ESPTQ: POST-TRAINING TERNARIZATION OF WEIGHTS AND KV CACHE FOR ELIMINATING PER-ELEMENT MULTIPLICATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Reading the weights and KV cache dominates large language model inference. Ternary quantization to $\{-1, 0, +1\}$ shrinks both and removes per-element multipliers: each weight product becomes a sign flip or skip, so a signed adder tree suffices. Ternary weights alone, however, leave multipliers in attention's $QK^\top$ and PV, whose operands are both run-time floating-point tensors. We present ESPTQ (element-sparse-plane ternary post-training quantization), which quantizes the weights and KV cache into one ternary storage format, removing per-element multipliers from every matrix product inside the Transformer layers except the rotation of layer inputs. By design, ESPTQ caps the residual multiplication ratio at the weight-side budget of two multiplications per 128-element reduction ($2/128 = 1.5625\%$). For weights, after a learnable butterfly rotation within each 128-column group, a rate-distortion criterion selects, at a continuously variable second-plane assignment ratio ρ, which elements receive a sparse second ternary plane. For the KV cache, static tables replace per-token offsets and the V-side scaling factor, meeting the same budget. We compare methods on effective bits per weight, serializing each in its own storage format with losslessly compressed discrete codes, so scaling factors, rotation angles and other auxiliary metadata count. On LLaMA-2-7B, ESPTQ matches TWLA in perplexity and average zero-shot accuracy at less than half the effective bits, and at comparable bits lowers perplexity by 36% against PT2-LLM. Ternarizing the KV cache as well shrinks it to about one fifth of its FP16 size, at less cost in perplexity and zero-shot accuracy than ternarizing the weights.

1 INTRODUCTION

Large language model (LLM) inference spends most of its time and energy reading the weights and KV cache and multiplying the values read. Quantization shrinks those reads on the weight (Gholami et al., 2022) and KV (Liu et al., 2024; Hooper et al., 2024) sides, but the values read are still multiplied. We present ESPTQ (element-sparse-plane ternary post-training quantization) to address both.

Post-training quantization (PTQ) such as GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024) and QuaRot (Ashkboos et al., 2024) nears FP16 accuracy at 4 bits. At 2 bits, GPTQ and QuIP (Chee et al., 2023) reach WikiText-2 perplexities of 29.23 and 82.92 on LLaMA-2-7B (Touvron et al., 2023) (Table 1), against FP16's 5.47. With ternary weights $\{-1, 0, +1\}$ the product reduces to a sign flip or skip (Ma et al., 2024), so a linear layer needs no per-element multiplier, as fused ternary CPU kernels (Zuo et al., 2026) confirm. We call this property **multiplication-free**. INT-k codes split into bit planes are also multiplication-free, but two signed adder trees give nine unevenly spaced levels with ternary planes and only four with INT2. Attention's $QK^\top$ and PV, with P the softmax output, still multiply two run-time operands.

Prior ternary PTQ declares one plane per weight, as in PT2-LLM (Yan et al., 2026), CAT-Q (Wang et al., 2026) and TWLA (Zhao et al., 2026), or stores two dense planes, as in PTQTP (Xiao et al., 2025). To our knowledge, none shows, on effective bits (defined below), the gain from a sparse, element-wise second plane correcting the first while staying multiplication-free. Prior KV-cache work, including ternary and signed-digit caches, removes multiplications from attention (Kang

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

et al., 2023; Duan et al., 2026; Zhang et al., 2024; Ahmed et al., 2026). To our knowledge, none gives the KV cache the weight-side storage format — two ternary planes and symmetric INT8 scaling factors — and stays within the weight-side residual multiplication ratio, cache writes included. The residual multiplication ratio counts multiplications by fully general multipliers, those with two run-time operands, per multiply-accumulate of a reduction, the sum over the shared index of a matrix product (§3.1). Our design caps it at the weight-side budget of $2/128 = 1.5625\%$, two multiplications per 128-element reduction.

First, the weights' second plane is **element-wise sparse**, assigned by a rate-distortion criterion. Second, we learn *blockfly*, a butterfly rotation within each group of 128 contiguous input channels, mainly to make the rotated weights fit the trimodal Gaussian mixture on the ternary grid that TWLA (Zhao et al., 2026) targets. ButterflyQuant (Xu et al., 2025) introduced learned butterfly rotations as a quantization preprocessor, trained for layer-output reconstruction under uniform quantization. Third, we ternarize the KV cache in the weights' storage format, making every matrix product inside the Transformer layers (Vaswani et al., 2017) multiplication-free except the rotation of each linear layer's input. With FP16 activations, we write this configuration as W1.58A16KV1.58, where 1.58 ($\log_2 3$) names the ternary alphabet, not the stored bits.

We compare on **effective bits per weight**. A single harness serializes each method in its own storage format, losslessly compressing only its discrete codes with zlib (Deutsch & Gailly, 1996). We divide the total size by the number of quantized weights, so scaling factors, offsets, rotation angles, masks and indices all count. TWLA declares 1.58 bits and reports about 1.6, both counting one ternary value per weight, yet its public code stores 5.416 bits on average (§5.1).

2 Related Work

ESPTQ needs only a small calibration set of text (Frantar et al., 2023), not the retraining of quantization-aware training (QAT; Chen et al., 2025). Uniform-grid methods treat outliers by high-precision retention (Dettmers et al., 2022), per-channel scaling (Lin et al., 2024; Xiao et al., 2023), learnable clipping (Shao et al., 2024) or rotation (Chee et al., 2023). We rotate: for activation x, weights W and orthogonal transform R, we store quantized WR and form xR at run time. Proposals for R range from randomized Hadamard transforms (Tseng et al., 2024; Ashkboos et al., 2024) to learned ones (Liu et al., 2025a; Xu et al., 2025). Our *blockfly* targets the trimodal Gaussian mixture of TWLA's Kronecker orthogonal tri-modal shaping (KOTMS; Zhao et al., 2026). PeRQ (Sanjeet et al., 2026) permutes channels before a block-confined Hadamard rotation and analyzes its limit on activation-outlier suppression in the deterministic worst case.

ABC-Net (Lin et al., 2017) stacks binary planes; BiLLM and ARB-LLM add a second binary plane on salient columns (Huang et al., 2024; Li et al., 2025), and SliM-LLM allocates bits per group (Huang et al., 2025b). PB-LLM (Shang et al., 2024), SpQR (Dettmers et al., 2024) and SqueezeLLM (Kim et al., 2024) keep salient or outlier weights sparsely in higher precision beside a dense low-bit grid. We instead place a second ternary plane sparsely, by element, without storing positions separately (§3.1). Appendix A details these lineages, ternarization's precursors (Li et al., 2016; Ma et al., 2024), our four ternary PTQ baselines and contemporaneous ternary quantization. TWLA is, to our knowledge, the only prior ternary-weight PTQ work with 4-bit activations.

KIVI (Liu et al., 2024), KVQuant (Hooper et al., 2024) and QuaRot quantize K and V to 2–4 bits. Their per-channel K (KIVI, KVQuant) or per-token V scaling factors vary along the reduction dimension (§3.1) of $QK^\top$ or PV, respectively, leaving multiplications inside the reduction or pre-multiplied into its left operand (§4.1, §4.2). The signed-digit K/V cache (Duan et al., 2026), to our knowledge the closest to ours, also runs attention on the weights' datapath but uses lookup tables, power-of-two ratios between planes, V scales per token interval and a BF16 V tail.

3 Ternary Quantization of the Weights

The core of our weight quantization ESPTQ-w is placing a second ternary plane sparsely per element by a rate-distortion criterion. At equal effective bits, this consistently gives lower perplexity than group placement (Figure 2(b); §5.1). ESPTQ-kv quantizes the KV cache; ESPTQ-wkv applies both. Subscripts give ρ, the second-plane assignment ratio (§3.3).

