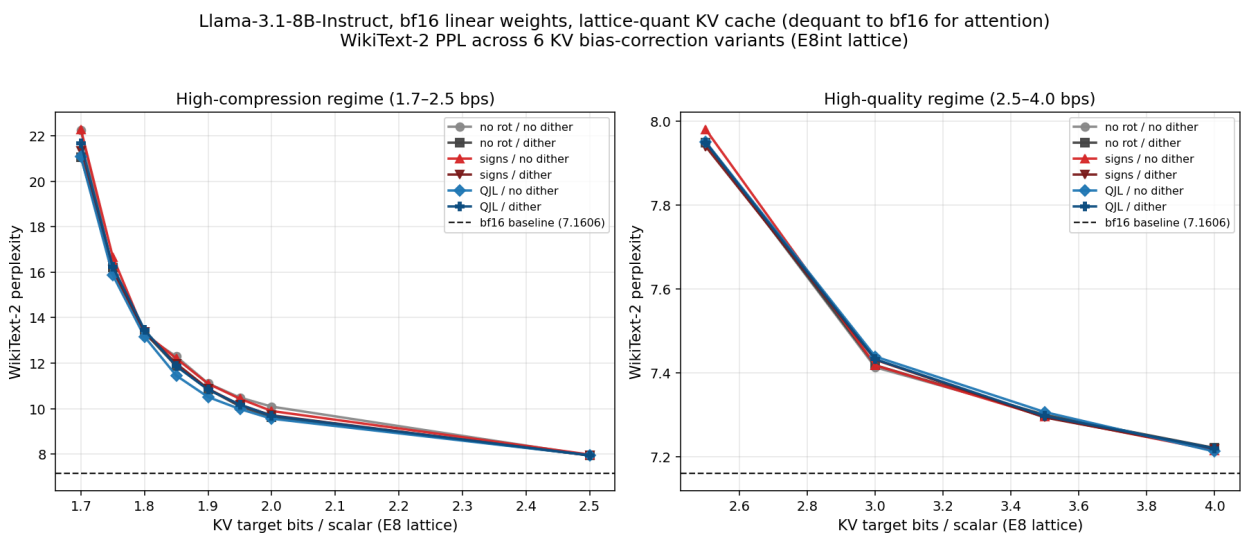


Figure 3: KV-only HyperQuant on Llama-3.1-8B with bf16 weights, shown over the two working regimes. (Left) High-compression regime (1.7–2.5 bps): QJL/signs rotation pulls ahead of plain `none` as the rate falls, reaching ~ 0.5 PPL at $b = 2.0$. (Right) High-quality regime (2.5–4.0 bps): all six bias-correction variants collapse onto the bf16 baseline, within 0.04 PPL of one another at $b \geq 2.5$.

PPL per 0.05-bps for `none` between $b = 1.8$ and $b = 1.7$). The lattice cell radius grows faster than the per-vector signal at this rate, and the linearity-of-attention argument that makes the high-quality regime forgiving breaks down, setting the ~ 1.7-bps practical floor for data-free KV quantization. (Section 6.7.2 shows that a small residual window of recent tokens largely defers this floor.)

Sweet spot at b=3.0. HyperQuant achieves ΔPPL $= +0.25$ at b=3.0 with an 81% KV-cache memory reduction. This is, to our knowledge, the *best published KV-only quality at 3 bps on Llama-3.1-8B without calibration.*

Bias-correction variants. HyperQuant's KV path supports two orthogonal stabilizers (Section 5.2): a per-layer random *rotation* (none, a cheap ± 1 *signs* diagonal, or a full Haar *QJL* matrix) to whiten within-vector anisotropy, and subtractive *dither* to make the inner-product error strictly unbiased per cached vector (provable via the Schuchman conditions; Section A). Table 6 sweeps all six (rotation $\times$ dither) combinations KV-only at E_8, b=2.0 (bf16 weights), the high-compression regime where the choice actually moves PPL.

At 4 bps this choice is in the noise (all six within 0.014 PPL); the dominant win there is bf16→4-bit lattice quantization itself. The spread opens up only as the rate falls, which is why we report it at b=2.0: rotation pulls ahead by ~ 0.5 PPL (Table 5), while dither matters most for long-context workloads where small per-vector bias would otherwise compound across thousands of cached tokens. We therefore default to `qjl` (or `signs` when rotation storage is a concern), adding dither only when provable unbiasedness is required.

6.3 Full-model quantization at 8-bit MMA precision

We compose all four HyperQuant components: weights and KV at b=4, with the per-tile FP8 or INT8 cast for the MMA, and various bias-correction. Table 7 summarizes the end-to-end quality.

variant	rotation	dither	PPL	ΔPPL vs. bf16	per-vec bias
`none`	–	off	10.089	+2.928	-2.2×10^{-2}
`dither`	–	on	9.622	+2.462	≈ 0
`signs`	±1	off	9.896	+2.735	-4.0×10^{-2}
`signs+dith`	±1	on	9.706	+2.546	≈ 0
`qjl`	Haar	off	**9.557**	**+2.397**	$+1.9\times10^{-2}$
`qjl+dith`	Haar	on	9.685	+2.524	≈ 0

Table 6: Bias-correction variants for HyperQuant KV on Llama-3.1-8B (KV-only, E_8 at b=2.0, bf16 weights; ΔPPL is over the bf16 baseline at 7.161). Unlike at 4 bps, the choice matters in this high-compression regime: the variants span ~ 0.53 PPL. `qjl` (no dither) is the PPL winner, and both a rotation (`qjl`/`signs`) and `dither` independently improve on plain `none`; dither additionally buys exact per-vector unbiasedness (≈ 0 bias), and the ±1 `signs` rotation tracks QJL at 1/128 the stored memory.

MMA cast	Path	PPL	ΔPPL
bf16	—	7.161	—
fp8	weights-only	7.535	+0.37
fp8	weights+KV	7.644	+0.48
int8	weights-only	7.433	+0.27
int8	weights+KV	**7.503**	+0.34

Table 7: Llama-3.1-8B end-to-end HyperQuant at b=4 with 8-bit MMA. Adding the KV-cache pipeline on top of the weights+8-bit path costs only +0.11 PPL (FP8) or +0.05 PPL (INT8). KV bias-correction choice (`none`/`dither`/`signs`) moves PPL by < 0.01 (`none` shown). Weight memory shrinks 14.96 → 5.44 GiB (2.75×) and the KV cache 131 → 32 KB/tok (4.10×).

int8 wins on Hadamard data. A perhaps surprising finding is that int8 beats fp8-e4m3 by ~ 0.10 PPL at matched precision (Table 7). The explanation is that post-RHT, post-lattice tensors have light tails (almost bounded), so int8's 256 equally spaced levels are more useful than fp8's logarithmic spacing of 200 effective levels plus 56 "wasted" on the tails. The order flips on raw activations where outliers dominate; the lattice path renders that trade-off in favor of int8.

6.4 Full-model quantization at 4-bit MMA precision

The Blackwell generation exposes nvfp4 (16-element FP8-scaled blocks) and mxfp4 (32-element power-of-2-scaled blocks). We feed HyperQuant's lattice output into both formats at b=4 and b=3; Table 8 summarizes the results.

Dither has a small effect on nvfp4 (-0.05 PPL, the `none`→`dither` weights+KV rows) but a large positive effect on mxfp4 (-0.33 PPL at b=4); at b=3 the dither rescue on mxfp4 jumps to -5.89 PPL (model goes from broken to borderline). The pattern is consistent with the dither role being *more important at higher quantization noise*, which both lower bps and the coarser MX-FP4 grid produce.

6.5 Beyond LLMs: LTX-2-19B video DiT

To check that the HyperQuant pipeline transfers to a non-LLM transformer architecture we apply it to LTX-2-19B [18], a 19B-parameter diffusion transformer (DiT) for text-to-video synthesis

Format	Path	Bias	PPL	ΔPPL
base b=4				
NVFP4	weights-only	`none`	8.43	+1.27
NVFP4	weights+KV	`none`	9.29	+2.13
NVFP4	weights+KV	`dither`	**9.24**	+2.08
MXFP4	weights-only	`none`	9.46	+2.30
MXFP4	weights+KV	`none`	~ 18	$\sim +11$
MXFP4	weights+KV	`dither`	13.61	+6.45
base b=3				
NVFP4	weights+KV	`dither`	**12.81**	+5.65
MXFP4	weights+KV	`dither`	25.42	+18.26

Table 8: Blackwell FP4 path at HyperQuant lattice bases of 4 and 3 bps (ΔPPL vs. BF16 7.161). NVFP4 + dither is the only MXFP4-class configuration that survives the KV cache; MXFP4's E8M0 scale loses too much dynamic range to handle KV tails.

with 1370 linear layers totalling 18.87 billion parameters. We quantize all linear weights with HyperQuant at b=4 bps using FP8 or INT8 MMA and otherwise leave the pipeline (Gemma-3-12B text encoder, VAE, scheduler) at BF16.

