IQR: GPTQ Integer Quantisation on a Rotated Int8 Grid

Technical report on IQR-V1 for Qwen3.8-27B

Deadcode
radiance project

October 2026

Abstract

IQR is a family of weight quantisation formats in which every format, from two to eight bits per weight, reconstructs onto the same grid: an int8 code times a bf16 scale per (row, 128 inputs) of a weight rotated by a 128-point Walsh–Hadamard transform. A container stores each tensor in its own narrow form (signed or asymmetric integers, integer Lloyd codebooks, or four-wide vector codebooks), which stays narrow in memory. The GEMM decodes it to exact int8 operands in-kernel, so every size shares one integer matrix core, one scale fold and one rotated int8 activation path, while memory and bandwidth scale with the format's bits. Codes are fitted by GPTQ against calibration Hessians, with scales chosen as GPTQ reaches each group and coordinate descent afterwards. Bits are allocated across layer bands by an additive sensitivity model. On Qwen3.8-27B, thirteen IQR-V1 recipes from 2.47 to 6.94 bpw each have 6–44% lower mean KL divergence than the Unsloth dynamic GGUF of the same size, measured on a private benchmark of 1.45M positions that includes 256K-token agentic traces.

1 Introduction

Low-bit LLM formats usually pair each bit width with its own arithmetic: a dequantisation to 16-bit floats, or a dedicated kernel per format. On hardware whose integer matrix units run int8 and fp8 at the same rate, a different design is available: make every narrower format a lossless *encoding* of int8 values. Every kernel then shares the 8-bit model's int8 matrix core, activations and scale fold, and a format adds only a decode stage that turns its codes into int8 operands between memory and the matrix units. The format decides how many bits sit in memory and cross the bus; the arithmetic is the 8-bit model's.

IQR starts from IQ8R, an int8 GPTQ quantisation of Qwen3.8-27B served by the radiance engine on two AMD Radeon AI PRO R9700 (RDNA4, gfx1201) GPUs with tensor parallelism. IQ8R scores a mean KL of 0.00046 on the benchmark of Section 6, against 0.00389 for the official FP8 checkpoint. IQR-V1 keeps IQ8R's recipe (rotation, GPTQ, integer codes, int8 activations) and spends fewer bits on the weights. The contributions are:

- a set of narrow encodings that all decode exactly to the int8 grid (Section 3);
- GPTQ extensions for codebook and vector grids, with scales chosen as GPTQ reaches each group, and a convert 2.5× faster with byte-identical codes (Section 4);
- an additive band-sensitivity model that predicts a mix's KL within a few percent and chooses each recipe (Section 5);
- a ladder of thirteen recipes, each of which beats the GGUF at its size (Section 6).