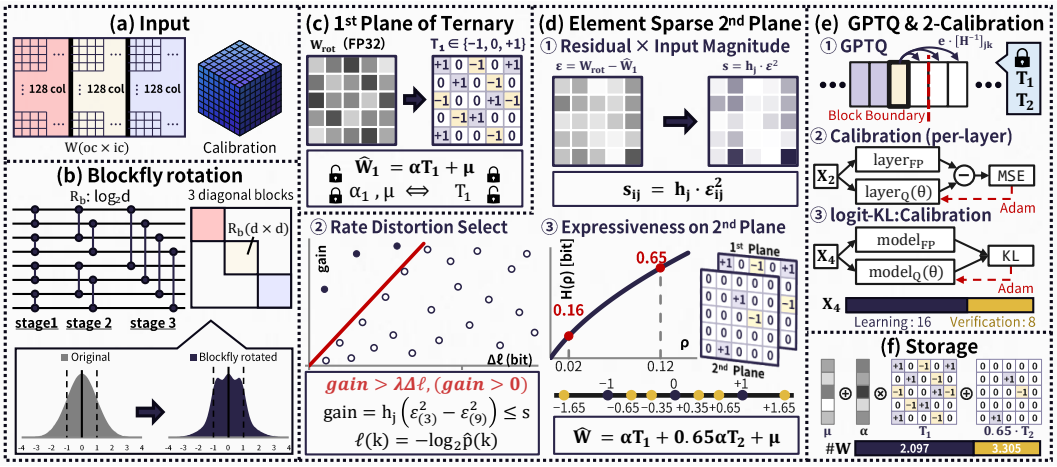


Figure 1: ESPTQ-w weight quantization. (b) R_b, one diagonal block of R, is a $d \times d$ butterfly of $\log_2 d$ stages with $d = G$. (d) $H(\rho)$ is the entropy $\mathcal{H}(\rho)$ of T_2. (e) 'Calibration (per-layer)' is block-wise calibration on X_2, the outputs of the quantized preceding layers; θ denotes ϕ_α, ϕ_μ. End-to-end logit distillation takes X_4: 16 training and 8 validation windows.

3.1 Storage Format and Residual Multiplication Ratio

A linear layer with i_c input and o_c output channels computes $y = xW^\top$ with an $o_c \times i_c$ weight matrix W. The reduction dimension is the one over which the summation index k of $C_{ij} = \sum_k A_{ik} B_{kj}$ runs, the input-channel axis in a linear layer's general matrix multiplication (GEMM); the circuit for the sum is the reduction datapath. We count only fully general multipliers, those with two run-time operands; a product with a design-time constant reduces to shifts and adds. Our circuit model assumes one datapath per linear layer, or on the KV side per layer and KV head. A value fixed before deployment for everything that datapath processes is a design-time constant. A value varying across the processed rows, groups, channels or tokens, such as a per-group scaling factor or rotation angle, is a run-time operand. The residual multiplication ratio counts multiplications by fully general multipliers in the reduction datapath per multiply-accumulate of the reduction, 100% in a naive floating-point implementation. The input-side rotation lies outside the reduction datapath but is not negligible; §4.4 counts its multiplications separately. Splitting each row into groups of $G = 128$ contiguous input columns, we store per row i and group g a scaling factor α_{ig} and offset μ_{ig}. Two ternary planes $T_1, T_2 \in \{-1, 0, +1\}$ and an inter-plane ratio $r = 0.65$ fixed at design time define the grid values $\Lambda = T_1 + rT_2$; the quantized weight and group-g partial sum are

$$\hat{W}_{ij} = \alpha_{ig}\Lambda_{ij} + \mu_{ig}, \qquad y_{ig} = \alpha_{ig} \sum_{j \in g} \Lambda_{ij} x_j + \mu_{ig} \sum_{j \in g} x_j. \tag{1}$$

Neither sum needs a fully general multiplier, so the reduction datapath multiplies only by α_{ig} and μ_{ig}, a ratio of $2/128 = 1.5625\%$, as in PT2-LLM. Λ takes nine unevenly spaced levels $\{0, \pm 0.35, \pm 0.65, \pm 1, \pm 1.65\}$ (Figure 1(d)③); we set r just below 0.658, the squared-error-optimal ratio for Gaussian weights on this grid; using 0.65 instead raises that error by 0.2%. Tied Trit-Planes (Grella, 2026) also ties two ternary planes by the corresponding scale ratio but fixes it at 3 ($r = 1/3$ in our notation), giving a uniform grid.

What we store. We store, per element, a nine-level joint symbol (T_1, T_2); per row and group, symmetric INT8 α and μ; per linear layer, FP16 reference values $\sigma_\alpha, \sigma_\mu$; and per group, rotation angles (Figure 1(f)). Elements lacking a second plane take $T_2 = 0$, so we store no position mask M. A reference value is the real value of one INT8 step: $\sigma_\alpha = \max_{i,g} |\alpha_{ig}|/127$, initially on the layer's prepass grid (§3.3), and σ_μ likewise, so $\alpha_{ig} = \sigma_\alpha \hat{\alpha}_{ig}$ and $\mu_{ig} = \sigma_\mu \hat{\mu}_{ig}$ with INT8 codes $\hat{\alpha}_{ig}, \hat{\mu}_{ig} \in \{-127, \ldots, 127\}$. Nominal #W is what a method declares for its storage format (1.58 for a ternary plane) and counts different things per method; effective #W* is measured as in §1, excluding the embedding and output layers from its denominator. For ESPTQ, zlib compresses each linear layer's symbols and INT8 codes as one stream, and the reference values and rotation angles are added uncompressed. Appendix B derives 0.658 and gives the byte layout, the baselines' writing rules and exceptions, the reference values and their correspondence with double quantization (Dettmers et al., 2023).

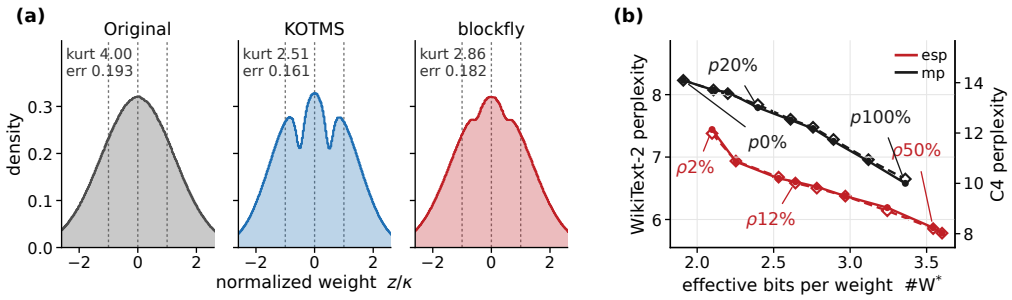


Figure 2: (a) Distribution of z/κ (layer 24, Q projection) for Original (unrotated), KOTMS computed with TWLA's public code on the weights alone, and *blockfly*; kurt: mean row kurtosis (Gaussian: 3); err: relative squared error of per-row asymmetric single-plane ternarization. (b) Second-plane granularity: element (esp: ESPTQ-w) vs. group (mp: gsp-p of §3.3); perplexity on WikiText-2 (left axis; circles, solid) and C4 (right axis; diamonds, dashed).

