IQR-V2: Trellis-Coded GPTQ Columns on a Rotated Int8 Grid

Technical report on IQR-V2 for Qwen3.8-27B

Deadcode
radiance project

October 2026

Abstract

IQR-V2 replaces the scalar and vector codebooks of IQR-V1 with a tail-biting bitshift trellis of 2^{12} states, run down each column of the rotated weight inside GPTQ. Every row of a column shares the column's Hessian diagonal, so the Viterbi search minimises GPTQ's own per-column objective, and GPTQ's error feedback to later columns stays exact. The trellis values are trained integer tables on IQR's int8 grid. Each row's value decodes independently from a window of its column's bit stream, so the GEMM decodes V2 in-kernel to the same int8 operands, matrix core and rotated int8 activations as every IQR size. On one projection, V2 has 23–33% less output error than V1 at every rate from 2 to 6 bits. On Qwen3.8-27B, thirteen V2 recipes from 2.47 to 6.94 bpw reduce mean KL by 21–40% against V1 at equal size, and two more extend the ladder to 8.04 bpw. IQ4R-V2 (4.46 bpw) reaches 0.00562 against 0.00904 for V1, and IQ4R-L-V2 (4.78 bpw) is below the official FP8 checkpoint at 55% of its size.

1 Background

IQR-V1 [1] quantises each linear layer of Qwen3.8-27B on one grid: the weight is rotated by a 128-point Walsh–Hadamard transform R, and every value is an int8 code times a scale per (row, 128 inputs). Narrow forms are exact encodings of that grid. They stay narrow in memory, and the GEMM decodes them in-kernel, so one int8×int8 matrix core with rotated int8 activations serves every size. Codes are fitted by GPTQ [2] against calibration Hessians. V1's best forms were integer Lloyd tables at 4–5 bits and four-wide vector codebooks at 2–3 bits.

Scalar and small vector codebooks leave a known gap to the rate-distortion bound. Trellis-coded quantisation [3] closes most of it by letting a long code sequence choose among many reconstruction values with only k bits per sample. QTIP [4] brought bitshift trellises to LLM weights, combining them with incoherence processing and feeding errors back between blocks of columns. IQR-V2 keeps everything in IQR except the codes, and puts the trellis where GPTQ can account for it exactly: down a column.

2 The trellis down a column

Code. A column j of the rotated weight $\tilde{W} \in \mathbb{R}^{N\times K}$ is coded as a stream of k-bit symbols $c_1, \ldots, c_N$, one per row. An L-bit state is the last L/k symbols,

$$\mathsf{s}_r = \left(\mathsf{s}_{r-1} \cdot 2^k + c_r\right) \bmod 2^L, \qquad \hat{w}_{r,j} = \sigma_{r,g}\, \mathrm{T}[\mathsf{s}_r], \quad g = \lfloor j/128 \rfloor, \tag{1}$$

where T is a table of 2^L integers in $[-127, 127]$ and $\sigma_{r,g}$ is the row's group scale. Each row therefore costs k bits, yet its value is drawn from 2^L levels whose choice depends on the neighbouring rows' symbols (Figure 1). V2 uses $L = 12$ and $k = 2, \ldots, 7$.

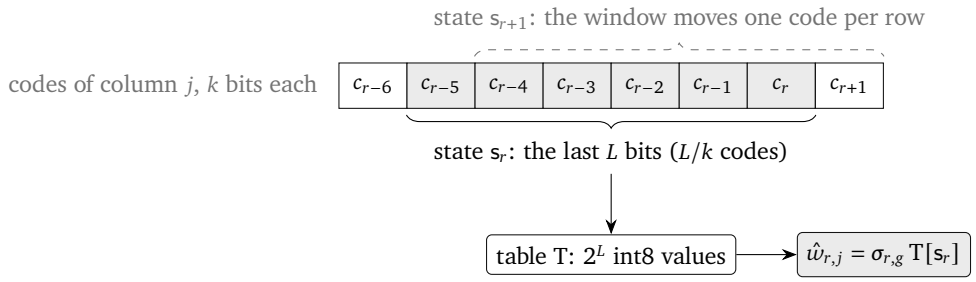

Figure 1. Bitshift trellis. Row r's value is the table entry of the window formed by its own code and the preceding ones. Shown for $k = 2$, $L = 12$.