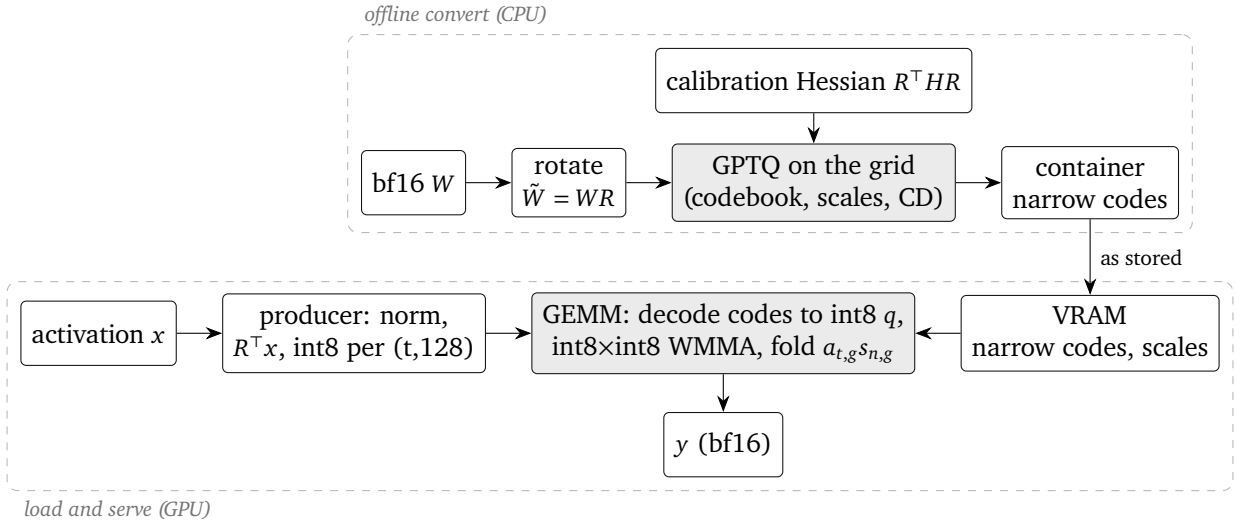

Figure 1. The IQR pipeline. Every format is fitted on the rotated int8 grid offline and kept in its own narrow form, in the container and in VRAM. The GEMM decodes each code to its exact int8 value on the way to the matrix units, so all sizes share the activation producers, the matrix core and the scale fold.

2 The rotated int8 grid

A linear layer computes $y = Wx$ with $W \in \mathbb{R}^{N\times K}$. Let $R = I_{K/128} \otimes \frac{1}{\sqrt{128}} H_{128}$ be the block-diagonal orthonormal Walsh–Hadamard transform. Since $RR^\top = I$, $y = (WR)(R^\top x)$. IQR quantises the rotated weight $\tilde{W} = WR$ and the rotated activation $\tilde{x} = R^\top x$:

$$\tilde{W}_{n,k} \approx s_{n,g}\, q_{n,k}, \qquad q_{n,k} \in \{-127, \dots, 127\}, \quad g = \lfloor k/128 \rfloor, \tag{1}$$

with a bf16 scale $s_{n,g}$ per row and 128-wide group. Activations are quantised per token and group to int8 codes $\hat{x}_{t,k}$ with scales $a_{t,g}$. The norm and gating kernels that produce each GEMM input write them already rotated, so the transform costs no separate pass. The GEMM then accumulates exact integer dot products per 128-block and folds the scales once per block:

$$y_{t,n} = \sum_g a_{t,g}\, s_{n,g} \sum_{k\in g} \hat{x}_{t,k}\, q_{n,k}. \tag{2}$$

The rotation spreads outliers across each block (incoherence processing, as in QuIP and QuaRot [2, 3]). This flattens the weight distribution, which every narrower format below relies on, and it keeps int8 activations accurate.

3 Formats

All formats share the scale layout of Section 2 and differ only in how a code maps to q (Table 1). The GEMM's decode stage maps each code to its int8 value, so the matrix core, the scales and the activations are the same for every size, and the weights take only the format's own bits in memory.

Codebooks. The scalar tables are integer Lloyd–Max quantisers [4] fitted to rotated, scale-normalised weights. The vector codebooks are four-wide integer k-means codebooks fitted the same way. Because the rotated weights are close to Gaussian, the tables are denser near zero than a uniform grid (Figure 2).

form	stored per weight	decoded as	used at
signed integer	i_N, $N = 2..7$	sign extension	–
asymmetric integer	u_N, plus a u_N zero per group	$c - z$	6 bits
scalar codebook	u_N index into 2^N int8 levels	the table entry	4, 5 bits
vector codebook	u_M index per 4 weights into E int8 4-vectors	the codeword	2–3 bits

Table 1. IQR-V1 forms. A scale costs 0.125 bpw (bf16 per 128). The vector codebooks have $E = 256, 512, 1024, 4096$ entries: 2, 2.25, 2.5 and 3 code bits per weight.

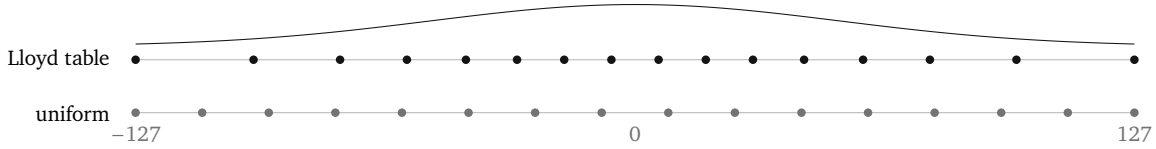

Figure 2. The 16-level integer codebook of the 4-bit form against a uniform 16-level grid, with a Gaussian of matching spread for reference.

4 The quantiser

GPTQ [1] minimises the layer's output error on calibration inputs X,

$$\mathcal{L}(\hat{W}) = \mathrm{tr}\big((\tilde{W} - \hat{W})\, H\, (\tilde{W} - \hat{W})^\top\big), \qquad H = R^\top X X^\top R + \lambda I, \tag{3}$$

with $\lambda = 0.05$ times the mean diagonal. With U the upper Cholesky factor of H^{-1}, GPTQ quantises column j, forms the error $e_j = (\tilde{w}_j - \hat{w}_j)/U_{jj}$ and updates the later columns by $\tilde{w}_{j'} \mathrel{-}= e_j U_{jj'}$ for $j' > j$. IQR adds the following to the standard algorithm.