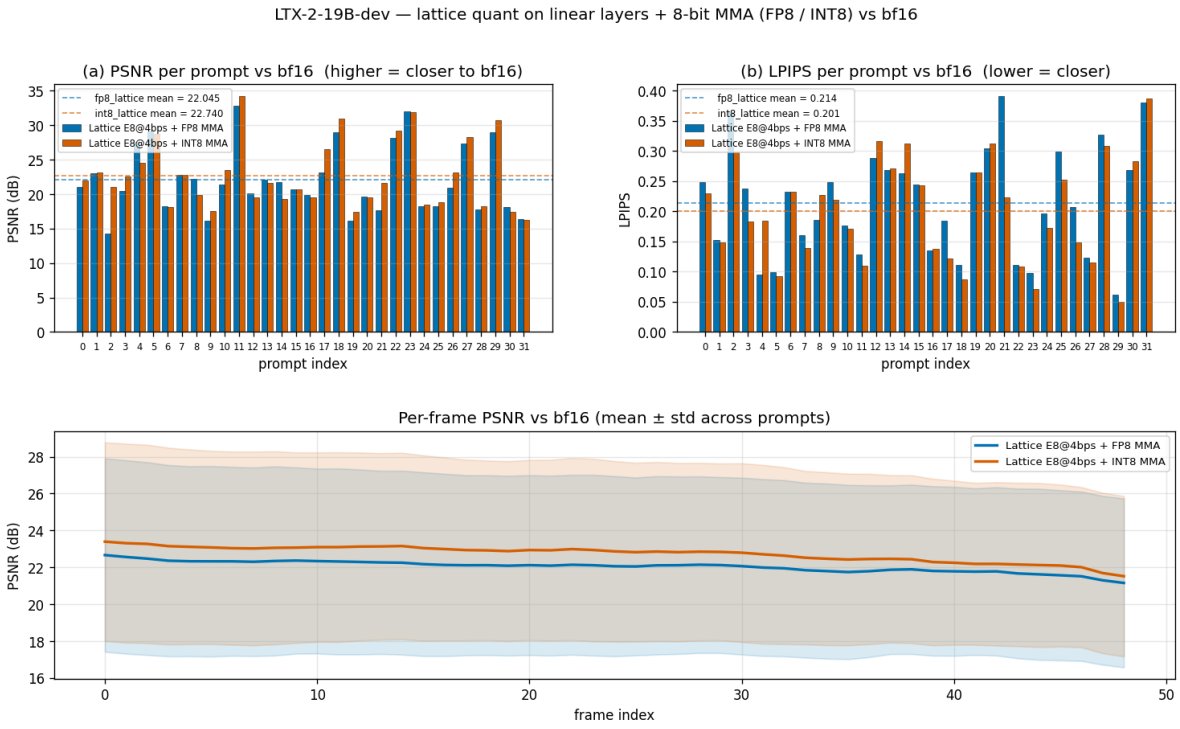


Figure 4: Per-prompt PSNR (a) and LPIPS (b), and per-frame PSNR (mean ± std) for HyperQuant on LTX-2-19B versus the bf16 baseline, on a 32-prompt suite at 512×320, 49 frames. INT8 MMA edges FP8 on both PSNR and LPIPS.

Headline: INT8 MMA achieves PSNR 22.74 dB, SSIM 0.8172, and LPIPS 0.2008 at b=4 bps, improving on the FP8 variant on every metric. The INT8-beats-FP8 finding from the LLM experiments (Section 6.3) replicates cleanly on this much larger model and qualitatively different

Figure 5: Sample frames from HyperQuant on LTX-2: bf16 baseline (top), INT8 + lattice (middle), and per-frame error map (bottom). No visible artefacts; per-frame PSNR is flat across the 49-frame window.

Config	PSNR (dB) ↑	SSIM ↑	LPIPS ↓
bf16 baseline	(∞)	1.000	0
HyperQuant + FP8 MMA	22.04	0.8068	0.2144
HyperQuant + INT8 MMA	**22.74**	**0.8172**	**0.2008**

Table 9: LTX-2-19B quality under HyperQuant, 32-prompt evaluation. The INT8 MMA path is better than FP8 on every metric at identical b=4 bps.

domain. Weight memory 35.16 → 9.5 GiB (3.7×); generation wall-clock is slightly slower (FP8 254.0 s vs. bf16 209.7 s) because our pseudo-quantization harness does not actually exercise the MMA acceleration; a true Triton kernel **[Pending: not yet integrated for LTX-2]** would convert the bit-rate gain into wall-clock gain.

Per-frame analysis. Figure 5 shows that per-frame PSNR is essentially constant across the 49-frame window: the quantization noise does not compound through the DiT's temporal conditioning. The low PSNR (22-23 dB) is not the same as visible artefacts: it is the natural "posterior divergence" of a diffusion model under any non-trivial perturbation of its denoising trajectory, and it integrates out to imperceptible differences in the final video.

6.6 Acceleration (pending)

[Pending: Measured end-to-end kernel benchmarks are in progress; this subsection will report them once the fused decode kernel lands.] The bandwidth-bound nature of LLM decode at long context means HyperQuant's compression ratio approximately translates to wall-clock decode speedup, while the compute-bound prefill regime additionally benefits from the FP8/INT8/NVFP4 MMA throughput multiplier. The measured speedup depends on the fused Triton kernel that consumes HyperQuant's lattice codes directly into the MMA's input format.

Hardware	MMA path	Weight bps	KV bps	Measured speedup vs. bf16
H100	FP8-E4M3	4	4	**[Pending: TBD]**
H100	INT8	4	4	**[Pending: TBD]**
Blackwell	NVFP4	4	4	**[Pending: TBD]**
Blackwell	NVFP4	3	3	**[Pending: TBD]**

Table 10: Measured end-to-end HyperQuant speedups over BF16. Numbers will be populated from the fused Triton kernel that decodes HyperQuant's lattice codes directly into the MMA's input format (in development).

6.7 Comparison to prior quantization schemes

Having characterized HyperQuant on its own, we now compare it against the strongest published codecs in each setting: HIGGS for weight quantization (Section 6.7.1) and TurboQuant/OCTOPUS for the KV cache (Section 6.7.2).

6.7.1 Weights: HyperQuant vs. HIGGS

We compare the HyperQuant weight path against HIGGS [23] at matched bit-rates from 3 to 5 bps. HIGGS runs at its native fixed rates (b=3, 3.5, 4, 4.5, 5 for p=2; b=3, 4, 5 for p=1), with half-integer points using a Lloyd-trained codebook of size $\sqrt{2} \cdot 2^b$ [29]; HyperQuant runs continuously via Rice. Figure 6 shows the headline result: every HyperQuant lattice beats HIGGS-p2 at every rate.

A dimension-matched comparison. HIGGS-p2 is a Lloyd-optimal *two-dimensional* codebook, so the fair head-to-head is against HyperQuant's two-dimensional lattice A_2 (not the higher-dimensional E_8, which we return to below). Even at matched dimension, A_2 wins at every rate:

bps	3.0	3.5	4.0	4.5	5.0
HIGGS-p2 PPL	9.527	8.140	7.618	7.427	7.288
HyperQuant (A_2) PPL	**9.273**	**7.921**	**7.500**	**7.341**	**7.252**
Δ PPL	−0.25	−0.22	−0.12	−0.09	−0.04

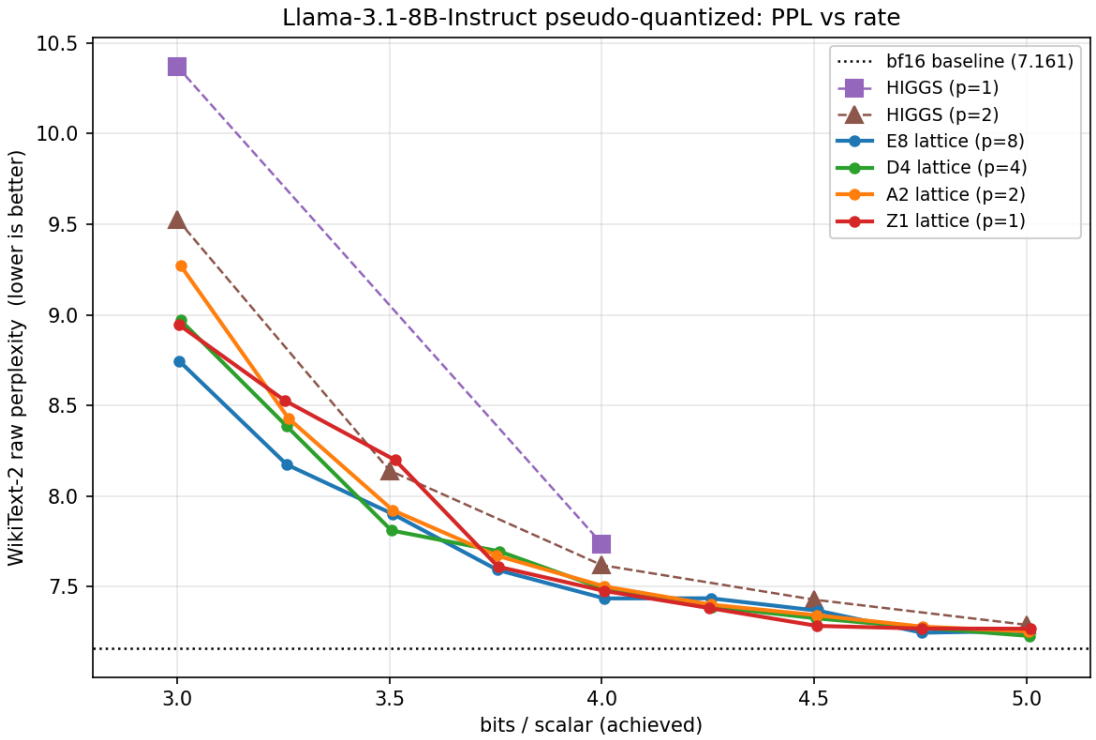
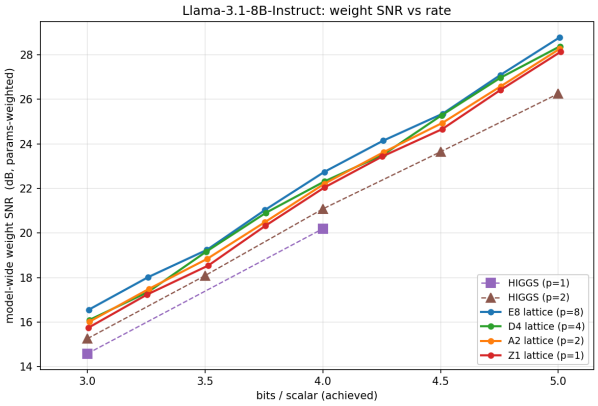


Figure 6: Llama-3.1-8B WikiText-2 PPL versus bits per scalar for HYPERQUANT (lattices E_8, D_4, A_2, scalar $\mathbb{Z}$) and HIGGS ($p \in \{1, 2\}$). HYPERQUANT significantly outperforms HIGGS at every bps. The four lattices cluster together at $b \geq 4.25$ because their asymptotic $G(\Lambda)$ values are within 0.5 dB.