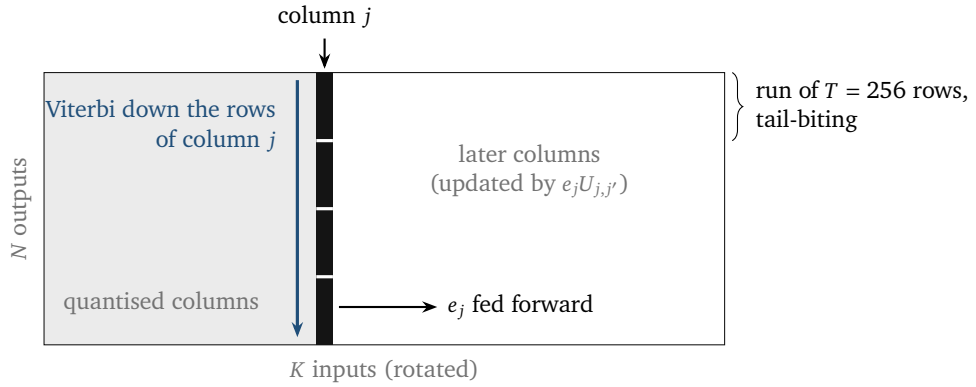

Figure 2. IQR-V2 inside GPTQ. Each column is trellis-coded over its rows in runs of 256, then its error is fed to the later columns exactly as in scalar GPTQ.

Why columns. GPTQ processes the columns of $\tilde{W}$ in order. With U the upper Cholesky factor of H^{-1}, quantising column j to $\hat{w}_j$ contributes

$$\mathcal{L}_j = \frac{\|\tilde{w}_j - \hat{w}_j\|^2}{U_{jj}^2} \tag{2}$$

to the layer objective, after which the error $e_j = (\tilde{w}_j - \hat{w}_j)/U_{jj}$ is fed into the later columns. The weight $1/U_{jj}^2$ is the same for every row of the column. A Viterbi search that minimises plain squared error over a column's rows therefore minimises exactly the quantity GPTQ would minimise for that column. The feedback afterwards is the ordinary GPTQ update, exact to the last column. A trellis along a row would instead need a different metric weight for every symbol and couple many columns into one decision. That is why QTIP feeds errors back only between blocks of columns [4].

Tail-biting runs. A column is coded in independent runs of $T = 256$ rows. Each run is tail-biting: its first state is formed from the run's own last symbols, so no bits are spent on a start state and runs decode independently. The seam state is fixed by an unconstrained Viterbi over a window of $8\lceil L/k \rceil$ rows on each side of the seam, and the run is then searched with its start state tied to its end. Runs align with the tensor-parallel split of the rows.

Scales. Each (row, 128) group has an E4M3 scale under one f32 scale for the tensor. The trellis is searched against the bf16 rounding of their product, the scale the kernel applies. Against V1's bf16 group scales this costs 0.1% more error and saves 0.0625 bpw. Group scales are chosen as GPTQ reaches each group, as in V1.