- **Dynamic scales** (`dyn=1`). A group's scale is chosen when GPTQ reaches the group's first column, from the weights as the earlier errors have moved them, rather than from the original weights.
- **Searched scales** (`rule=search`). Candidate scales are tried per group and the one with the smallest group error is kept.
- **Vector codewords.** At the first column of each run of four, the codeword nearest the error-updated vector is chosen. Its four columns then feed their errors forward one at a time, as scalar columns do.
- **Coordinate descent** (`cd=`n). After the GPTQ loop, n passes re-choose each code, or each whole codeword, to minimise $\mathcal{L}$ exactly with the others fixed.
- **Head.** The output head is fitted by GPTQ against its own input Hessian rather than rounded to nearest.

The calibration Hessians come from a disjoint 10M-token calibration corpus, accumulated over the full (unsplit) inputs of every projection.

Convert speed. A threaded rotation of H, one Cholesky factorisation followed by a triangular inverse in place of a three-step LAPACK path, and a tiled, vectorised update of the trailing columns cut the IQ8R convert from 88 to 35 minutes on 16 CPU threads, with byte-identical codes. Mixes are not converted from scratch: each is assembled from whole-trunk *base* converts, one per grid, by copying every tensor whose options match a base. A mix then takes a few minutes, which made about fifty scored candidates feasible.

Layer error. Figure 3 shows the relative output error $\mathrm{tr}(EHE^\top)/\mathrm{tr}(\tilde{W}H\tilde{W}^\top)$ of each form on one projection. The error falls about 4× per bit. The Lloyd tables beat the asymmetric uniform grid by 9–25% at two to five bits, and the asymmetric grid wins at six. The vector codebooks beat the scalar ones by 21% at two bits and 18% at three, and each half bit of codebook roughly halves the error.

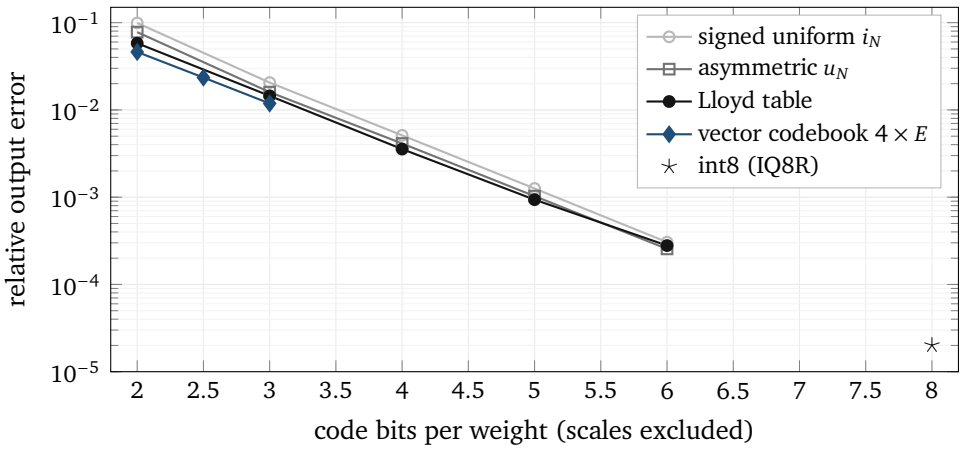

Figure 3. Relative output error of layer 3's `ffn_gate_up` (34816×5120) under GPTQ for each form. Vector codebooks with one descent pass. Lower is better.

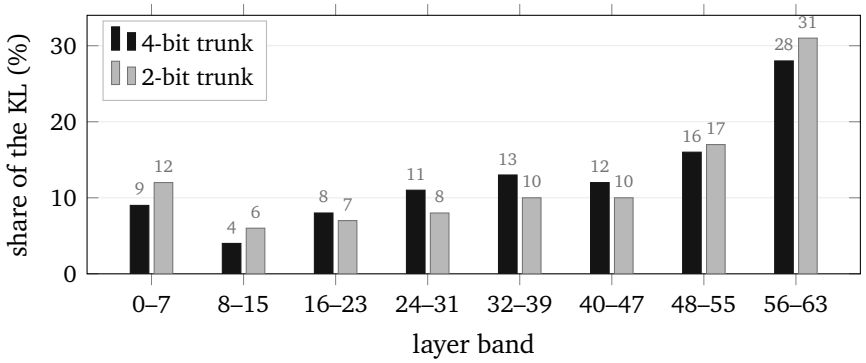

Figure 4. Band shares σ_b of the KL. The last eight layers carry 28–31% of it at both rates.