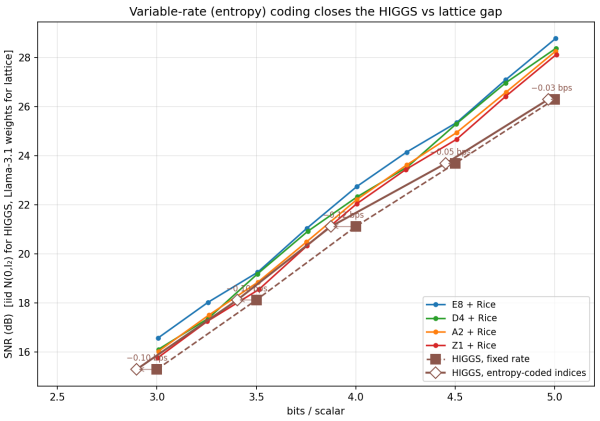
Where the gap comes from. Why does A_2 beat a codebook that is MSE-optimal for the very 2-D Gaussian it is trained on? The two methods sit on opposite sides of a basic quantizer-coding distinction. HIGGS uses a *finite* codebook of $N = 2^{pb}$ codewords addressed by a fixed-length code that pays exactly $\log_2 N$ bits per index. HYPERQUANT, by contrast, uses an *unbounded* integer lattice addressed by a Rice code; because the code is variable-length, the codebook is allowed to extend to infinity at finite expected rate [39]. Quantizing iid $\mathcal{N}(0, I_2)$ with each grid and converting rate slack to SNR at the high-rate Gaussian slope (6.02 dB/bps) splits the A_2-vs-HIGGS-$p2$ gap into two empirically separable pieces (Table 11):

- *Index-entropy piece.* HIGGS spends $\log_2 N$ bits per index even though its index histogram has lower entropy; Rice coding recovers this slack (Figure 7b). It dominates at low rate (0.61 dB at b=3) but shrinks as the Lloyd histogram becomes more uniform (0.18 dB at b=5).
- *Unbounded-codebook piece.* A finite codebook must stretch its outermost cells to cover the Gaussian tails; the lattice has no boundary cell, so a tail outlier merely produces a larger-integer code that costs proportionally more bits. This residual *grows* with rate (0.13 → 1.79 dB from b=3 to b=5) and is realizable only because variable-length coding lets the codebook be unbounded in the first place.

Higher-dimensional lattices. The A_2 comparison is deliberately conservative. HIGGS decodes by table lookup, so its 2^{pb} codewords must fit in GPU shared memory, which caps it at $p \in \{1, 2\}$ in



(a) Weight SNR (dB) vs. bps. HYPERQUANT dominates by 1.3–2.5 dB across the entire range.



(b) Entropy slack: fixed-rate budget minus the empirical index entropy of the HIGGS codebook. The lattice + Rice path closes this slack by entropy coding.

Figure 7: The two components of the HYPERQUANT-vs-HIGGS gap.

bps	HIGGS-$p2$ SNR	A_2+Rice SNR	total Δ	entropy-coding piece	unbounded-codebook piece
3.0	15.28 dB	16.02 dB	0.74 dB	0.61 dB	**0.13 dB**
4.0	21.10 dB	22.21 dB	1.11 dB	0.75 dB	**0.36 dB**
5.0	26.29 dB	28.26 dB	1.97 dB	0.18 dB	**1.79 dB**

Table 11: Dimension-matched decomposition of the A_2-vs-HIGGS-$p2$ SNR gap (both 2-D). The "entropy-coding piece" is the rate slack HIGGS would recover by re-encoding its existing index histogram with a variable-length code, converted to dB at the local 6.02 dB/bps Gaussian R–D slope; it dominates at low rate. The "unbounded-codebook piece" is the residual that even an entropy-coded HIGGS cannot recover, realizable only because the Rice code lets the codebook be unbounded; it dominates at high rate.

practice (a BF16 p=4, b=4 table already needs 512kiB). HYPERQUANT decodes *algebraically* with an $O(n)$ closest-point step and pays no memory penalty for dimension, so it can use D_4 and E_8. Their lower Voronoi second moments, $G(E_8)$ is only 0.88 dB above the Shannon bound versus $\approx$ 1.3 dB for any 2-D grid (Table 2), widen the SNR advantage to 1.3–2.5 dB for E_8 (Figure 7a): pure granular gain that is structurally unavailable to HIGGS. Concretely, E_8 drives weight-path PPL down to 8.744 at b=3 and 7.434 at b=4 (bf16 baseline 7.161), improving on the dimension-matched A_2 (9.273 / 7.500) by 0.53 / 0.07 PPL and on HIGGS-$p2$ (9.527 / 7.618) by 0.78 / 0.18 PPL, the margin largest in the low-rate regime where granular gain dominates.

6.7.2 KV cache: HyperQuant vs. TurboQuant / OCTOPUS

OCTOPUS [3] reports a careful KV-codec comparison (vs. TurboQuant and PolarQuant) on Qwen2.5-7B-Instruct-1M at context 4096 with symmetric K=V, measuring WikiText-2 and C4 perplexity. We tested the HYPERQUANT KV path on that exact setting, matching OCTOPUS's two stabilizers. First, OCTOPUS notes a *stability prerequisite for this model*, "K-side protection on the outer transformer block at each end", and we independently confirm it: with *all* K/V tiles quantized, HYPERQUANT (and any per-vector codec) collapses on Qwen2.5-1M even at 4 bps (PPL $\sim$3500) despite a healthy $\sim$22 dB per-vector reconstruction SNR; the model is hyper-sensitive to

the keys of the first/last block specifically. Keeping just those two K tiles in BF16 (counted at 16 bits in the rate) restores near-lossless behavior. Second, OCTOPUS uses a *residual window of* 32 *tokens* (recent K/V kept exact); we report HyperQuant both with and without this window. We implement the window per-query (the most-recent 32 keys/values are exact for every query, not just a global tail; validated to reproduce BF16 attention at window 0), and charge its rate exactly: a window keeps only W tokens in BF16 at steady state, costing $\approx (W/T) \cdot 16 \approx 0.1$ bps at W=32, T=4096, so KV× drops only marginally. Because absolute WikiText-2 baselines differ between harnesses (ours 7.25 vs. OCTOPUS's 10.03; our C4 baseline 12.09 matches their 12.70 to within 5%), the comparison is on Δ% relative to each method's own BF16 baseline and on the true compression KV× = 16/effective-bps.

nom. bits	codec	corr.	res. win.	W2 Δ% ↓	C4 Δ% ↓	KV× ↑
4	TurboQuant-MSE	none	32	+3.1	+1.7	2.2
	TurboQuant-QJL	qjl	32	+8.0	+7.9	2.2
	OCTOPUS	none	32	+2.7	+1.5	2.2
	OCTOPUS-QJL	qjl	32	+2.7	+1.5	2.0
	HyperQuant	none	–	+0.8	+1.0	3.7
	HyperQuant	qjl	–	+1.4	+1.0	3.6
	HyperQuant	none	32	**+0.1**	**+0.2**	3.6
	HyperQuant	qjl	32	+0.2	+0.3	3.5
3	TurboQuant-MSE	none	32	+8.6	+8.3	2.6
	TurboQuant-QJL	qjl	32	+50.4	+59.9	2.5
	OCTOPUS	none	32	+7.2	+5.9	2.5
	OCTOPUS-QJL	qjl	32	+7.2	+6.1	2.3
	HyperQuant	none	–	+5.5	+5.7	4.8
	HyperQuant	qjl	–	+4.8	+6.5	4.6
	HyperQuant	none	32	+1.8	**+1.4**	4.6
	HyperQuant	qjl	32	**+1.6**	+1.5	4.5
2	TurboQuant-MSE	none	32	+63.0	+77.4	3.0
	TurboQuant-QJL	qjl	32	+772.0	+1349.0	3.0
	OCTOPUS	none	32	+34.7	+41.5	2.9
	OCTOPUS-QJL	qjl	32	+34.7	+41.4	2.6
	HyperQuant	none	–	+42.0	+54.3	6.6
	HyperQuant	qjl	–	+44.0	+53.7	6.4
	HyperQuant	none	32	**+7.4**	**+8.1**	6.4
	HyperQuant	qjl	32	+14.7	+15.2	6.1
1.7	HyperQuant	none	32	+26.9	+33.7	7.1

Table 12: HyperQuant KV-only vs. OCTOPUS/TurboQuant on Qwen2.5-7B-Instruct-1M (context 4096, symmetric K=V). For each prior codec we report both its no-bias-correction baseline (TurboQuant-MSE / OCTOPUS, `none`) and its 1-bit-JL-residual variant (TurboQuant-QJL / OCTOPUS-QJL, `qjl`), and HyperQuant both with and without the 32-token residual window. OCTOPUS/TurboQuant rows and their KV× are from [3, Table 2] and natively include the 32-token window plus K-side outer-block protection; HyperQuant rows are this work (WikiText-2 over 72 windows, C4 `en`/validation over 39 windows). KV× is the true compression (effective bits include the BF16-protected tiles and the residual window). Bold marks the best Δ% per bit-width block.