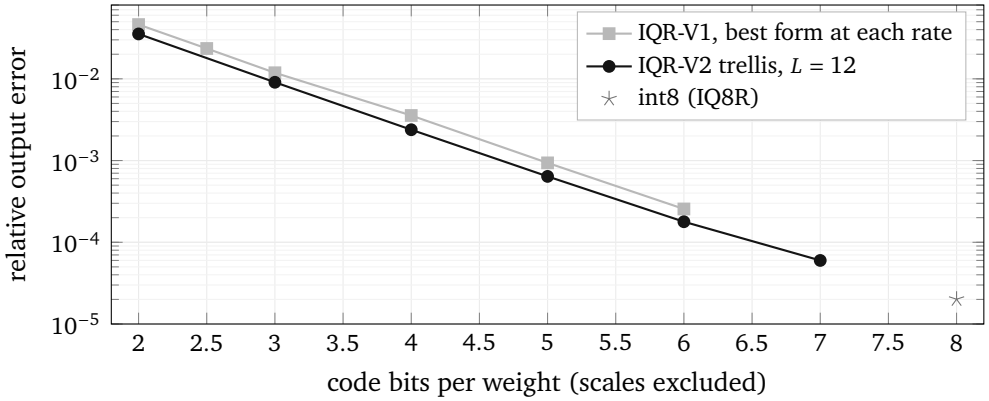

Figure 3. Relative output error $\mathrm{tr}(EHE^\top)/\mathrm{tr}(\tilde{W}H\tilde{W}^\top)$ of layer 3's `ffn_gate_up`. V1's forms are the vector codebooks at 2–3 bits, the Lloyd table at 4–5 and the asymmetric grid at 6.

Tables. The 2^{12}-entry integer tables are trained for each k by Lloyd iterations on Viterbi paths: code a sample of rotated, scale-normalised weights, then move each entry to the mean of the values assigned to it, rounded to an integer. They start from normal quantiles (the outermost at ±127) placed by a seeded Fisher–Yates shuffle. A hash-based placement was worse. Table 1 puts the trellis against the bound on a unit Gaussian at two bits.

quantiser at $R = 2$ bits	MSE	gap to $D(R)$
rate-distortion bound $D(R) = 2^{-2R}$	0.0625	–
trellis, $L = 14$	0.0702	0.50 dB
trellis, $L = 12$ (V2)	0.0733	0.69 dB
Lloyd–Max scalar (V1 style)	0.1175	2.74 dB

Table 1. Unit Gaussian source, two bits per sample.

Cost. A step evaluates 2^k predecessors for each of 2^L states. An AVX-512 implementation, 12× faster than the generic one and producing byte-identical codes, codes the largest projection of a layer (`ffn_gate_up`, 178M weights) in about 10 s. A whole-trunk base convert takes 35–100 minutes on 16 threads.

3 Layer results

Figure 3 compares the two generations on one projection. The trellis has 23% less output error than V1's best form at 2 and 3 bits, and 30–33% less at 4 to 6 bits. Against V1's forms with their tuned coordinate-descent passes the gap is 21–29%. `ffn_down` behaves the same way. A 16384-state trellis ($L = 14$) gains a further 4–9% at four times the search cost and is left to the next generation.

4 Whole-model allocation

Bases. One base convert per k puts the whole trunk on that trellis. On the public split, the 4-bit trellis base scores a KL of 0.00733 against 0.01082 for V1's tuned 4-bit base (−32%), and the 3-bit one 0.02715 against 0.03758 (−28%) (Figure 4).

Allocator. V1's additive band model carries over with a finer split. Each band of eight layers is divided into three units (`ffn_gate_up`, `ffn_down`, and the attention and delta-net projections), 24 in

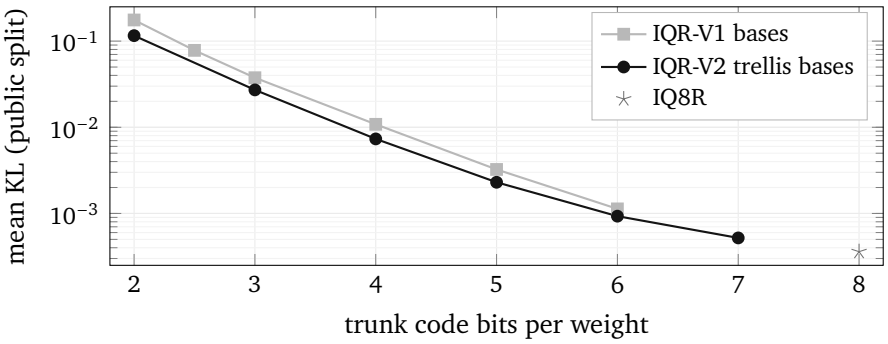

Figure 4. Whole-trunk bases, one grid for every projection of all 64 layers, scored on the public split. These measurements are the inputs to the allocator.