5 Allocating bits

Measured on whole models, the KL contributed by quantising a band of layers adds up across bands to within a few percent. A mix is therefore chosen with an additive model over the eight bands of eight layers:

$$\mathrm{KL}(\mathrm{mix}) \approx \mathrm{KL}_0 + \sum_{b=1}^{8} \sigma_b \big(\mathrm{KL}(g_b) - \mathrm{KL}_0\big), \tag{4}$$

where KL_0 is IQ8R's KL, $\mathrm{KL}(g)$ is the KL of a whole trunk on grid g, and σ_b is band b's share, measured by raising one band at a time. Projection types turned out about equally sensitive per weight. The model's predictions landed within 1–7% of the measured KL of the final mixes.

Embedding and head. Each holds 1.27B parameters, so one bit on either costs 0.047 bpw. Narrowing them is expensive at the high sizes: a 4-bit embedding by rounding costs IQ8R +0.00058 KL, and a 6-bit one +0.00003. The embedding is rounded to nearest in every recipe, at 6 bits (asymmetric) in the IQ6R sizes and IQ4R-L, a 5-bit table in IQ4R and IQ5R, 4 bits from IQ3R-S to IQ4R-S, and 3 and 2 bits in IQ2R-L and IQ2R. The head is int8 in IQ6R and IQ6R-L; elsewhere it is a GPTQ-fitted table of 3 to 6 bits.

The grid per rate. The chosen forms (Figure 5) are vector codebooks at 2–3 bits, 16- and 32-level tables at 4 and 5 bits, the asymmetric 6-bit grid, and int8 for the last band where it pays. At two

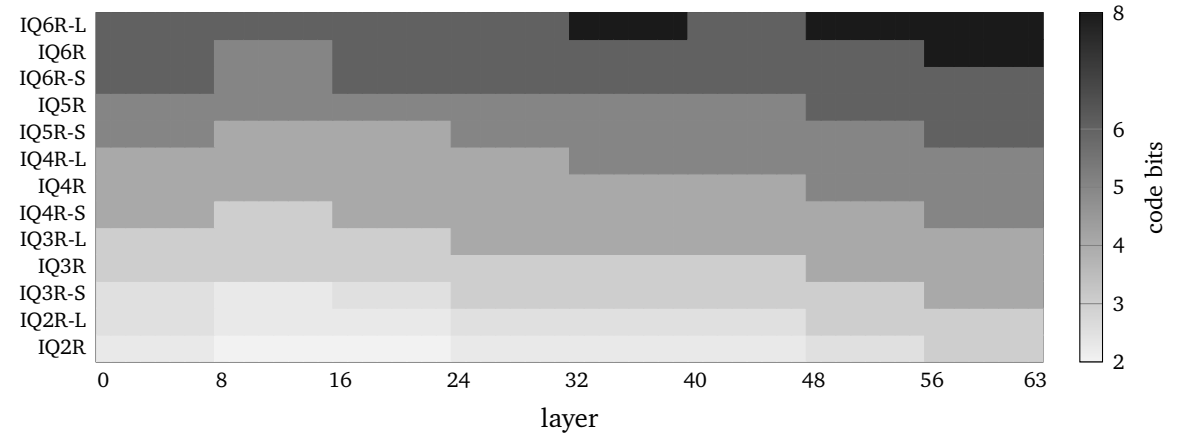

Figure 5. Code bits per layer of each IQR-V1 recipe, weighted by parameter count over the layer's projections. The bits go to the last layers first.