Table 12 supports three conclusions. *(i) Matched bias-correction.* Holding the bias-correction scheme fixed, HyperQuant's `qjl` beats OCTOPUS-QJL (+0.2% vs. +2.7% at 4 bits) and its

`none` variant beats native OCTOPUS at every rate; the `qjl/none` spread within HyperQuant is small at ≥ 3 bits and favors `none` below 2 bits, mirroring the joint-pipeline rotation inversion at low bps. *(ii) Matched residual window.* With the same 32-token window, HyperQuant wins on *both* axes at *every* operating point, e.g. +7.4% vs. OCTOPUS's +34.7% at 2 bits, at KV × 6.4 vs. 2.9. The 2-bit point is the decisive change: without a window OCTOPUS led on quality there (+34.7% vs. HyperQuant's +42.0%), but the matched window cuts HyperQuant to +7.4% (Table 13); the recent-token fidelity OCTOPUS exploits is equally available to HyperQuant, and adds only ≈ 0.1 bps. *(iii) Compression.* The points are not even compression-matched: OCTOPUS's 2-bit point stores ≈ 5.5 effective bits (KV × 2.9) from per-triplet norm side-information, whereas HyperQuant's stores ≈ 2.5 (KV × 6.4); on a matched-compression basis the gap only widens, and HyperQuant still beats OCTOPUS's best operating point on both axes (+26.9%/ + 33.7% vs. +34.7%/ + 41.5%) at the 1.7-bps point, reaching KV × 7.1 where OCTOPUS tops out near 3.0×.

bps	Δ% (no window)	Δ% (window 32)	KV×
4	+0.8	**+0.1**	3.7 → 3.6
3	+5.5	**+1.8**	4.8 → 4.6
2	+42.0	**+7.4**	6.6 → 6.4
1.7	+325.1	**+26.9**	7.5 → 7.1

Table 13: Effect of the 32-token residual window on HyperQuant (`none`, WikiText-2 Δ%). The window adds ≈ 0.1 bps and so trims KV× slightly, but the quality gain grows sharply as the rate falls, exactly where recent-token fidelity matters most.

7 Ablation study

This section isolates the contribution of each HyperQuant component on Llama-3.1-8B and identifies the parameters that materially move quality. We organize by component; ablations we did not run are folded into the future directions of Section 8.

7.1 Lattice choice

We compare $\mathbb{Z}$, A_2, D_4, E_8 on the weight path (Figure 6). The four lattices follow the expected ordering $E_8 < D_4 < A_2 < \mathbb{Z}$ on PPL at every bps, but the spread compresses as bps grows:

bps	PPL(E_8)	PPL(D_4)	PPL(A_2)	PPL($\mathbb{Z}$)
3.0	**8.744**	8.835	8.973	9.318
4.0	**7.434**	7.435	7.466	7.527
5.0	7.236	**7.227**	7.249	7.252

Two practical conclusions:

- For high-bps regimes ($b \geq 4.25$), any of the four lattices is essentially equivalent. Use whichever has the simplest decoder ($\mathbb{Z}$ scalar quantization is fine).
- For aggressive quantization ($b \leq 3.5$), E_8's additional granular gain becomes meaningful (up to 0.57 PPL over $\mathbb{Z}$ at b=3). This is the regime in which the lattice choice has any practical purchase.

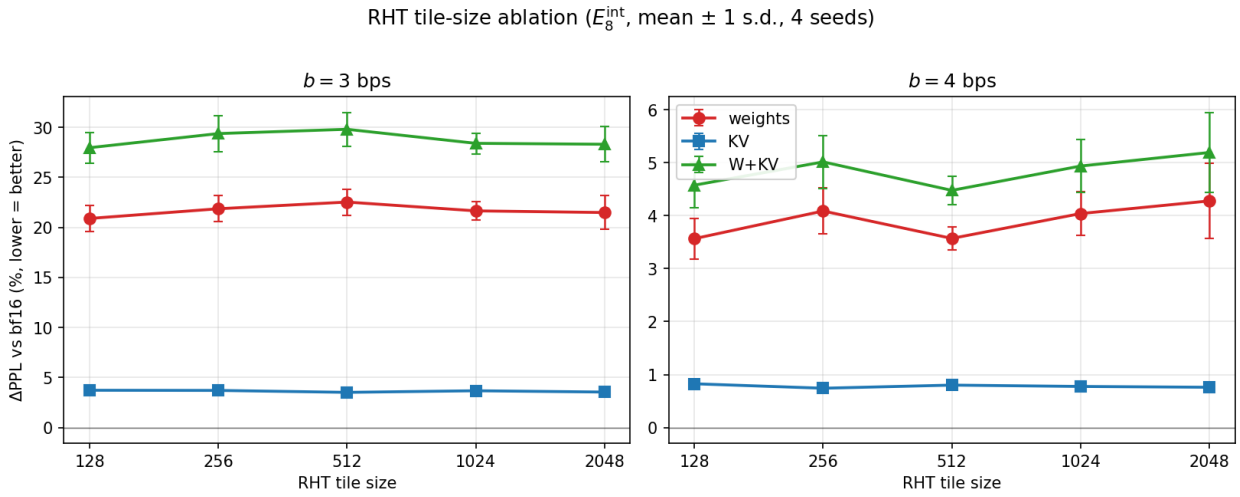


Figure 8: PPL vs. RHT tile size on Llama-3.1-8B (E_8^{int}, no MMA cast, true RHT), as ΔPPL relative to the BF16 baseline for the weight, KV, and joint W+KV paths at b=3 (left) and b=4 (right). Markers are the mean over four RHT seeds; error bars are ±1 standard deviation. KV is nearly tile-invariant with negligible seed variance, while for the weight and W+KV paths the per-tile differences fall within the seed error bars at both rates, i.e. tile size is not a quality lever. W+KV tracks the sum of the two independent paths.

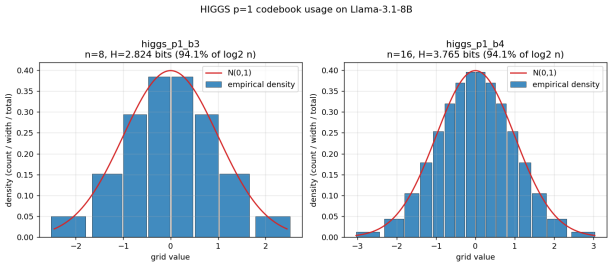
7.2 RHT tile size

The RHT tile is the block length over which we whiten and per-tile `amax`-scale before lattice quantization. We sweep it over $\{128, 256, 512, 1024, 2048\}$ on Llama-3.1-8B with E_8^{int} at $b \in \{3, 4\}$ on the weight, KV, and joint W+KV paths (Figure 8; WikiText-2 PPL over 141 windows, BF16 baseline 7.161). To measure the tile effect rather than the alignment of any one fixed Hadamard basis, we use a *true* RHT, a per-tile random sign diagonal applied before the Hadamard, and repeat the entire sweep over four independent RHT seeds, reporting the across-seed mean ±1 standard deviation. This separates a genuine tile dependence from the seed-to-seed noise of the random rotation itself. Per target b the SNR is calibrated once on iid Gaussian and held fixed across tiles; the realized rate stays within ±0.01 bps of target, and larger tiles code marginally fewer bits (≈ 0.005 bps less) by tightening the post-RHT Gaussian fit.

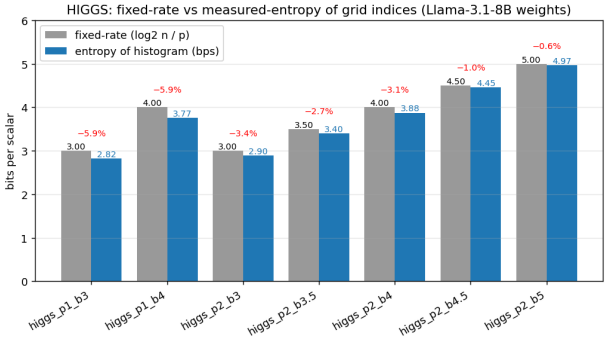
Two conclusions:

- **KV is essentially tile-insensitive**, and this is the one rock-solid effect: the per-tile means span only 3.5–3.7% at b=3 and 0.74–0.83% at b=4, with tiny seed variance (≤ 0.24 pp, mostly < 0.1). KV vectors are low-dimensional and well-conditioned per head, so the whitening block size barely matters once the tile covers a head or more. Per-head whitening (tile = 128) is a fine, cheap default.
- **For weights and W+KV, tile size is not a quality lever.** Across seeds the weight path stays at $\approx$ 21–22.5% at b=3 and $\approx$ 3.6–4.3% at b=4, with per-tile gaps (≤ 1.6 pp) that are smaller than the ±1 s.d. seed noise (≈ 1.3 pp at b=3, 0.2–0.7 pp at b=4); W+KV behaves the same way. Apparent "best" tiles from any single basis (e.g. a deterministic Hadamard, or one random seed) do not survive averaging over rotations, so the marginally better Gaussianization and lower rate of a longer transform do not translate into a reproducible PPL gain.

The W+KV damage tracks the sum of the independent paths (e.g. at b=4, tile 128: weights



(a) Index frequency for HIGGS p=1, b=3.



(b) Empirical entropy efficiency H/b for HIGGS codebooks at $b \in \{3, \ldots, 5\}$, $p \in \{1, 2\}$.

Figure 9: HIGGS codebook index distributions are non-uniform on Llama weights, with 0.6–5.9% entropy slack relative to the fixed-rate budget $\log_2 N$ (red labels). HyperQuant recovers this slack via Rice coding.

+3.57% and KV +0.83% compose to W+KV +4.58%), confirming the two error sources are roughly additive in PPL. Because the tile has no reproducible effect on quality, we set it on kernel-efficiency grounds: the default tile of 128 matches the MMA K-dim and is therefore preferred for downstream kernel fusion at no measurable accuracy cost.

7.3 HIGGS-codebook efficiency analysis

To corroborate the rate-gain decomposition in Section 6.1, we measured HIGGS's empirical codebook index histograms on Llama-3.1-8B weights (Figure 9). The Lloyd grids exhibit clear non-uniform usage: the most-used codeword is consistently 10–22× more frequent than the least-used, and the empirical index entropy is 0.6–5.9% below the fixed-rate budget $\log_2 N$ (equivalently, 94–99% coding efficiency). This is *not* a bug in HIGGS; it is the expected behavior of any finite codebook on a smooth distribution. It *is* however a recoverable bit-rate gap, which Rice coding closes.