3.2 The *blockfly* Rotation

One scaling factor per row and group sets a ternary plane's three levels, so a group mixing large and small values fits only one of them well. The orthogonal transform R narrows this spread. In each group, R is a separate seven-stage butterfly (Dao et al., 2019) of Givens rotations, each rotating two coordinates by an angle. Starting at $\pi/4$, the angles are learned once per linear layer (Appendix C) so that, for each full row w of W, $z = wR - \mathrm{mean}(wR)$ clusters at $-\kappa, 0, +\kappa$, with $\kappa = \mathrm{mean}\,|z|$. As in KOTMS (Zhao et al., 2026), the objective is the negative log-likelihood of an equal-weight trimodal Gaussian mixture at these points, plus an occupancy-balancing term pulling each point's element-averaged posterior toward $1/3$. Anticipating 4-bit activations, all our runs add an activation-flattening term that aims to even out the rotated channels' activation energy, computed from GPTQ's Hessian $H = X^\top X$ of calibration inputs X.

We avoid a butterfly across the whole hidden dimension (Xu et al., 2025) to keep data movement local and angle storage small. Outliers stay in their group, and the rotated distribution remains unimodal, not trimodal, with weak shoulders near $\pm 0.6\kappa$ (Figure 2(a)). Yet on LLaMA-2-7B, identity-initialized *blockfly* without the activation-flattening term matches KOTMS in perplexity at almost equal effective bits. It is also lighter there than the released KOTMS, with far less storage than its Kronecker factors and far fewer multiplications per input element. Linear layers whose input width is not a multiple of 128 stay unrotated.

3.3 Prepass — Provisional First Plane and Second-Plane Assignment

On the rotated W and H, we provisionally fit the first plane with PT2-LLM's asymmetric ternary quantizer (ATQ; Yan et al., 2026) and choose second-plane positions from the residual $\varepsilon = W - (\alpha T_1 + \mu)$. ATQ's activation-aware stage weights the error by the group-g submatrix H_g; we keep only its diagonal $h_j = H_{jj}$, the activation energy of input channel j. Besides the salience $s_{ij} = h_j \varepsilon_{ij}^2$, each element's contribution to the layer output error (Figure 1(d)①), we also weigh $\Delta\ell_{ij}$, the increase in ideal code length under the joint-symbol distribution $\hat{p}$ from a second plane on (i, j). With $\varepsilon^{(3)}, \varepsilon^{(9)}$ the errors at the nearest three- and nine-level grid points, the rate-distortion criterion is

$$\mathrm{gain}_{ij} = h_j\Big(\big(\varepsilon_{ij}^{(3)}\big)^2 - \big(\varepsilon_{ij}^{(9)}\big)^2\Big), \qquad M_{ij} = 1 \iff \mathrm{gain}_{ij} > 0 \ \wedge\ \mathrm{gain}_{ij} > \lambda\,\Delta\ell_{ij}. \tag{2}$$

The Lagrange multiplier λ, the slope in Figure 1(d)②, prices one bit of code length in units of distortion, as in Hessian-weighted entropy-constrained scalar quantization (Choi et al., 2017). We set λ per linear layer so that the layer's second-plane assignment ratio, the fraction of its weights with $M_{ij} = 1$, is ρ; operating points differ only in ρ. Starting from a fixed prior, $\hat{p}$ is re-estimated from the layer's first-pass weight symbols for a second pass of both selection and the main loop (§3.4). Appendix D details ATQ, $\hat{p}$ and λ.

#W* follows ρ continuously via $\mathcal{H}(\rho)$, the entropy of T_2 (Figure 1(d)③). The group-granularity control gsp-p (group-sparse plane) puts a second plane on the higher-salience half of the columns in the top fraction p of each linear layer's 128-column groups by total salience. These groups store a per-row INT8 second-plane scale α_2 independent of $r\alpha$ but, like ours, no position mask.

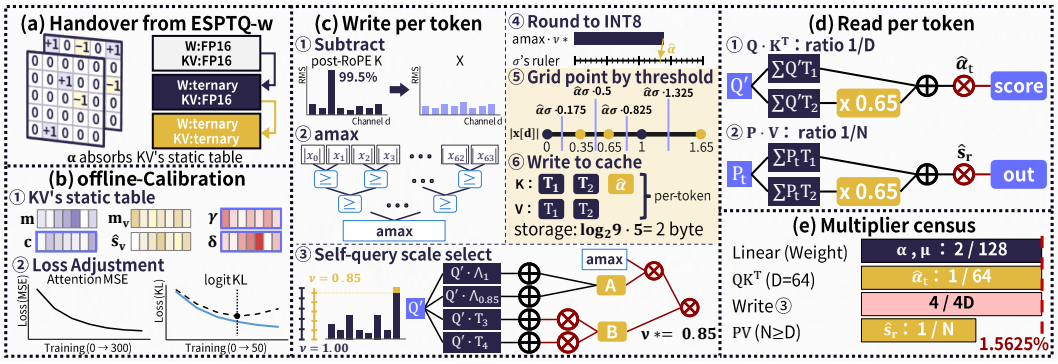


Figure 3: ESPTQ-kv KV-cache quantization. (c) is the write path and (d) the read path. The storage label in (c)⑥ means that five nine-level joint symbols, $5 \log_2 9 \approx 15.85$ bits, fit in one 2-byte word.

3.4 Quantization and Calibration

GPTQ (Frantar et al., 2023) minimizes each layer's output error, not the whole-model degradation. None of the six approaches we tried for minimizing the layer error more precisely improved perplexity consistently (Appendix I). We therefore decide the ternary assignment from the layer error, freeze it and calibrate only α and μ over the whole model. The main loop quantizes one group at a time, compensating its error with unmodified GPTQ over all remaining columns. We re-solve group g's grid on the compensated W_g: ATQ on W_g and H_g gives (α_{ig}, μ_{ig}), which we project to symmetric INT8 with the prepass reference values. On that grid, the nearest point determines (T_1, T_2): three levels where $M_{ij} = 0$, nine where $M_{ij} = 1$ (Figure 1(e)①).

Calibration. We learn, per row and group, only a scaling-factor multiplier Δ_α and an offset correction Δ_μ in $\hat{W}(\phi) = (1 + \Delta_\alpha)\,\alpha\Lambda + \mu + \Delta_\mu$, where $\Delta_\alpha = \frac{1}{2}\tanh(\phi_\alpha)$ and $\Delta_\mu = \frac{1}{2}\tanh(\phi_\mu)\,|\alpha|$. This bounded form follows OmniQuant (Shao et al., 2024), and ϕ_α, ϕ_μ start at 0, the identity. The ternary planes and mask stay frozen, so re-projecting to INT8 with reference values recomputed after each stage keeps the storage format. Calibration has two stages (Appendix E). Block-wise calibration minimizes each decoder layer's mean squared error to its FP16 counterpart on the same input (Figure 1(e)②). Once all layers are quantized, end-to-end logit distillation minimizes the Kullback–Leibler divergence (KL) between the FP16 teacher's and the quantized model's next-token distributions (Figure 1(e)③). It corresponds to the end-to-end fine-tuning of QuIP# (Tseng et al., 2024) and AQLM (Egiazarian et al., 2024). Block-wise calibration takes 256 steps per layer and logit distillation 128 steps on 512-token segments; each stage is reverted unless it lowers the monitored error.

4 Ternary Quantization of the KV Cache

We ternarize the KV cache in the weight-side storage format (Appendix F). The per-element products in $QK^\top$ and PV then reduce to sign flips and skips, so Q and P can stay in high precision.

4.1 Assumptions and Storage Format

Here the denominator of the residual multiplication ratio becomes the reduction length: the head dimension D for $QK^\top$, and for PV the number of tokens N the query reads. A per-token scaling factor and offset would need two multipliers, a ratio of $2/64 = 3.125\%$ at $D = 64$, twice the budget, so our design fits it for $D, N \geq 64$. Generation writes K_t and V_t one token at a time: sharing scaling factors across tokens would hold unquantized K in FP16 until those tokens arrive, and a written K_t cannot be requantized. Quantization may thus use only the token's own K_t, V_t, Q_t, plus the reference value σ and static tables, fixed offline and independent of the token. Their offline calibration adds nothing to the numerator, unlike run-time multiplication by a per-channel table.