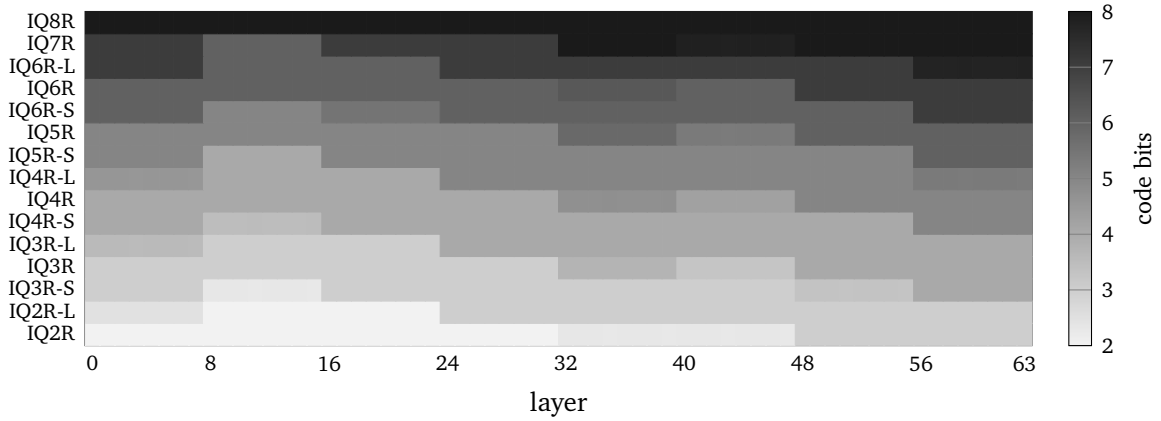

Figure 5. Code bits per layer of each IQR-V2 recipe (suffix -V2 omitted), weighted by parameter count over the layer's projections.

all. Each unit may take any trellis width $k = 2..7$ or int8. A unit's KL share is its band's measured share times its fraction of the band's weights, since the projection types are about equally sensitive per weight. The best mix at each size is found exactly by merging the units' (bytes, KL) Pareto fronts. The chosen mixes (Figure 5) use two or three adjacent widths, with the widest at the last band.

Embedding and head. The embedding is trellis-coded in the rotated basis. A lookup decodes the token's row and undoes the rotation with one inverse transform per row. This costs +0.00006 KL at 4 bits. An unrotated trellis embedding or head clips outliers and costs 40–120× more (+0.0024 for the embedding, +0.0070 for the head), so the head keeps V1's GPTQ-fitted scalar tables, or int8 at the top sizes. IQ6R-L-V2 to IQ8R-V2 keep a 6-bit asymmetric embedding.

5 Evaluation

The benchmark is IQR-V1's [1]: mean KL to the bf16 model (bf16 KV cache) over the private split's leaderboard tiers T1+T2. These are about 1.45M positions, including two 256K-token agentic traces, with KL computed exactly from top-512 reference log-probabilities plus a tail term. Mixes were selected on the public split. Bits per weight count 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.