8 Conclusion and discussion

We presented HyperQuant, a data-free post-training quantization pipeline that unifies four classical ideas, per-tile Walsh–Hadamard whitening, optimal low-dimensional lattice vector quantization, Rice entropy coding, and Schuchman–Zamir–Feder subtractive dither, into a single recipe that targets both the linear weights and the KV cache of modern transformer models. The pipeline plugs into Hopper's 8-bit and Blackwell's 4-bit MMA paths via a standard per-tile FP8/INT8 cast, which is near-optimal once the RHT has whitened each tile.

Summary of empirical findings.

- *Weight quantization*: HyperQuant's E_8+Rice path dominates HIGGS at every bps from 3 to 5. The gap decomposes into (i) a small index-entropy piece (0.6–0.8 dB across the range) that any entropy-coded HIGGS could recover, and (ii) a larger structural "unbounded-codebook" piece that even an entropy-coded HIGGS cannot match (0.67 dB at b=3, 0.91 dB at b=4, 2.34 dB at b=5), enabled by the variable-length coding that allows the lattice to be unbounded.

- *KV-cache quantization*: a clean two-regime story emerges. In the high-quality regime ($b \geq 2.5$) all bias-correction choices are equivalent; in the high-compression regime (1.7–2.5 bps) QJL rotation pulls ahead by up to ~ 0.5 PPL. Run head-to-head on OCTOPUS's own Qwen2.5-7B protocol, HyperQuant beats both TurboQuant and OCTOPUS at matched bias correction; with a matched 32-token residual window it wins on *both* quality and compression at every operating point (+7.4% vs. OCTOPUS's +34.7% perplexity at 2 bits, at KV $\times$ 6.4 vs. 2.9), and reaches KV $\times$ 7.1 at 1.7 bps where OCTOPUS tops out near 3.0×.
- *8-bit MMA*: INT8 consistently beats FP8 on Hadamard-rotated lattice data by ~ 0.1 PPL (LLM) and ~ 0.7 dB PSNR (LTX-2 video), reversing the conventional wisdom that FP8 is preferred for outlier-heavy distributions: post-RHT the distribution is no longer outlier-heavy.
- *4-bit MMA*: NVFP4 is viable; MX-FP4's E8M0 scale cannot accommodate KV-cache tails without dither rescue.
- *Generalization*: the entire pipeline transfers cleanly from an 8B language model to a 19B video DiT.

When to use HyperQuant. The defaults in Table 4 deliver $\Delta\text{PPL} \leq 0.3$ on Llama-3.1-8B at b=4 for both weights and KV with no fine-tuning, no calibration set, and a $\sim$ 30-second post-training pass. For workloads where KV-cache memory is the bottleneck (long-context decoding, batch inference, multi-tenancy) we recommend $b_{\text{kv}} = 3$, which delivers $\sim 81\%$ KV memory reduction for +0.25 PPL. For aggressive memory-constrained deployments at $b_{\text{kv}} = 2$, enable QJL rotation; below 1.7 bps the operating regime is too noisy for any data-free method we know, and either calibration-based methods or fine-tuning is needed.

Limitations.

1. *Speedups are not yet measured.* Our quality results are exact end-to-end PPL; the corresponding wall-clock speedups depend on a fused Triton kernel that is in active development and will be reported in Table 10 once available **[Pending: kernel benchmark in a future revision]**.
2. *Single-bps allocation.* HyperQuant currently uses uniform bps; the dynamic-programming bit allocator of HIGGS can be composed with our entropy code and should help in the very low-bit regime.

Future directions. The most natural extensions, in order of expected impact:

1. Fuse the Rice decoder + FP8 cast + MMA into a single Triton kernel and ship measured end-to-end decode-latency speedups vs. bf16 on H100 and B100, populating Table 10.
2. *Close the FP–INT gap with a Gaussian-aware cast.* We observed that INT8 beats FP8 on the whitened lattice output (~ 0.1 PPL on Llama, ~ 0.7 dB PSNR on LTX-2) because, after the RHT, each tile is approximately iid Gaussian and therefore light-tailed: FP8's wider exponent range buys nothing while it forfeits mantissa bits, and INT8's uniformly spaced levels are a better match. This suggests the gap is not fundamental but a property of the *cast* rule: an FP8 cast that exploits the known post-RHT Gaussian density (e.g. a companding/non-uniform pre-scale that maps the Gaussian to FP8's level spacing, or a per-tile saturation point chosen from the analytic Gaussian tail rather than the empirical amax) should let FP8 recover most of INT8's advantage and even surpass it where the residual tails matter. Because the RHT fixes the marginal distribution data-free, this cast can be derived in closed form rather than calibrated. We expect the same idea to lift NVFP4/MXFP4 on the 4-bit path.

3. *Add a calibration pass such as LDLQ.* HyperQuant is currently fully data-free; composing it with a one-shot LDLQ-style update [34, 37], which adjusts each layer's still-unquantized weights to absorb the quantization error already committed in earlier columns, should close the residual gap to calibration-based lattice methods such as NestQuant while adding only a single, data-light pass over the model.
4. Sweep per-layer bit allocation under a global bps budget, composing HIGGS's dynamic-programming allocator [23] with our entropy code; we expect the gains to concentrate in the very low-bit regime.
5. Evaluate higher-dimensional lattices: the Leech lattice Λ_{24} offers a ~ 0.4 dB granular-gain advantage over E_8 at the cost of a more expensive decoder, to be weighed against its PPL benefit at $b \in \{3, 3.5, 4\}$.

References

[1] Meta AI. The Llama-3 herd of models. Meta AI research publication, 2024. URL https://ai.meta.com/research/publications/the-llama-3-herd-of-models/.

[2] Nir Ailon and Bernard Chazelle. The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009.

[3] Anonymous. OCTOPUS: Optimized KV cache for transformers via octahedral parametrization under optimal squared error quantization, 2026. URL https://octopus-quant.github.io/. arXiv:2605.21226 (under review).

[4] Saleh Ashkboos, Maximilian L. Croci, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *ICLR*, 2024. URL https://arxiv.org/abs/2401.15024.

[5] Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *NeurIPS*, 2024. URL https://arxiv.org/abs/2404.00456.

[6] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *NeurIPS*, 2023. URL https://arxiv.org/abs/2307.13304.

[7] John H. Conway and Neil J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer, 3rd edition, 1999.

[8] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2nd edition, 2006.

[9] Anirban Dasgupta, Ravi Kumar, and Tamás Sarlós. A sparse Johnson–Lindenstrauss transform. In *STOC*, 2010.

[10] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *ICLR*, 2024. URL https://arxiv.org/abs/2306.03078.

[11] Uri Erez and Ram Zamir. On the closeness of the random-dither mapping to the information-theoretic optimum for vector quantization. *IEEE Transactions on Information Theory*, 51(10): 3617–3631, 2005.

[12] Patrick Esser et al. Scaling rectified flow transformers for high-resolution image synthesis. *arXiv preprint arXiv:2403.03206*, 2024. URL https://arxiv.org/abs/2403.03206.

[13] G. David Forney and Lee-Fang Wei. Multidimensional constellations—Part I: Introduction, figures of merit, and generalized cross constellations. *IEEE Journal on Selected Areas in Communications*, 7(6):877–892, 1989.

[14] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *ICLR*, 2023. URL https://arxiv.org/abs/2210.17323.

[15] Herbert Gish and John N. Pierce. Asymptotically efficient quantizing. *IEEE Transactions on Information Theory*, 14(5):676–683, 1968.

[16] Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011. URL https://arxiv.org/abs/0909.4061.

[17] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *NeurIPS*, 2024. URL https://arxiv.org/abs/2401.18079.

[18] Lightricks. LTX-Video: A real-time video generation model. GitHub repository, 2024. URL https://github.com/Lightricks/LTX-Video.

[19] 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. In *MLSys*, 2024. URL https://arxiv.org/abs/2306.00978.

[20] 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 *NeurIPS*, 2024. URL https://arxiv.org/abs/2405.16406.

[21] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *ICML*, 2024. URL https://arxiv.org/abs/2402.02750.

[22] Thomas D. Lookabaugh and Robert M. Gray. High-resolution quantization theory and the vector quantizer advantage. *IEEE Transactions on Information Theory*, 35(5):1020–1033, 1989.

[23] Vladimir Malinovskii, Andrei Panferov, Ivan Ilin, Han Guo, Peter Richtárik, and Dan Alistarh. Pushing the limits of large language model quantization via the linearity theorem. In *NAACL*, 2025. URL https://arxiv.org/abs/2411.17525. arXiv:2411.17525.

[24] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *ICLR*, 2017. URL https://arxiv.org/abs/1609.07843.

[25] Paulius Micikevicius et al. FP8 formats for deep learning. *arXiv preprint arXiv:2209.05433*, 2022. URL https://arxiv.org/abs/2209.05433.

[26] NVIDIA. NVIDIA hopper h100 architecture white paper. NVIDIA white paper, 2022. URL https://resources.nvidia.com/en-us-tensor-core.

[27] NVIDIA. NVIDIA blackwell architecture technical brief. NVIDIA technical brief, 2024. URL https://resources.nvidia.com/en-us-blackwell-architecture.

[28] Open Compute Project. OCP Microscaling Formats (MX) specification v1.0. OCP specification, 2023. URL https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf.

[29] Gilles Pagès and Jacques Printems. Optimal quadratic quantization for numerics: The Gaussian case. *Monte Carlo Methods and Applications*, 9(2):135–165, 2003.

[30] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. URL https://arxiv.org/abs/2212.09748.

[31] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xu, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *MLSys*, 2023. URL https://arxiv.org/abs/2211.05102.

[32] Robert F. Rice. Some practical universal noiseless coding techniques. *JPL Publication 79-22*, 1979.

[33] Bita Darvish Rouhani, Nitin Garegrat, Tom Madian, Jeremy Lo, Brian Cook, Daniel Pinto, et al. Microscaling data formats for deep learning. *arXiv preprint arXiv:2310.10537*, 2023. URL https://arxiv.org/abs/2310.10537.