The budget condition: the residual multiplication ratio is at most 1.5625%. **The storage-format condition**: the two ternary planes, inter-plane ratio and symmetric INT8 scaling factors with reference values match the weight side. Only the offset differs: static per-channel FP16 tables replace INT8 μ per row and group. **The accumulator-output condition**: multipliers sit at the accumulator

output only, neither inside the reduction nor pre-multiplied into the left operand. **The write-side condition**: the write path, which quantizes each new token into the cache, also uses only reductions with a ternary operand (ternary reductions) and multiplications at the accumulator output.

Offline calibration and the static tables. Calibration fixes six static quantities. Four hold one value per KV head and channel. On the K side are the static offset m, initially the per-channel mean of K after rotary position embedding (RoPE; Su et al., 2024), and the static channel shape c, which evens out channel magnitudes. On the V side are the static offset m_v, initially the per-channel mean of V, and the write-side scaling factor $\hat{s}_v$. The gain γ and correction bias δ of the read path, which computes attention from the cache, hold one value per query head and channel. With the ternary weights frozen (Figure 3(a)), we calibrate the KV side in two stages as in §3.4, with a different objective and teacher. Block-wise calibration updates $m, c, m_v, \hat{s}_v$ one decoder layer at a time, minimizing the mean squared error to its attention output with unquantized K and V. End-to-end logit distillation then updates all six in every layer at once; its teacher is the model with ternary weights and an FP16 KV cache, not the FP16 model (Figure 3(b)). Finally, absorbing c into the key and query projection weights W_k and W_q, already ternary, moves each row's α and μ off the INT8 grid, so we re-round them. The ternary planes, storage format and multiplier count stay unchanged.

The grid and its thresholds. The KV grid is the weight side's $\Lambda = T_1 + rT_2$ ($r = 0.65$). The quantization thresholds $\tau = \{0.175, 0.5, 0.825, 1.325\}$ are the four midpoints between adjacent non-negative levels, the sign handled separately. K has one per-token scaling factor $\alpha_t = \hat{\alpha}_t\sigma$, with $\hat{\alpha}_t$ an INT8 code and σ fixed per layer and KV head. The V-side tables $\hat{s}_v$ and $\hat{s}_r = \hat{s}_v\gamma$, stored in symmetric INT8, enter Equations 3 and 4 scaled by their per-layer reference values. Every KV element takes the nearest of the nine levels, without an element-sparse second plane, which would save no storage in a fixed-length cache. With $\odot$ the per-channel product, the reconstructions are

$$\hat{K}_t = m + c \odot \left(\hat{\alpha}_t\, \sigma\, \Lambda_t^k\right), \qquad \hat{V}_t = m_v + \hat{s}_v \odot \Lambda_t^v. \tag{3}$$

Prefill, which processes the prompt in one pass, writes each token likewise, so, in our ESPTQ-wkv perplexity and zero-shot evaluations, attention reads quantized K and V, a token's own included.

Channel-wise skew of K. K's magnitude spans orders of magnitude across head-dimension channels. In Qwen2.5-0.5B, much of that is a token-independent constant component, shared by every token and, in some layers, carrying most of the energy of K across the whole head. Subtracting m removes it; where magnitudes still differ, a few channels dominate the grid width and the rest collapse into the lowest step, so we divide by c. Since the score $\sum_d Q_d K_d$ equals $\sum_d (Q_d c_d)\,(K_d/c_d)$, absorbing $1/c$ into the rows of W_k and c into those of W_q leaves c no table of its own, and the write path subtracts m/c. As c is absorbed before RoPE, calibration keeps $\mathrm{diag}(c)$ commuting with it ($c_d = c_{d+D/2}$), as in QServe's SmoothAttention (Lin et al., 2025).

4.2 The Read Path

With $Q' = Q \odot c$ the query with c absorbed, and O one query head's attention output before the output projection W_o, the score and O split as

$$Q \cdot \hat{K}_t = \alpha_t \sum_d Q'_d\, \Lambda^k_{t,d} + Q \cdot m, \qquad O = \hat{s}_r \odot \sum_t P_t \Lambda_t^v + \sum_t P_t m_v + \delta. \tag{4}$$

In Equation 4, the score's second term is a constant common to every key in a query row, so it cancels in the softmax and we do not compute it. In the first term, $Q'_d T_{p,t,d} \in \{\pm Q'_d, 0\}$ for each plane $p \in \{1, 2\}$, so two signed adder trees suffice; multiplying by the constant inter-plane ratio r is shift-and-add; and σ can be absorbed into W_q. The only multiplication left at the accumulator output, by $\hat{\alpha}_t$, gives a residual multiplication ratio of $1/D$ (Figure 3(d)①). On the V side, each row of P sums to exactly 1, so $\sum_t P_t m_v$ reduces to m_v and, with δ, can be absorbed into the bias of W_o. The remaining ternary reduction also takes only one multiplication at the accumulator output, by $\hat{s}_r$, which already carries γ. Its ratio $1/N$ approaches 0 as the context grows (Figure 3(d)②).

4.3 The Write Path

The K path has six stages, numbered as in Figure 3(c): ① subtract the static table m/c to form $x = (K_t - m)/c$; ② take the largest absolute value $\mathrm{amax} = \max_d |x[d]|$ over that token's D channels with a comparator tree; ③ choose between the design-time candidates $\nu \in \{1.00, 0.85\}$;

④ round the grid step to the symmetric INT8 $\hat{\alpha}$; ⑤ select the grid point; ⑥ write the ternary planes T_1, T_2 and $\hat{\alpha}$ into the cache. The V path, with no per-token quantity, keeps only ①, ⑤ and ⑥.

③ **Self-query scale selection.** We choose between candidates because tightening the grid uniformly can degrade accuracy sharply. Prior work on clipping shares this motivation (Lin et al., 2024; Shao et al., 2024). Comparing the reconstruction directly against the floating-point x would leave a multiply-accumulate, so we replace the reference with the ternary $y_{\mathrm{ref}} = s_1\Lambda_1 + 0.675\,\beta_1 T_3 + 0.675\,\beta_2 T_4$. Here Λ_ν is the grid point under candidate ν, $s_\nu = \nu\,\mathrm{amax}/1.65$ its step and $y_\nu = s_\nu\Lambda_\nu$ its reconstruction. T_3 and T_4 are uncached ternary residual planes thresholded at 0.35β, and β_1, β_2 are the largest absolute residuals of the two passes. The factor $0.675 = (0.35 + 1)/2$, the midpoint between that threshold and β, is not the inter-plane ratio r. We choose by $J(\nu) = \sum_\eta \big(Q'[\eta] \cdot (y_\nu - y_{\mathrm{ref}})\big)^2$, evaluated with the token's own query $Q' = Q_t \odot c$, where η runs over the query heads sharing that KV head (Ainslie et al., 2023). Stage ③ selects $\nu^* = 0.85$ when $\sum_\eta A_\eta\big(\mathrm{amax}\,A_\eta - 2B_\eta\big) < 0$ and $\nu^* = 1$ otherwise, with $A_\eta = \big(0.85\,Q'_\eta \cdot \Lambda_{0.85} - Q'_\eta \cdot \Lambda_1\big)/1.65$, $B_\eta = 0.675\big(\beta_1\,Q'_\eta \cdot T_3 + \beta_2\,Q'_\eta \cdot T_4\big)$ and $Q'_\eta = Q'[\eta]$. That sum equals $\big(J(0.85) - J(1)\big)/\mathrm{amax}$, so the condition is equivalent to $J(0.85) < J(1)$. Four products of two run-time values remain for the four ternary reductions of length D in A_η and B_η (Figure 3(c)③), so the ratio stays at $1/D$.