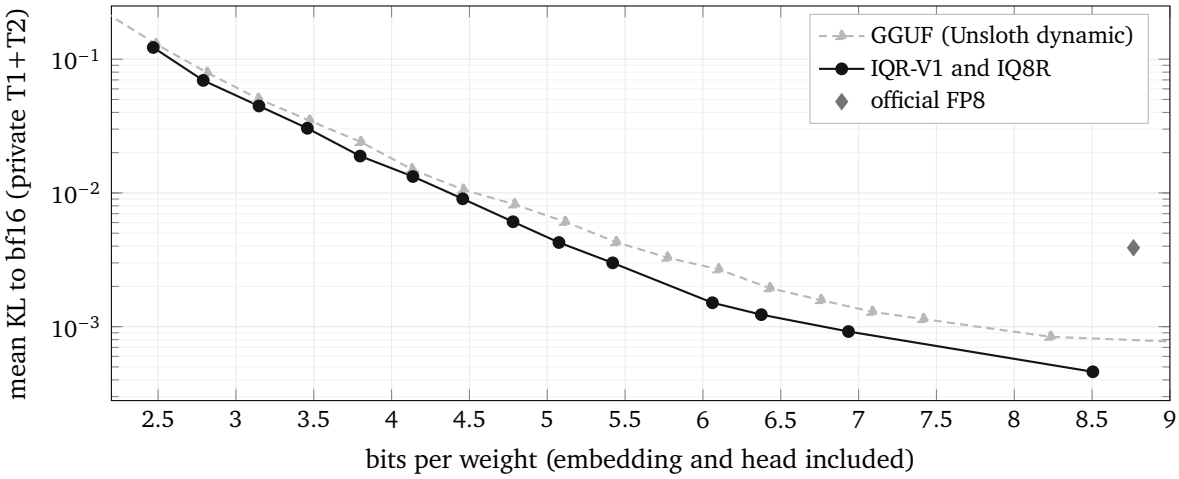

Figure 6. Size against KL on the private tiers. IQR-V1 is below the GGUF curve at every size from 2.5 to 7.4 bpw.

bits a narrower scale group (64 or 32) leaves the error almost unchanged; bits buy more as wider codebooks than as finer scales.

6 Evaluation

Benchmark. Scores are on the private split of a Qwen3.8-27B quantisation benchmark built from three disjoint 10M-token corpora: calibration, public and private. Its leaderboard subset (tiers T1+T2) holds every eighth private document plus two 256K-token agentic traces, about 1.45M scored positions. The reference is the bf16 model with bf16 KV cache. The metric is the mean per-position KL divergence from the reference distribution, computed exactly from the reference's top-512 log-probabilities plus a tail-mass term. All IQR candidates were selected on the public split; the private split was used only for final numbers (one exception is noted in Section 8). Bits per weight are counted over the text model's 26.9B parameters, embedding and head included. For scoring, every code was decoded to int8 once when the model loaded. The decode is exact, so the scores are those of the in-kernel decode. The GGUF references are Unsloth's dynamic quants of the same model, run in llama.cpp on the same token ids.

recipe	bpw	KL	top-1	GGUF at that size	bpw	KL	ΔKL
IQ6R-L	6.935	0.00092	98.91%	UD-Q6_K_L	7.089	0.00129	−29%
IQ6R	6.375	0.00123	98.73%	UD-Q6_K	6.431	0.00194	−37%
IQ6R-S	6.061	0.00151	98.48%	UD-Q5_K_XL	6.102	0.00268	−44%
IQ5R	5.420	0.00300	98.00%	UD-Q5_K_S	5.444	0.00428	−30%
IQ5R-S	5.075	0.00426	97.66%	UD-Q4_K_XL	5.115	0.00608	−30%
IQ4R-L	4.780	0.00609	97.19%	UD-Q4_K_M	4.790	0.00820	−26%
IQ4R	4.457	0.00904	96.56%	UD-Q4_K_S	4.461	0.01057	−14%
IQ4R-S	4.137	0.01327	95.50%	UD-IQ4_XS	4.132	0.01499	−11%
IQ3R-L	3.798	0.01887	94.78%	UD-Q3_K_XL	3.803	0.02403	−21%
IQ3R	3.459	0.03043	93.59%	UD-IQ3_S	3.474	0.03473	−12%
IQ3R-S	3.148	0.04469	92.29%	UD-IQ3_XXS	3.145	0.05052	−12%
IQ2R-L	2.790	0.06937	90.32%	UD-Q2_K_XL	2.816	0.07920	−12%
IQ2R	2.469	0.12234	87.02%	UD-IQ2_S	2.487	0.12976	−6%
IQ8R	8.505	0.00046	99.18%	Q8_0	8.502	0.00071	−35%

Table 2. IQR-V1 on the private T1+T2 tiers against the GGUF of the nearest size. IQ6R is also below UD-Q6_K_L (7.089 bpw, 0.00129), and IQ6R-L below UD-Q6_K_XL (7.417 bpw, 0.00114).