[34] Semyon Savkin, Eitan Porat Chen, Or Lou, and Yury Polyanskiy. NestQuant: Nested lattice quantization for matrix products and LLMs. In *ICML*, 2025. URL https://arxiv.org/abs/2502.09720. arXiv:2502.09720.

[35] Leonard Schuchman. Dither signals and their effect on quantization noise. *IEEE Transactions on Communication Technology*, 12(4):162–165, 1964.

[36] Wenqi Shao, Mengzhao Chen, Zhaoyang Zhang, Peng Xu, Lirui Zhao, Zhiqian Li, Kaipeng Zhang, Peng Gao, Yu Qiao, and Ping Luo. OmniQuant: Omnidirectionally calibrated quantization for large language models. In *ICLR*, 2024. URL https://arxiv.org/abs/2308.13137.

[37] Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with hadamard incoherence and lattice codebooks. In *ICML*, 2024. URL https://arxiv.org/abs/2402.04396.

[38] 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 *ICML*, 2023. URL https://arxiv.org/abs/2211.10438.

[39] Ram Zamir. *Lattice Coding for Signals and Networks*. Cambridge University Press, 2014.

[40] Ram Zamir and Meir Feder. On universal quantization by randomized uniform/lattice quantizers. *IEEE Transactions on Information Theory*, 38(2):428–436, 1992.

[41] Amir Zandieh, Majid Daliri, and Insu Han. QJL: 1-bit quantized JL transform for KV cache quantization with zero overhead. *arXiv preprint arXiv:2406.03482*, 2024. URL https://arxiv.org/abs/2406.03482.

[42] Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. TurboQuant: Online vector quantization with near-optimal distortion rate. In *ICLR*, 2026. URL https://arxiv.org/abs/2504.19874. arXiv:2504.19874.

A Proof of subtractive-dither unbiasedness

This appendix gives a self-contained proof that subtractive-dithered lattice quantization satisfies the Schuchman conditions and is hence *exactly* unbiased under inner products on every realization: for any deterministic query q and source x, the dithered reconstruction $\hat{x}$ satisfies $\mathbb{E}_U[\langle q, \hat{x}\rangle \mid x] = \langle q, x\rangle$ where the expectation is over the dither U alone. We then show the guarantee survives composition with the full HyperQuant KV pipeline and contrast it with the weaker approximate unbiasedness of QJL-without-dither.

A.1 Setup and notation

Definition 1 (Lattice and Voronoi cell)**.** A lattice $\Lambda \subset \mathbb{R}^n$ is a discrete subgroup of $(\mathbb{R}^n, +)$ of full rank n, i.e., $\Lambda = \{\sum_{i=1}^n k_i b_i : k_i \in \mathbb{Z}\}$ for some $\mathbb{R}$-basis $b_1, \ldots, b_n$ of $\mathbb{R}^n$. The Voronoi cell of Λ at the origin is

$$\mathcal{V}(\Lambda) := \{x \in \mathbb{R}^n : \|x\| \le \|x - \lambda\| \text{ for all } \lambda \in \Lambda\}. \tag{6}$$

The Voronoi cell $\mathcal{V}(\Lambda)$ is a closed convex polytope. It is centrally symmetric, $\mathcal{V}(\Lambda) = -\mathcal{V}(\Lambda)$, because the defining inequalities (6) are invariant under $x \mapsto -x$ together with $\lambda \mapsto -\lambda$ (which is a bijection of Λ). The translates $\{\mathcal{V}(\Lambda) + \lambda : \lambda \in \Lambda\}$ tile $\mathbb{R}^n$, overlapping only on the measure-zero boundary $\partial\mathcal{V}(\Lambda)$.

Definition 2 (Fundamental domain)**.** A measurable set $D \subset \mathbb{R}^n$ is a fundamental domain for Λ if $\mathbb{R}^n = \bigsqcup_{\lambda\in\Lambda}(D + \lambda)$ up to sets of Lebesgue measure zero.

By the tiling property, $\mathcal{V}(\Lambda)$ is itself a fundamental domain. The covolume of Λ is $\mathrm{vol}(\mathcal{V}(\Lambda)) = |\det(b_1, \ldots, b_n)|$.

Definition 3 (Nearest-neighbor quantizer and mod-Λ projection)**.** Define

$$Q_\Lambda(y) := \arg\min_{\lambda\in\Lambda} \|y - \lambda\|, \qquad \pi_\Lambda(y) := y - Q_\Lambda(y),$$

with a fixed measurable tie-break rule on $\partial\mathcal{V}(\Lambda)$. The map π_Λ sends $y \in \mathbb{R}^n$ to the unique representative of $y + \Lambda$ in $\mathcal{V}(\Lambda)$ (uniqueness up to the boundary).

The map π_Λ has two properties we will use repeatedly:

(P1) *Range.* $\pi_\Lambda(\mathbb{R}^n) \subseteq \mathcal{V}(\Lambda)$.

(P2) *Lattice periodicity.* $\pi_\Lambda(y+\lambda) = \pi_\Lambda(y)$ for all $\lambda \in \Lambda$ and $y \in \mathbb{R}^n$, because $Q_\Lambda(y+\lambda) = Q_\Lambda(y)+\lambda$. Hence π_Λ descends to a well-defined map $\mathbb{R}^n/\Lambda \to \mathcal{V}(\Lambda)$.

A.2 The mod-Λ pushforward is uniform

Lemma 1. *Let $D \subset \mathbb{R}^n$ be a fundamental domain of Λ with finite positive measure. If $Y \sim \mathrm{Uniform}(D)$, then $\pi_\Lambda(Y) \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$.*

Proof. Let $A \subseteq \mathcal{V}(\Lambda)$ be measurable. We have $\Pr[\pi_\Lambda(Y) \in A] = \mathrm{vol}(\pi_\Lambda^{-1}(A) \cap D)/\mathrm{vol}(D)$. By (P2), $\pi_\Lambda^{-1}(A) = \bigsqcup_{\lambda\in\Lambda}(A + \lambda)$, which tiles $A + \Lambda$ exactly once when restricted to any fundamental domain D. Hence $\mathrm{vol}(\pi_\Lambda^{-1}(A) \cap D) = \mathrm{vol}(A)$, so $\Pr[\pi_\Lambda(Y) \in A] = \mathrm{vol}(A)/\mathrm{vol}(D) = \mathrm{vol}(A)/\mathrm{vol}(\mathcal{V}(\Lambda))$, which is the uniform-on-$\mathcal{V}(\Lambda)$ probability. □

The lemma has the following "shift-invariance" consequence, which is the engine of the proof.

Corollary 1 (Crypto Lemma)**.** *Let $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$. For every deterministic $x \in \mathbb{R}^n$,*

$$\pi_\Lambda(x + U) \sim \mathrm{Uniform}(\mathcal{V}(\Lambda)), \quad \textit{independent of } x.$$

Proof. Since $\mathcal{V}(\Lambda)$ is a fundamental domain, so is the translate $x + \mathcal{V}(\Lambda)$. The variable $Y := x + U$ has distribution $\mathrm{Uniform}(x + \mathcal{V}(\Lambda))$, and Lemma 1 applies with $D = x + \mathcal{V}(\Lambda)$. □

A.3 Subtractive dither produces an unbiased estimator

Definition 4 (Dithered reconstruction)**.** Fix $x \in \mathbb{R}^n$ and let $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$ be independent of any other randomness. The subtractive-dithered reconstruction of x is

$$\hat{x}(x; U) := Q_\Lambda(x + U) - U. \tag{7}$$

The quantization error is $e(x; U) := \hat{x}(x; U) - x$.

Theorem 1 (Schuchman)**.** *With U and $\hat{x}$ as in Definition 4, for every $x \in \mathbb{R}^n$ the error $e(x; U)$ is distributed as $-U' \sim \mathrm{Uniform}(-\mathcal{V}(\Lambda))$, independent of x. In particular, by central symmetry, $e(x; U) \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$ as well, and*

$$\mathbb{E}_U[e(x; U) \mid x] = 0. \tag{8}$$

Proof. Expand the error:

$$e(x; U) = \hat{x}(x; U) - x = Q_\Lambda(x + U) - U - x = -((x + U) - Q_\Lambda(x + U)) = -\pi_\Lambda(x + U).$$

By the Crypto Lemma (Corollary 1), $\pi_\Lambda(x + U) \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$, so $e(x; U) = -\pi_\Lambda(x + U) \sim \mathrm{Uniform}(-\mathcal{V}(\Lambda))$ independent of x. The mean is zero because $\mathcal{V}(\Lambda)$ is centrally symmetric. □

Corollary 2 (Inner-product unbiasedness)**.** *For any deterministic $q \in \mathbb{R}^n$ and any source $x \in \mathbb{R}^n$,*

$$\mathbb{E}_U[\langle q,\, \hat{x}(x; U)\rangle \mid x] = \langle q,\, x\rangle.$$

Proof. By linearity of expectation and Theorem 1, $\mathbb{E}_U[\langle q, \hat{x}\rangle \mid x] = \langle q, x\rangle + \langle q, \mathbb{E}_U[e \mid x]\rangle = \langle q, x\rangle + \langle q, 0\rangle = \langle q, x\rangle$. □

Corollary 3 (Variance bound)**.** *With the same notation, and writing $\sigma_\mathcal{V}^2 := \frac{1}{n}\mathbb{E}_{U\sim\mathrm{Uniform}(\mathcal{V})}[\|U\|^2]$ for the per-coordinate second moment of the Voronoi cell,*

$$\mathrm{Var}_U(\langle q, \hat{x}\rangle \mid x) = q^\top \mathrm{Cov}_U(U) q \le \lambda_{\max}(\mathrm{Cov}_U(U))\|q\|^2.$$

If $\mathcal{V}(\Lambda)$ is isotropic, i.e., $\mathrm{Cov}(U) = \sigma_\mathcal{V}^2 I_n$, this becomes $\mathrm{Var}_U(\langle q, \hat{x}\rangle \mid x) = \sigma_\mathcal{V}^2\|q\|^2$.