④ **and ⑤: no divider on the path.** Stage ④ forms $\hat{\alpha} = \mathrm{round_int8}\big(\mathrm{amax} \cdot \nu^*/(1.65\,\sigma)\big)$. With the top grid level $1.65 = 1 + r$ and σ fixed before deployment, ④ multiplies amax by one of two constants. Rather than dividing each element by $\hat{\alpha}\sigma$ and picking the nearest level, ⑤ compares $|x[d]|$ against the four quantization thresholds scaled to the actual grid, $\hat{\alpha}\,\sigma\tau$. The grid is symmetric about 0 and monotone, so both pick the same point, and D divisions per token and head become four constant multiplications and $4D$ comparisons. For V, the thresholds $\hat{s}_v\tau$ are precomputed.

4.4 Multiplier and Bit Accounting

The weight-side ratio is 1.5625%. The rotation xR lies outside the reduction datapath; its ratio to the GEMM's $o_c i_c$ multiply-accumulates is $14/o_c$, whose weighted average over all linear layers is 0.246% on LLaMA-2-7B and 0.96% on Qwen2.5-0.5B. On the KV side, $QK^\top$ and the write path are at $1/D$ and PV at $1/N$, so for $N \geq D$ the ratio is at most $1/D$: the budget at $D = 64$ and 0.78125% at $D = 128$ (Figure 3(e)). Since linear layers, $QK^\top$ and PV take a ternarized right operand, every matrix product inside the Transformer layers except xR fits the budget; the unquantized embedding and output layers lie outside them. These ratios are analytical counts of fully general multipliers in our circuit model, not a measured speed-up (the evaluation uses reconstructed weights); latency, area and energy depend on circuit design, which is out of scope.

Stored bits. We store the cache in fixed-length base-9 packing, five joint symbols to a 2-byte word: per token and KV head, K and V each take $\lceil D/5 \rceil$ words, and K adds a 1-byte $\hat{\alpha}$. This averages 3.3125 bits per element at $D = 64$, a $4.830\times$ compression against FP16 K and V, and 3.28125 bits, $4.876\times$, at $D = 128$. A fixed cache memory thus holds about 4.8 times as many tokens; we did not measure throughput. For each ESPTQ-wkv model-ρ pair of Table 2, we checked on a 512-token WikiText-2 window that unpacking the stored cache reproduces bit for bit the evaluated configuration's cache. The static tables are counted as a total, not per element: 78.0 KiB on Qwen2.5-0.5B.

5 Experiments

Most runs use one NVIDIA RTX A6000; its 48 GB caps our models at Qwen2.5-14B, because end-to-end logit distillation backpropagates through the whole model. Under low-bit weight PTQ the gap to FP16 tends to be largest for small models (Frantar et al., 2023; Dettmers et al., 2024; Wang et al., 2026), so our range includes that regime. Yet beyond 13B the perplexity ratio to FP16 need not keep shrinking (Lin et al., 2024; Xiao et al., 2025), so we make no claim beyond our range.

We report WikiText-2 perplexity at sequence length 2048 (lower is better) and 0-shot[8] (0-sh[8] in the tables; higher is better), the mean accuracy on eight zero-shot tasks ARC-e, ARC-c, PIQA, HellaSwag, WinoGrande, BoolQ, OpenBookQA and LAMBADA. Appendix G gives software versions, each task's accuracy variant, baseline settings and our PTQTP reproduction. §5.2 adds GSM8K accuracy (8-shot chain-of-thought, 1319 questions; GSM in the tables), because only GSM8K requires sequential decoding, over which KV-cache error accumulates. No evaluation exceeds 2048 tokens.

Table 1: Weight-only quantization. **Seven of 13 models** (all in Appendix H). #W and #W* (13-model mean; 12 for PB-LLM) are nominal and effective bits. Wiki is WikiText-2 perplexity. Parenthesized values are self-reported. Qwen3-4B is Qwen3-4B-Base.

Method	Bits		SmolLM-135M		LLaMA-3.2-1B		Qwen3-4B		LLaMA-2-7B		Qwen2.5-7B		LLaMA-2-13B		Qwen2.5-14B	
	#W	#W*	0-sh[8]	Wiki	0-sh[8]	Wiki	0-sh[8]	Wiki	0-sh[8]	Wiki	0-sh[8]	Wiki	0-sh[8]	Wiki	0-sh[8]	Wiki
FP16	16	16	47.63	17.56	57.63	9.76	67.82	7.90	67.56	5.47	70.49	6.85	70.19	4.88	72.97	5.29
GPTQ	2.00	1.986	31.00	1.85e4	31.12	1.62e3	34.92	82.78	36.06	29.23	43.57	23.84	48.19	12.40	43.43	22.59
QuIP	2.00	1.869	31.57	2.26e3	31.64	1.06e3	36.77	47.61	32.42	82.92	49.26	17.17	45.42	15.67	46.95	16.73
PB-LLM	1.70	2.491	–	–	34.70	203.10	44.81	41.85	31.54	74.99	47.51	30.71	31.38	268.36	53.76	28.28
BiLLM	1.08	3.343	32.22	5.98e3	31.79	1.76e3	36.67	205.34	39.12	32.10	48.83	29.50	46.55	22.87	56.50	23.38
ARB-LLM	1.08	2.757	34.98	313.91	35.93	136.65	51.40	25.41	51.44	15.97	56.24	17.81	58.71	12.11	60.53	14.66
PT2-LLM	1.58	2.057	31.51	260.75	31.23	78.91	34.82	38.82	43.73	11.73	32.45	55.39	46.04	9.81	43.69	18.85
PTQTP†	1.58	–	–	–	–	(17.15)	–	–	–	(**6.32**)	–	(8.31)	–	(**5.29**)	–	–
CAT-Q‡	1.58	–	–	–	–	–	–	–	56.36	11.70	–	–	–	–	–	–
TWLA	1.58	5.416	38.28	58.93	44.22	19.62	59.40	10.82	61.30	7.02	63.48	8.60	**66.61**	5.76	**70.72**	7.43
gsp-0.2	1.58	2.416	36.89	46.81	42.28	20.30	55.51	11.19	54.69	7.88	61.82	9.21	60.53	6.52	63.22	8.06
ESPTQ-w$_{0.02}$	1.58	2.115	35.72	43.54	43.91	19.24	57.51	10.97	56.63	7.51	60.32	9.01	63.17	6.29	66.31	7.98
ESPTQ-w$_{0.12}$	1.58	2.656	**40.16**	**31.08**	**47.72**	**15.67**	**60.45**	**9.71**	**61.33**	6.66	**66.65**	**8.18**	66.06	5.72	70.26	**7.05**

Baselines keep their papers' calibration sets, WikiText-2 for PT2-LLM and TWLA. Ours mixes 96 WikiText-2 and 32 C4 training-split windows of 2048 tokens. Weight-side end-to-end logit distillation uses 24 separate windows from the same splits (Figure 1(e)), so no WikiText-2 test text enters any calibration stage, KV side included. We skip conditions the upstream code does not support. Each result is a single fixed-seed run; three calibration seeds give LLaMA-2-7B's WikiText-2 perplexity a 0.254% standard deviation.

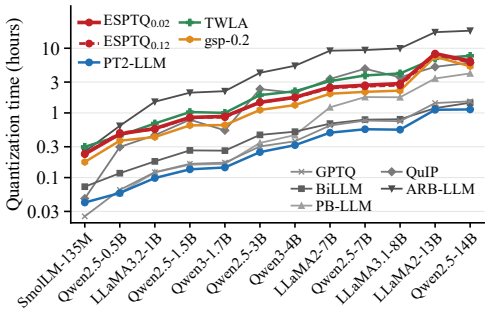


Figure 4: Quantization time on one A6000.