recipe	bpw	KL	top-1	V1 bpw	V1 KL	ΔKL	trunk widths
IQ8R-V2	8.040	0.00047	99.18%	8.505	0.00046	+2%	int8
IQ7R-V2	7.418	0.00061	99.11%	–	–	–	int8, 7, 6
IQ6R-L-V2	6.937	0.00073	99.05%	6.935	0.00092	−21%	7, 6, int8
IQ6R-V2	6.375	0.00094	98.92%	6.375	0.00123	−24%	6, 7
IQ6R-S-V2	6.062	0.00115	98.78%	6.061	0.00151	−24%	6, 5, 7
IQ5R-V2	5.423	0.00204	98.34%	5.420	0.00300	−32%	5, 6
IQ5R-S-V2	5.076	0.00273	98.07%	5.075	0.00426	−36%	5, 4, 6
IQ4R-L-V2	4.784	0.00377	97.78%	4.780	0.00609	−38%	5, 4, 6
IQ4R-V2	4.457	0.00562	97.29%	4.457	0.00904	−38%	4, 5
IQ4R-S-V2	4.130	0.00832	96.67%	4.137	0.01327	−37%	4, 5, 3
IQ3R-L-V2	3.798	0.01198	96.04%	3.798	0.01887	−37%	4, 3
IQ3R-V2	3.458	0.01880	95.06%	3.459	0.03043	−38%	3, 4
IQ3R-S-V2	3.145	0.02798	93.75%	3.148	0.04469	−37%	3, 4, 2
IQ2R-L-V2	2.792	0.04698	92.03%	2.790	0.06937	−32%	3, 2
IQ2R-V2	2.467	0.07303	90.15%	2.469	0.12234	−40%	2, 3

Table 2. IQR-V2 on the private T1+T2 tiers against the V1 recipe of the same size. Trunk widths are the trellis k values used, most frequent first.

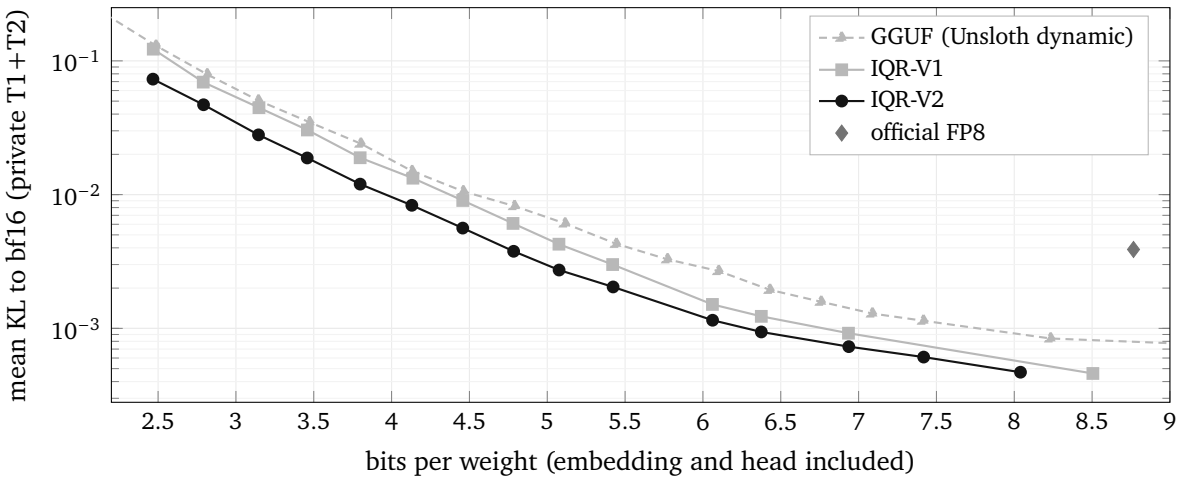

Figure 6. Size against KL on the private tiers. IQR-V2 sits roughly half a bit to the left of IQR-V1 across the range.

Results. Table 2 and Figure 6 give the final numbers. From 2.5 to 6.9 bpw V2 has 21–40% less KL than V1 at the same size, and the margin is largest between 3 and 5 bits. Put another way, each V2 recipe roughly matches the V1 recipe half a bit larger. IQ3R-V2 at 3.46 bpw (0.01880) matches IQ3R-L at 3.80; IQ4R-V2 (0.00562) beats IQ4R-L at 4.78. Against outside references:

- IQ4R-L-V2 (4.784 bpw, 0.00377) is below the official FP8 checkpoint (8.766 bpw, 0.00389) at 55% of its size;
- IQ7R-V2 (7.418 bpw, 0.00061) has 46% less KL than UD-Q6_K_XL at the same size (0.00114);
- IQ2R-V2 (2.467 bpw, 0.07303) is below UD-Q2_K_XL (2.816 bpw, 0.07920), a third of a bit larger.