Remark. For the lattices used in HyperQuant ($\mathbb{Z}$, A_2, D_4, E_8), the Voronoi cell is isotropic — the lattice's symmetry group acts irreducibly on $\mathbb{R}^n$, and by Schur's lemma any invariant rank-2 tensor is a scalar multiple of I_n. Hence $\mathrm{Cov}(U) = \sigma_\mathcal{V}^2 I_n$ exactly. Numerically, $\sigma_\mathcal{V}^2(\mathbb{Z}) = 1/12 = 0.0833$ and $\sigma_\mathcal{V}^2(E_8) \approx 0.287$ in our scaling.

A.4 Composition with the HyperQuant KV pipeline

Proposition 1. *Let R be any orthogonal matrix ($R^\top R = I$), let $\alpha > 0$ be the lattice's calibration scale, and let $\hat{x}$ be the HyperQuant KV reconstruction of x:*

$$\begin{aligned} s(x) &:= \frac{\alpha\sqrt{n}}{\|x\|} \cdot Rx, \\ \widehat{s} &:= Q_\Lambda(s(x) + U) - U, \\ \hat{x} &:= \frac{\|x\|}{\alpha\sqrt{n}} R^\top \widehat{s}. \end{aligned}$$

Then $\mathbb{E}_U[\hat{x} \mid x] = x$, and in particular $\mathbb{E}_U[\langle q, \hat{x}\rangle \mid x] = \langle q, x\rangle$ for every deterministic $q \in \mathbb{R}^n$.

Proof. Apply Theorem 1 in the lattice's coordinate system to $s := s(x)$: $\mathbb{E}_U[\widehat{s} \mid s] = s$. The post-quantization map $\widehat{s} \mapsto (\|x\|/(\alpha\sqrt{n}))R^\top \widehat{s}$ is linear and depends only on x (not on U), so by linearity of conditional expectation, $\mathbb{E}_U[\hat{x} \mid x] = (\|x\|/(\alpha\sqrt{n}))R^\top s(x) = R^\top Rx = x$. Inner-product unbiasedness follows. □

Proposition 1 holds for any orthogonal R — deterministic identity, deterministic permutation, random Haar, random sign diagonal — including when R is itself random but *independent* of U, because the proof conditions on R and then averages.

A.5 Contrast: QJL alone is biased per-vector

The QJL-without-dither variant uses random rotation but no dither, so its reconstruction is

$$\hat{x}_{\mathrm{QJL}}(x; S) := S^\top Q_\Lambda(Sx), \qquad S \sim \mathrm{Uniform}(\mathrm{O}(n)), \tag{9}$$

with error $e_{\mathrm{QJL}}(x; S) = -S^\top \pi_\Lambda(Sx)$.

Proposition 2. *For every fixed realization $S = S_0 \in \mathrm{O}(n)$ and every fixed source $x \in \mathbb{R}^n$, the QJL error is a deterministic vector $-S_0^\top \pi_\Lambda(S_0 x)$, generally non-zero, so $\hat{x}_{\mathrm{QJL}}(x; S_0)$ is a biased estimator of x. There is no per-vector analog of Corollary 2 for QJL alone.*

Proof. Self-evident from (9): with S_0 fixed, neither side of the equation depends on any further randomness. □

Why the empirical QJL bias appears small. In a typical sweep test one averages $\frac{1}{N}\sum_{i=1}^N \langle q^{(i)}, e_{\mathrm{QJL}}(x^{(i)}; S_0)\rangle$ over many iid $(q^{(i)}, x^{(i)})$. Because q is independent of everything else and has zero mean, the sample average converges to $\langle \mathbb{E}q, \cdot\rangle = 0$. This is a consequence of $\mathbb{E}q = 0$, *not* of any QJL property; a non-rotated lattice quantizer passes the same empirical test. What QJL *does* provide is approximately isotropic error covariance: $S_0^\top \mathrm{Cov}_x(\pi_\Lambda(S_0 x))S_0$ is close to a scalar multiple of I_n when S_0 is a generic Haar draw. This is the genuine, non-vacuous benefit of the random rotation, but it is not the same property as unbiasedness.

A.6 Practical sampler

The proof requires $U \sim \mathrm{Uniform}(\mathcal{V}(\Lambda))$. The implementation uses the "mod-Λ trick":

$$U_{\mathrm{cube}} \sim \mathrm{Uniform}([-2, 2)^n), \quad U := \pi_\Lambda(U_{\mathrm{cube}}).$$

This is exact iff $[-2,2)^n$ is itself a fundamental domain of Λ (Lemma 1). For E_8 the cube $[-2,2)^8$ is not a union of Λ-translates of $\mathcal{V}(\Lambda)$, so the projection is only *approximately* uniform. We empirically validate the sampler by checking $\frac{1}{n}\mathbb{E}\|U\|^2$ matches the analytical $\sigma^2_{\mathcal{V}}$ to within 5% on all four lattices. An exact sampler is the fundamental-parallelepiped variant: sample $T_i \overset{\text{iid}}{\sim} \text{Uniform}[0,1)$ for $i = 1,\dots,n$, form $Y = \sum T_i b_i$ where $\{b_i\}$ is the lattice basis, and project $U := \pi_\Lambda(Y)$; this is uniform on $\mathcal{V}(\Lambda)$ exactly by Lemma 1 since the parallelepiped is by construction a fundamental domain.

B Calibration: setting the operating point

HyperQuant exposes one user-facing knob, the target rate b in bits/scalar, and turns it into a concrete quantizer in two steps: choose the quantization SNR that yields b, then set the per-vector scale α that realizes that SNR. The first step needs an empirical rate curve $b(\text{SNR})$, because the Rice-coded rate has no closed form; the second is a closed-form formula in the lattice's Voronoi second moment. Both are built once, on synthetic iid-Gaussian data, and, crucially, apply unchanged to every weight and KV tensor in any model (Section B.3). This is what lets HyperQuant hit an *arbitrary* fractional rate, a property fixed-rate codebooks cannot match.

B.1 From a target rate to an SNR (empirical)

The rate of the stripped, Rice-coded stream is the composition of the lattice's granular gain, the bit-stripping transform, an *integer*-parameter Rice coder, and the 8-bit clip, none of which admits a clean closed form at the operating points of interest. We therefore measure it. For each lattice we draw $N \sim 10^5$ iid-Gaussian tiles, quantize at a grid of SNRs (the closed-form α of Section B.2 sets each point), and record the *realized* Rice rate. Figure 10 is the result.

Two properties make this usable. *(i) Invertibility.* $b(\text{SNR})$ is monotone increasing, so the implementation interpolates the tabulated curve to recover the unique SNR achieving any requested b; because the table stores the realized Rice rate (not an entropy estimate), selecting against it hits any target rate to within $\sim$0.01 bps (Section 6.1 confirms this on-model). *(ii) Tightness.* The realized rate sits ≈ 0.1 bps above the lattice ideal. This gap is almost entirely the redundancy of the stateless, power-of-two Rice code over the symbols' *marginal* entropy; that marginal entropy already meets the lattice ideal (Section B.4), so the 0.1 bps reflects the coder's simplicity, not any residual lattice or inter-symbol inefficiency.

The per-lattice Rice structure is what the table is the rate *of*: E_8 uses a single parameter k_s (the coset bit folded into the combined symbol); D_4 and A_2 use two (k_s with $k_t = k_s - 1$ for D_4, and k_{t_y}, k_{n_x} for A_2); $\mathbb{Z}$ uses one. The halving identity $k_t = k_s - 1$ is checked at runtime (Section 4.7).

B.2 From an SNR to the scale (closed form)

Given the SNR, the scale is analytic. For $x \sim \mathcal{N}(0, I_N)$ the high-rate quantization error has mean square equal to the lattice's Voronoi second moment, $\text{MSE}_{\text{vor}} = G(\Lambda)\, N\, V_\Lambda^{2/N}$, while the signal power is $\mathbb{E}\|\alpha x\|^2 = \alpha^2 N$. Setting their ratio to the target $\text{SNR}_{\text{lin}} = 10^{\text{SNR}/10}$ gives

$$\alpha(\text{SNR}, \Lambda) \;=\; \sqrt{\frac{\text{SNR}_{\text{lin}}\,\text{MSE}_{\text{vor}}}{N}} \;=\; \sqrt{\text{SNR}_{\text{lin}}\; G(\Lambda)\, V_\Lambda^{2/N}}, \tag{10}$$

with V_Λ the covolume of the integer realization. No calibration data are needed for α; the only assumption is the high-rate approximation, which we verify: the empirical Voronoi second moment is

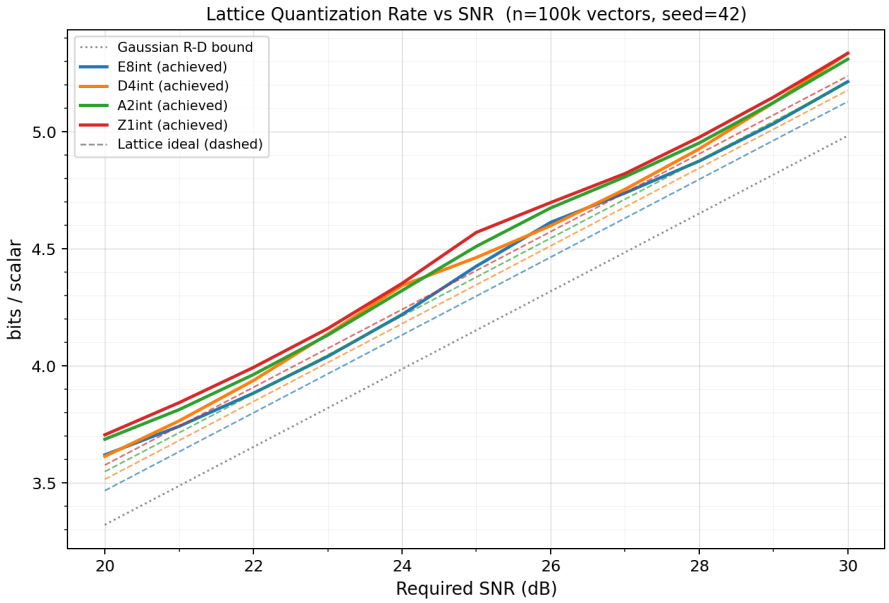


Figure 10: Empirical Rice rate versus target SNR on iid-Gaussian tiles (N=10^5, seed 42), for the four lattices. Solid: achieved bits/scalar; dashed: the lattice ideal $R_D + \frac{1}{2}\log_2(2\pi e\, G(\Lambda))$; dotted: the Gaussian rate–distortion bound $R_D = \frac{1}{2}\log_2 \mathrm{SNR}_{\mathrm{lin}}$. Each curve is smooth and monotone, so inverting it sends any target rate to a unique SNR; the achieved rate stays ≈ 0.1 bps above the lattice ideal throughout, ordering $E_8 < D_4 < A_2 < \mathbb{Z}$.

within 5% of $\mathrm{MSE}_{\mathrm{vor}}$ for E_8, D_4, A_2 (within 20% for $\mathbb{Z}$, whose tiny cell is sensitive to bf16 rounding), and the realized SNR lands within ≈ 0.1 dB of the target across 20–30 dB.