5.1 Weight Quantization

Element-wise placement. In Table 1,[1] PT2-LLM stores 2.057 bits, nearly as much as our $\rho = 0.02$ at 2.115, but ours has lower WikiText-2 perplexity on all seven models, 36% lower on LLaMA-2-7B (11.73 to 7.51). Most of this gain does not come from the second plane; Appendix I ablates the rotation and gives further comparisons and negative results. In Figure 2(b), on the same model, our pipeline (with *blockfly* and calibration) reaches 8.23 at 1.91 bits with no second plane ($p = 0$), and $\rho = 0.02$ lowers it to 7.44 at 2.10 bits. That figure uses another calibration set and machine, so this split is approximate. The control gsp-p, which shares our pipeline but gives its second plane an independent scale α_2, comes closest to isolating element versus group placement. Using 0.30 bits less than gsp-0.2, our $\rho = 0.02$ is still lower in WikiText-2 perplexity on all seven: judged by perplexity, element placement is more expressive, though the bit gap depends on the storage format. In 0-shot[8], our $\rho = 0.02$ beats gsp-0.2 on five and loses on two.

Quantization time. Figure 4 plots the time to load, quantize and save a model for Table 1's methods except PTQTP and CAT-Q. At $\rho = 0.02$, ESPTQ takes 2.5 hours on LLaMA-2-7B (PT2-LLM: 0.5; TWLA: 3.1) and at most 8.3 hours, on LLaMA-2-13B. On the figure's 12 models it takes 4.8–8.2× as long as PT2-LLM, which lacks our rotation learning, prepass and two-stage calibration.

Nominal versus effective bits. TWLA declares 1.58 bits and reports about 1.6 bits from an efficiency test packing five ternary values, one per weight, into each byte. Its public code, run with its defaults, closely reproduces its published perplexity (7.02 against 6.97 on LLaMA-2-7B) but stores 5.416 bits (13-model mean), 3.4 times its nominal bits. These bits hold zlib-compressed symbols (two ternary planes and a four-way partition index per weight), FP16 scaling factors and offsets, and KOTMS's Kronecker factors, not the one ternary value per weight it declares. Without entropy coding, a layout with five T_1 symbols per byte, plus a 7-bit in-group position and a sign

[1] PB-LLM's #W of 1.70 comes from later work (Huang et al., 2024; Li et al., 2025; Yan et al., 2026); its own count, with a position bitmap, is at most 2.7 bits for its example of 10% of weights in INT8 (Shang et al., 2024). The PTQTP (†) row shows only published values, because our reproduction fell short. CAT-Q (‡) releases no quantization code, so its row is our evaluation of its public LLaMA-2-7B weights.

Table 2: Ternary KV cache (ESPTQ-wkv) on ternary weights (ESPTQ-w), FP16 activations. ESPTQ-wkv stores 3.29 bits per cached element (#KV*). #W* averages each row's measured models; ESPTQ-w rows also average Qwen3-4B, not shown. Quantized rows declare 1.58 bits (#W). ◇Measured outside Table 1's environment (RTX A5000 or RTX 4500 Ada).

Method	Bits	SmolLM-135M◇			Qwen1.5-4B◇			Qwen2.5-3B◇			LLaMA-3.2-3B◇			LLaMA-2-7B			Qwen2.5-7B		
	#W*	0-sh^8	Wiki	GSM	0-sh^8	Wiki	GSM	0-sh^8	Wiki	GSM	0-sh^8	Wiki	GSM	0-sh^8	Wiki	GSM	0-sh^8	Wiki	GSM
FP16	16.000	47.59	17.56	2.05	62.27	8.94	55.19	66.13	8.03	59.21	65.36	7.82	27.22	67.56	5.47	13.50	70.49	6.85	80.36
PT2-LLM	2.061	31.11	200.33	0.00	37.64	25.90	0.99	33.89	50.34	0.99	35.50	39.17	0.00	43.73	11.73	1.67	32.45	55.39	0.08
TWLA	5.431	38.02	51.98	0.76	55.61	11.48	9.10	**58.10**	10.92	**21.15**	56.28	11.98	4.17	61.30	7.02	**4.09**	63.48	8.60	**54.74**
ESPTQ-w$_{0.02}$	2.115	37.39	42.89	0.53	51.87	12.55	4.25	53.39	11.90	6.14	51.73	12.59	2.12	56.63	7.51	1.82	60.32	9.01	27.52
ESPTQ-w$_{0.12}$	2.654	**39.73**	**30.97**	**1.52**	**55.96**	**11.10**	**12.28**	56.94	**10.38**	13.65	**57.41**	**10.69**	**5.53**	**61.33**	**6.66**	3.64	**66.65**	**8.18**	46.25
ESPTQ-wkv$_{0.02}$	2.115	36.63	50.03	0.38	49.73	13.33	1.82	51.76	12.64	2.88	49.23	13.69	2.12	53.37	8.10	1.67	59.07	10.05	17.97
ESPTQ-wkv$_{0.12}$	2.655	38.51	35.80	**1.52**	54.06	11.81	6.67	55.29	10.99	7.58	52.88	11.69	2.43	58.12	7.13	2.81	64.84	9.05	26.61

bit per nonzero T_2, needs at most 1.99 bits per weight on LLaMA-2-7B at $\rho = 0.02$ and 2.79 at $\rho = 0.12$ (13-model #W*: 2.115 and 2.656), counting scaling factors, offsets, rotation angles and read positions of the variable-length T_2 lists, and PT2-LLM 1.88, its column permutation included; these sizes are analytical, with no kernel built.

Against TWLA we match accuracy at less than half the storage. Our $\rho = 0.12$ stores 49% of its effective bits. WikiText-2 perplexity is lower on all seven, though the 0.69% gap on LLaMA-2-13B is below $2\sqrt{2}$ times §5's seed standard deviation (0.72%). 0-shot^8 is higher on five, though by only 0.03 points on LLaMA-2-7B, and lower on LLaMA-2-13B and Qwen2.5-14B. On SmolLM-135M, however, $\rho = 0.12$ stays far from FP16 in WikiText-2 perplexity (31.08 vs. 17.56).

5.2 KV Cache Quantization

In Table 2, the ESPTQ-w and ESPTQ-wkv rows at each ρ quantize the weights with the same settings in separate runs, except that on LLaMA-2-7B and Qwen2.5-7B the ESPTQ-w perplexity and 0-shot^8 come from Table 1's run, which used another code version and zero-shot batch size. Apart from re-rounding the α and μ of W_q and W_k when c is absorbed, run-to-run variation and that evaluation run, the two rows differ by the cost of ternarizing the KV cache. Their #W* gaps reflect different model sets. Per cached element, K and V take 3.375 and 3.25 bits on SmolLM-135M ($D = 64$) and 3.3125 and 3.25 bits on the other models ($D = 128$).

Of the two stages, ternarizing the KV cache is the cheaper. On all 12 model-ρ pairs, this stage costs less in perplexity and 0-shot^8 than ternarizing the weights. The stages quantize different things, so this compares their increments. On LLaMA-2-7B at $\rho = 0.12$, the KV stage raises WikiText-2 perplexity by 7.1% and lowers 0-shot^8 by 3.21 points, against 21.8% and 6.23 points for ternarizing the weights. To our knowledge, no prior work holds the KV cache in our storage format at our residual multiplication ratio, so our weight-only row is the control. GSM8K keeps the order on models with a high FP16 score: on Qwen2.5-7B at $\rho = 0.02$ the weights take it from 80.36 to 27.52 and the KV cache to 17.97, and on Qwen1.5-4B at $\rho = 0.12$ from 55.19 to 12.28, then 6.67. The KV stage costs GSM8K relatively more than perplexity, consistent with KV-cache error accumulating over sequential decoding. SmolLM-135M and LLaMA-2-7B score only 2.05 and 13.50 in FP16, and their quantized scores, 0.00 to 4.09, are too near the floor to show the KV stage's cost: at about 3% accuracy, the 95% confidence interval over 1319 questions is ± 0.9 points.