Results. Table 2 and Figure 6 give the final numbers. Every GGUF on the board between 2.5 and 7.4 bpw is beaten at its size, by 6 to 44% in mean KL. The margin is largest at five to six bits and smallest at two. By source, IQ4R's lead over UD-Q4_K_S is widest on agentic traces and reasoning (0.71–0.74 of its KL; 0.00595 against 0.00822 on the 256K traces) and narrowest on chat, encyclopaedic text and long fiction (0.95–0.99).

7 Runtime

An IQR weight stays in its narrow form in VRAM. The GEMM reads the codes, maps them to int8 in registers or local memory (a sign extension, a subtraction or a table lookup), and feeds the same int8×int8 WMMA tile and per-block scale fold for every format. The weights therefore take the format's bits in memory: IQ4R is 15.0 GB against IQ8R's 28.6 GB.

Decode. At batch 1 a GEMV is bound by the weight bytes it reads, so narrow codes pay off directly. In a GEMV core with each form's decoder (Table 3), every V1 form runs 1.2–2.4× faster than int8 on `ffn_gate_up`, from the 6-bit asymmetric grid to the 256-entry vector codebook, and 1.04–2.0× on the smaller `ffn_down`.

Prefill. At prefill size the GEMM is compute-bound, and the decode competes with the matrix instructions for issue slots. In a common reference tile at $M = 2048$, the scalar tables and the 256- and 512-entry vector codebooks cost at most 10% over int8. The 4096- and 1024-entry codebooks cost 21–33%, and the 6-bit asymmetric form 52%. The reference tile reaches about half the throughput of radiance's production int8 tile, so the table gives the decode's relative cost, not absolute speed. The production int8 tile itself is 19% slower than the FP8 one: at a 2048-token prefill, one rank of two, a layer's GEMMs take 3.77 ms against 3.17 ms. The cause is int8's per-block scale fold, not its multiply rate.

μs, one rank	int8	table 4b	table 5b	vq 4×256	vq 4×1024	vq 4×4096	u_6+zero
GEMV batch 1, `ffn_gate_up`	154.7	84.7	110.9	64.5	82.3	100.7	129.0
GEMV batch 1, `ffn_down`	77.3	49.8	65.0	38.5	45.5	54.2	74.1
prefill M=2048, `ffn_gate_up`	3334	3481	3670	3371	4449	4024	5066
relative to int8	1.00	1.04	1.10	1.01	1.33	1.21	1.52

Table 3. GEMM with in-kernel decode of each form, one rank of two. GEMV: a bandwidth-bound decode core, with weights rotated through more than the last-level cache. Prefill: a common reference tile. Over each recipe's whole trunk the reference tile's per-layer mean is 7.6–10.3 ms, against 7.0 ms for int8 and 6.7 ms for FP8. Single runs: under sustained load the card's clocks drift, so differences below about 10% are indicative only.

8 Limitations

- The decode kernels of Section 7 are standalone measurements. Fusing the decoders into the production tiles is the remaining engineering, and the wider codebooks' prefill cost is its main open item.
- IQ2R's margin (6%) is within reach of selection noise. Its final candidate was picked on the private number because the public split ranked two candidates the other way within noise; every other recipe was selected on the public split alone.
- The calibration Hessians come from a single 10M-token corpus, and every recipe was selected on one public split.
- All results are for one model, Qwen3.8-27B (a hybrid of gated delta-net and attention layers).

9 Conclusion

Making every low-bit format an exact encoding of one int8 grid keeps a single integer matrix core, scale fold and activation path for every size. Each format contributes only a decoder, its bits set the memory footprint and the decode bandwidth, and format design becomes a pure rate-distortion problem. Within that constraint, GPTQ with codebook grids, dynamic scales and descent, plus a band-sensitivity allocator, beats the dynamic GGUF quants at every size from 2.5 to 7.4 bpw. IQR-V2 keeps the grid and replaces the codes with a trellis run down each GPTQ column.

References

[1] E. Frantar, S. Ashkboos, T. Hoefler, D. Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *ICLR*, 2023.

[2] J. Chee, Y. Cai, V. Kuleshov, C. De Sa. QuIP: 2-bit quantization of large language models with guarantees. *NeurIPS*, 2023.

[3] S. Ashkboos et al. QuaRot: Outlier-free 4-bit inference in rotated LLMs. *NeurIPS*, 2024.

[4] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* 28(2):129–137, 1982.