Equation (10) also fixes the code magnitudes, and hence the 8-bit budget. Across the full 20–30 dB sweep the simulation records *zero* overflows: $P(|c_i| > 127)$ is exactly 0 at every operating point. With $\sim 10^5$–10^6 codes sampled per point, observing no events bounds the true rate only at $\lesssim 10^{-5}$ (rule of three); it cannot be estimated more precisely, and the 10^{-12} floor in Figure 11 (bottom) is merely a plotting device so the empty bins render on the log axis. The operative fact is that no code ever left the one-signed-byte range. The top panel previews the rate sitting just above the lattice ideal (detailed in Section B.1).

B.3 One calibration for all data

Both pieces above are built on iid-Gaussian tiles, yet HyperQuant applies them to real weights and KV activations with no per-tensor recalibration. The justification is the per-tile RHT followed by ℓ_2 normalization (Sections 4.1 and 4.3): the Hadamard rotation spreads each tile's energy across its coordinates, and normalization fixes its radius, so every tile is approximately an isotropic Gaussian on the sphere of radius $\alpha\sqrt{n}$, precisely the distribution the calibration was measured on. A single table and the closed form of Equation (10) therefore serve all tensors, layers, and models, which is what makes HyperQuant calibration-data-free.

B.4 Stripping is rate-optimal: marginal entropy meets the lattice ideal

A striking feature of the sweep is that the per-scalar *marginal* entropy of the stripped symbols ($\frac{1}{N}\sum_i H(s_i)$, reported as `scalar_entropy`) coincides with the lattice ideal $R_{\mathrm{ideal}} = R_D +$

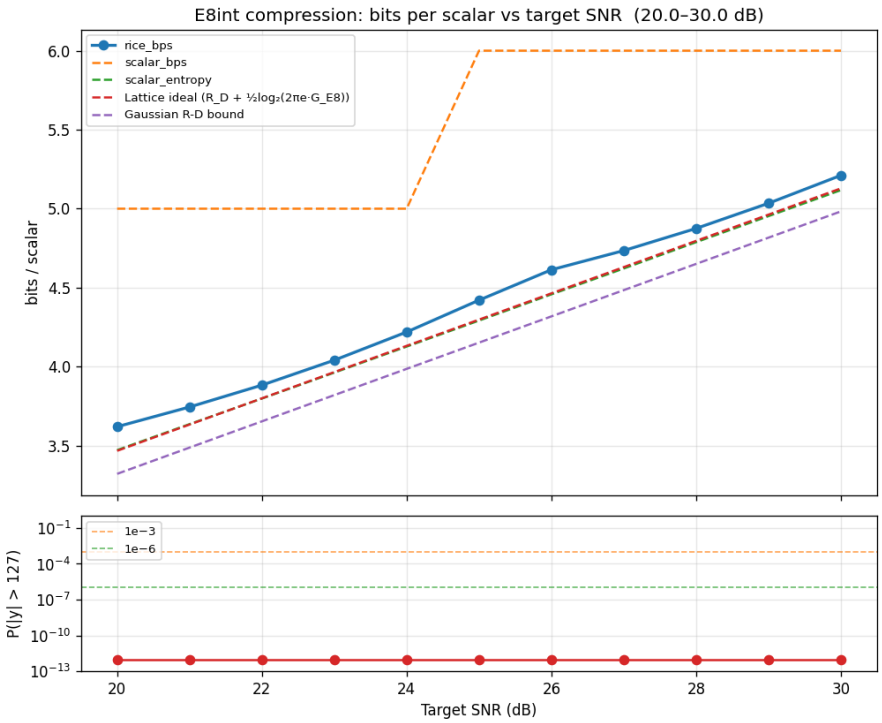


Figure 11: E_8 calibration detail. *Top:* achieved Rice rate tracks the lattice ideal within ≈ 0.1 bps over 20–30 dB (the dashed `scalar_bps` is the byte-aligned int8 fallback, which steps with the integer byte budget; the `scalar_entropy` curve, the marginal entropy of the stripped indices, lies essentially on the lattice ideal, cf. Section B.4). *Bottom:* no 8-bit overflow was ever sampled: the measured $P(|c_i| > 127)$ is 0 at every operating point; markers are floored at 10^{-12} only so they render on the log axis (guides at 10^{-3} and 10^{-6}). The sample size bounds the true rate at $\lesssim 10^{-5}$, no tighter.

$\frac{1}{2}\log_2(2\pi e\, G(\Lambda))$ to within $\sim 10^{-2}$ bps at every operating point and for every lattice (Figure 11, top, where the two curves overlie). This is not a coincidence: it is the composition of two classical high-resolution results with the design of the strip.

(i) The index entropy equals the lattice ideal. For a source X with finite differential entropy quantized by a lattice Λ at fine resolution, the index entropy obeys the high-resolution law

$$H\big(Q_\Lambda(X)\big) \;=\; h(X) \;-\; \log_2 \mathrm{vol}(\mathcal{V}(\Lambda)) \;+\; o(1), \tag{11}$$

the $o(1)$ vanishing as the cell shrinks [15, 22]. With per-dimension distortion $D = G(\Lambda)\ \mathrm{vol}(\mathcal{V}(\Lambda))^{2/N}$ [7, Ch. 3] and a white Gaussian source $X \sim \mathcal{N}(0, \sigma^2 I_N)$, for which $h(X)/N = \frac{1}{2}\log_2(2\pi e\,\sigma^2)$, dividing (11) by N gives

$$\frac{1}{N} H\big(Q_\Lambda(X)\big) \;\longrightarrow\; \underbrace{\tfrac{1}{2}\log_2 \frac{\sigma^2}{D}}_{R_D} \;+\; \tfrac{1}{2}\log_2(2\pi e\, G(\Lambda)) \;=\; R_{\mathrm{ideal}}. \tag{12}$$

The excess $\frac{1}{2}\log_2(2\pi e\, G(\Lambda))$ over the Gaussian rate–distortion bound R_D is the lattice's space-filling loss, exactly the redundancy of an entropy-coded dithered lattice quantizer above $R(D)$ [11, 39, 40].

(ii) Stripping reduces the marginal sum to the joint entropy. The quantity `scalar_entropy` is not the joint entropy (12) but the per-scalar *sum of marginals*, $\frac{1}{N}\sum_i H(s_i)$. The strip Strip_Λ is a lossless bijection $\Lambda \leftrightarrow \mathbb{Z}^N$ (Section 4.6), hence preserves the joint entropy, $H(s_1, \ldots, s_N) = H(Q_\Lambda(X))$, and by subadditivity

$$\frac{1}{N}\sum_i H(s_i) \;=\; \frac{1}{N}H(Q_\Lambda(X)) \;+\; \frac{C}{N}, \qquad C \;=\; \sum_i H(s_i) - H(s_1, \ldots, s_N) \;\geq\; 0, \tag{13}$$

with C the total correlation (multi-information) of the symbols [8, Ch. 2]. Two design choices send $C \to 0$. First, the strip is built to annihilate *exactly* the lattice's deterministic dependencies, the parity and coset constraints of Section 4.6, the only exact couplings among the integer coordinates. Second, the residual *statistical* dependence vanishes in the high-rate limit: as the cell shrinks $Q_\Lambda(X) \to X$, so each stripped symbol converges to a scaled copy of an i.i.d. Gaussian source coordinate ($s_i \to \alpha X_i/2$ for E_8^{int}, and likewise for the others) and the symbols become mutually independent. The subtractive dither of Corollary 1 makes this precise: it renders the quantization error independent of X [35, 40], removing the input-dependent part of the residual correlation at any rate. Hence $C/N \to 0$ and, combining (13) with (12), $\frac{1}{N}\sum_i H(s_i) \to R_{\mathrm{ideal}}$.

Residual gap and consequence. The measured discrepancy is $\lesssim 0.02$ bps and changes sign across the sweep: the non-negative total correlation C/N pushes `scalar_entropy` *above* R_{ideal}, while finite-rate corrections to (11) and the ≈ 0.06 dB bf16 deficit between the requested SNR (at which R_{ideal} is evaluated) and the achieved SNR (at which the entropy is measured) push it *below*; both are $O(10^{-2})$ at these rates. The consequence justifies the strip-then-Rice design: because the stripped symbols are statistically near-independent and their marginal entropy already sits at the lattice ideal, a *memoryless* entropy coder is near rate-optimal: no context model or joint coder can recover more than the vanishing C/N. Stripping is thus rate-optimal by construction, not a heuristic, and it is what lets the stateless Rice coder (Section 4.7) give up only ~ 0.1 bps to the ideal.