6 Conclusion

ESPTQ holds the weights and the KV cache in one ternary storage format, so every matrix product in a Transformer layer except the rotation xR needs only ternary reductions and multiplications at the accumulator output. By design, ESPTQ caps the residual multiplication ratio at $2/128 = 1.5625\%$. The element-wise sparse second plane makes effective bits continuous in ρ. Appendix J discusses the limitations. Four directions remain: **multi-step reasoning** (GSM8K), which degrades most in Table 2, mainly from the weights rather than the KV cache (§5.2), must improve; **a fused decode kernel** may turn memory savings into speed; **PeRQ's pre-rotation permutation** may compensate for group confinement (Sanjeet et al., 2026); and **the circuit** awaits building and measurement.

AI USE STATEMENT

Following the ICLR 2027 AI Policy for Authors, we report the tasks with required disclosure first and those with recommended disclosure after them. We used generative AI tools for five tasks with required disclosure. (i) Implementing methods: under the authors' direction, they wrote and debugged the code for quantization, the KV cache, evaluation and bit accounting, including the ports of the baselines into our harness. (ii) Assisting with translation: they translated the authors' drafts of the paper and its appendices into English. (iii) Providing feedback on research methodology and experiments: for example, estimating run times on the prepared machines, periodically checking running jobs for errors, and collecting experimental results (writing the commands and retrieving the results in bulk). (iv) Assisting in the writing of proofs: from the authors' formulations, they wrote out and numerically checked the derivations in the appendices, such as the squared-error-optimal inter-plane ratio and the reduction of the write-path selection test to ternary reductions. (v) Interpreting results: they drafted sentences describing measured results, such as the comparisons across methods and models in the appendices, which the authors revised.

We have not used generative AI tools for proposing the method itself, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, or designing the research methodology or experiments. Generating synthetic datasets, cleaning and reformatting datasets, and supporting qualitative or thematic data analysis are not applicable to this work.

Additionally, we used generative AI tools for tasks with recommended disclosure: creating or editing software code beyond the methods (typesetting, table-generation and checking scripts); cropping and masking figures, and checking the correctness and layout of the figures we made (e.g., Figures 1 and 3); drafting parts of the paper, mainly condensing the appendices and supplementary sentences from the authors' material; suggesting the structure of the appendices; editing the paper to improve readability; searching for, identifying and summarizing existing literature; checking citations against the cited papers; formatting references; and deleting the commented-out parts of the supplementary material and translating it into English. So that our use of AI can be observed objectively, we also used generative AI tools to write this AI use statement.

We have reviewed all AI-assisted work. For code, each baseline ported from its public code into our harness was checked, before use, against its published values or against the results of its public code's own harness. Bit counts are measured from the files actually written, and for every ESPTQ-wkv configuration of Table 2, unpacking the stored cache reproduced the evaluated cache bit for bit on a 512-token window. For numbers, Table 2 and the result tables of the appendices are generated by scripts, which check every value they share with Tables 1 and 2. For text, the translation was checked against the authors' drafts, terminology was fixed by a term list and an automated checker, and the derivations were checked numerically. Every cited statement was checked against the cited paper for consistency. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix E gives the full ESPTQ-w procedure (Algorithm 1) with its calibration settings, and Appendix F the KV-side derivations, calibration settings and write and read procedure (Algorithm 2). Appendix B defines the storage format and the rule by which every method's effective bits are measured. Appendix G lists the machines and software versions, the evaluation protocol, the default settings and seeds, and each baseline's code, settings and reproduction against published values. All models and datasets are public. Every result is a single fixed-seed run, and Appendix G reports the seed variation we measured.

REFERENCES

Khubaib Ahmed, Amna Noor, and Ahsan Ul Haq. Shift-accumulate attention: Multiplier-free query–key products for Transformer decoding. arXiv preprint arXiv:2609.09208, 2026.

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query Transformer models from multi-head check-

points. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, et al. Qwen technical report. arXiv preprint arXiv:2309.16609, 2023.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Leandro von Werra, and Thomas Wolf. SmolLM – blazingly fast and remarkably powerful. Hugging Face blog. `https://huggingface.co/blog/smollm`, 2024.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, et al. SmolLM2: When smol goes big – data-centric training of a small language model. arXiv preprint arXiv:2502.02737, 2025.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432, 2013.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, et al. Lessons from the trenches on reproducible evaluation of language models. arXiv preprint arXiv:2405.14782, 2024.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher De Sa. QuIP: 2-bit quantization of large language models with guarantees. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Mengzhao Chen, Wenqi Shao, Peng Xu, Jiahao Wang, Peng Gao, Kaipeng Zhang, and Ping Luo. EfficientQAT: Efficient quantization-aware training for large language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Towards the limit of network quantization. In *International Conference on Learning Representations (ICLR)*, 2017.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. arXiv preprint arXiv:1803.05457, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Tri Dao, Albert Gu, Matthew Eichhorn, Atri Rudra, and Christopher Ré. Learning fast algorithms for linear transforms using butterfly factorizations. In *International Conference on Machine Learning (ICML)*, 2019.

Hassan Dbouk, Hetul Sanghvi, Mahesh Mehendale, and Naresh Shanbhag. DBQ: A differentiable branch quantizer for lightweight deep neural networks. In *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pp. 90–106. Springer, 2020.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-bit matrix multiplication for Transformers at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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 *International Conference on Learning Representations (ICLR)*, 2024.

L. Peter Deutsch and Jean-Loup Gailly. ZLIB compressed data format specification version 3.3. RFC 1950, Internet Engineering Task Force, 1996.

Ziang Duan, Jiajun Wu, Zetian Chen, Hao Song, Yanwen Deng, Zixuan Shen, Nuobei Xie, Simo Wu, et al. Unified lookup-table inference with signed-digit K/V caches for ternary LLMs. arXiv preprint arXiv:2608.03229, 2026.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. In *International Conference on Machine Learning (ICML)*, 2024.

Elias Frantar, Sidak Pal Singh, and Dan Alistarh. Optimal brain compression: A framework for accurate post-training quantization and pruning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained Transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation. Zenodo, v0.4.0, 2023.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022.

Matteo Grella. Tied trit-planes: Constraining PTQTP to a uniform nine-level quantizer, with a persistent folded format for disk-streamed mixture-of-experts serving. arXiv preprint arXiv:2608.08910, 2026.

Yiwen Guo, Anbang Yao, Hao Zhao, and Yurong Chen. Network sketching: Exploiting binary structure in deep CNNs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4040–4048, 2017.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations (ICLR)*, 2021.

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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Chi-Wei Huang and Chia-Chi Tsai. Qift: Shift-friendly no-zero W2 post-training quantization for rotated W2A4/KV4 LLM inference. arXiv preprint arXiv:2606.02823, 2026.

Hong Huang, Decheng Wu, Rui Cen, Guanghua Yu, Zonghang Li, Kai Liu, Jianchen Zhu, Peng Chen, Xue Liu, and Dapeng Wu. Tequila: Trapping-free ternary quantization for large language models. arXiv preprint arXiv:2509.23809, 2025a.

Hong Huang, Decheng Wu, Qiangqiang Hu, Guanghua Yu, Jinhai Yang, Jianchen Zhu, Xue Liu, and Dapeng Wu. Sherry: Hardware-efficient 1.25-bit ternary quantization via fine-grained sparsification. arXiv preprint arXiv:2601.07892, 2026.