IQ8R-V2 is IQ8R with a 6-bit embedding and holds IQ8R's KL in 5% fewer bits.

6 Runtime

A V2 weight stays in VRAM as its trellis bit stream, with the scales and the 2^{12}-entry table, so the weights take k bits each in memory, as in the file. The GEMM decodes them on the way to the matrix units.

Decode. Stored in reverse, each column's code stream makes row r's state a contiguous L-bit window of the bit stream, plus a wrapped tail at the start of each tail-biting run. Every row's value is then decoded independently with one shift, one mask and one table lookup, with no sequential dependency. A standalone GEMV built on this window form has each lane own eight columns, uses packed signed dot products and reduces across sixteen lanes in four DPP steps. Weights rotate through more than 512 MB so that the 64 MB last-level cache cannot hold them. At batch 1 it beats the int8 GEMV on the large projections up to $k = 4$–5 (19% on `ffn_gate_up` at $k = 2$), but is 5–39% slower on the smallest (Table 3). At batch 4 it takes 1.2–1.9× the int8 time. The table lookups, not the bytes, set its speed: V1's scalar tables decode faster at the same width.

shape (one rank)	int8	fp8	$k = 2$	3	4	5	6	7
`ffn_gate_up`	149.5	158.3	121.3	122.7	124.5	129.3	149.2	173.9
`ffn_down`	74.8	84.7	69.6	70.4	73.0	74.8	78.3	85.6
`ssm_inz`	71.3	79.4	68.3	66.7	68.3	71.6	74.4	82.5
`ssm_out`	30.8	34.0	32.4	33.1	36.4	38.2	40.5	42.9

Table 3. Decode GEMV at batch 1, μs, on one R9700 (one rank of two): the production int8 and FP8 kernels against the native trellis kernel at each k. Single runs: under sustained load the card's clocks drift, so differences below about 10% are indicative only.

Prefill. At prefill sizes the decode is no longer free. In a common reference GEMM core with in-kernel decode at $M = 2048$, the trellis forms take about 1.55× the int8 time. A V2 recipe's trunk averages 9.2–11.1 ms per layer there, against 7.0 ms for int8 and 6.7 ms for FP8. A two-pass alternative, which decodes a weight slice into the production int8 tile layout and then runs the production GEMM, costs 8–23% over int8 at $M = 2048$. Fusing that decode into a persistent production GEMM, which decodes its next slice while it multiplies the current one, is the open engineering step.

7 Limitations and outlook

- The decode kernels are standalone measurements, not yet fused into the production tiles. At batch above one and at prefill size, the trellis decode costs more than V1's table decodes.
- In production the int8 prefill GEMM is about 19% slower than FP8. The cause is its per-block scale fold: int8 and FP8 matrix instructions run at the same rate.
- Above 7 bits V2 has no trellis advantage: IQ8R-V2's trunk is int8, and its saving comes from the embedding alone.
- The trellis search is CPU-bound. A 16384-state table would help most at 4–6 bits.

The next generation targets both prefill limits without giving up V2's quality: trellis values on an int8 grid with one scale per block of rows, so that the integer GEMM folds scales as cheaply as FP8, and 16384-state tables. That work is in progress.

References

[1] Deadcode. IQR: GPTQ integer quantisation on a rotated int8 grid. Technical report on IQR-V1, radiance project, 2026.

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

[3] M. W. Marcellin, T. R. Fischer. Trellis coded quantization of memoryless and Gauss–Markov sources. *IEEE Trans. Commun.* 38(1):82–93, 1990.

[4] A. Tseng, Q. Sun, D. Hou, C. De Sa. QTIP: Quantization with trellises and incoherence processing. *NeurIPS*, 2024.