Wei Huang, Yangdong Liu, Haotong Qin, Ying Li, Shiming Zhang, Xianglong Liu, Michele Magno, and Xiaojuan Qi. BiLLM: Pushing the limit of post-training quantization for LLMs. In *International Conference on Machine Learning (ICML)*, 2024.

Wei Huang, Haotong Qin, Yangdong Liu, Yawei Li, Qinshuo Liu, Xianglong Liu, Luca Benini, Michele Magno, Shiming Zhang, and Xiaojuan Qi. SliM-LLM: Salience-driven mixed-precision quantization for large language models. In *International Conference on Machine Learning (ICML)*, 2025b.

Ruipeng Kang, Xiao Zheng, Patrick Lp, Liufeng Wang, Minquan Fang, Zhengbiao Wu, and Xiantao Zhang. KVTQ: Compressing the KV cache to hardware efficient ternary digits by fine-grained dynamic quantization. OpenReview preprint, submitted to ICLR 2024. `https://openreview.net/forum?id=eZAlb8fX5y`, 2023.

Ayush Kaushal, Tejas Vaidhya, Arnab Kumar Mondal, Tejas Pandey, Aaryan Bhagat, and Irina Rish. Spectra: Surprising effectiveness of pretraining ternary language models at scale. In *International Conference on Learning Representations (ICLR)*, 2025.

Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W. Mahoney, and Kurt Keutzer. SqueezeLLM: Dense-and-sparse quantization. In *International Conference on Machine Learning (ICML)*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. arXiv preprint arXiv:1605.04711, 2016.

Zhiteng Li, Xianglong Yan, Tianao Zhang, Haotong Qin, Dong Xie, Jiang Tian, Zhongchao Shi, Linghe Kong, Yulun Zhang, and Xiaokang Yang. ARB-LLM: Alternating refined binarizations for large language models. In *International Conference on Learning Representations (ICLR)*, 2025.

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 LLM compression and acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Yujun Lin, Haotian Tang, Shang Yang, Zhekai Zhang, Guangxuan Xiao, Chuang Gan, and Song Han. QServe: W4A8KV4 quantization and system co-design for efficient LLM serving. In *Proceedings of Machine Learning and Systems (MLSys)*, 2025.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. In *International Conference on Learning Representations (ICLR)*, 2025a.

Zechun Liu, Changsheng Zhao, Hanxian Huang, Sijia Chen, Jing Zhang, Jiawei Zhao, Scott Roy, Lisa Jin, Yunyang Xiong, Yangyang Shi, Lin Xiao, Yuandong Tian, Bilge Soran, Raghuraman Krishnamoorthi, Tijmen Blankevoort, and Vikas Chandra. ParetoQ: Improving scaling laws in extremely low-bit LLM quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025b.

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 *International Conference on Machine Learning (ICML)*, 2024.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28 (2):129–137, 1982.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit LLMs: All large language models are in 1.58 bits. arXiv preprint arXiv:2402.17764, 2024.

Anirudh Malik, M Sparsh Mehra, and Poojith Devan. Scaling post-training ternarisation to Qwen3-8B capability retention, reproduction, lossless packing, and packed execution. arXiv preprint arXiv:2609.09240, 2026.

Joel Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, 6(1):7–12, 1960.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

Qwen Team. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115, 2024.

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

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Sai Sanjeet, Ian Colbert, Pablo Monteagudo-Lago, Giuseppe Franco, Yaman Umuroglu, and Nicholas J. Fraser. Pushing the limits of block rotations in post-training quantization. In *International Conference on Machine Learning (ICML)*, 2026.

Yuzhang Shang, Zhihang Yuan, Qiang Wu, and Zhen Dong. PB-LLM: Partially binarized large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

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 *International Conference on Learning Representations (ICLR)*, 2024.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced Transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Zunhai Su, Zhe Chen, Wang Shen, Hanyu Wei, Linge Li, Huangqi Yu, and Kehong Yuan. RotateKV: Accurate and robust 2-bit KV cache quantization for LLMs via outlier-aware adaptive rotations. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2025.

Qian Tao, Wenyuan Yu, and Jingren Zhou. AsymKV: Enabling 1-bit quantization of KV cache with layer-wise asymmetric quantization configurations. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2025.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, et al. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. QuIP#: Even better LLM quantization with Hadamard incoherence and lattice codebooks. In *International Conference on Machine Learning (ICML)*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Hongyu Wang, Shuming Ma, and Furu Wei. BitNet a4.8: 4-bit activations for 1-bit LLMs. arXiv preprint arXiv:2411.04965, 2024.

Shigeng Wang, Chao Li, Yangyuxuan Kang, Jiawei Fan, and Anbang Yao. CAT-Q: Cost-efficient and accurate ternary quantization for LLMs. In *International Conference on Machine Learning (ICML)*, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, et al. HuggingFace's Transformers: State-of-the-art natural language processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020.

Chaodong Xiao, Zhengqiang Zhang, and Lei Zhang. BinaryAttention: One-bit QK-Attention for vision and diffusion Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning (ICML)*, 2023.

He Xiao, Runming Yang, Qingyao Yang, Wendong Xu, Zheng Li, Yupeng Su, Zhengwu Liu, Hongxia Yang, and Ngai Wong. PTQTP: Post-training quantization to trit-planes for large language models. arXiv preprint arXiv:2509.16989v1, 2025.

Bingxin Xu, Zhen Dong, Oussama Elachqar, and Yuzhang Shang. ButterflyQuant: Ultra-low-bit LLM quantization through learnable orthogonal butterfly transforms. arXiv preprint arXiv:2509.09679, 2025.

Xianglong Yan, Chengzhu Bao, Zhiteng Li, Tianao Zhang, Kaicheng Yang, Haotong Qin, Ruobing Xie, Xingwu Sun, and Yulun Zhang. PT^2-LLM: Post-training ternarization for large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. WKVQuant: Quantizing weight and key/value cache for large language models gains more. arXiv preprint arXiv:2402.12065, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

Tianyi Zhang, Jonah Wonkyu Yi, Bowen Yao, Zhaozhuo Xu, and Anshumali Shrivastava. NoMAD-Attention: Efficient LLM inference on CPUs through multiply-add-free attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Zhixiong Zhao, Zukang Xu, Zhixuan Chen, Xing Hu, Zhe Jiang, and Dawei Yang. TWLA: Achieving ternary weights and low-bit activations for LLMs via post-training quantization. In *International Conference on Machine Learning (ICML)*, 2026.

Fei Zuo, Xiaoyan Xi, Quanyi Zeng, Feiyu Wang, and Ho Fai Leung. FairyFuse: Multiplication-free LLM inference on CPUs via fused ternary kernels. arXiv preprint arXiv:2604.20913, 2026.

APPENDIX OVERVIEW

Each appendix supports the main-text sections listed on its row; its first sentence repeats them. Algorithm 1 is in Appendix E and Algorithm 2 in Appendix F.

Appendix	Title	Supports
Appendix A	Extended Related Work	§1, §2 (§5.2)
Appendix B	Storage Format, Grid and Weight-Side Accounting	§3.1 (§1, §4.1, §4.4)
Appendix C	The *blockfly* Rotation	§3.2 (§2, §4.4)
Appendix D	The Prepass	§3.3 (§5.1)
Appendix E	Compensation, Calibration and the Full ESPTQ-w Procedure	§3.4 (§5)
Appendix F	Ternary KV Cache: Derivations and Settings	§4.1–§4.4 (§5, §5.2)
Appendix G	Experimental Setup and Reproducibility	§5 (§5.1 footnote)
Appendix H	Full Results	§5.1, §5.2 (§1)
Appendix I	Additional Comparisons, Ablations and Negative Results	§1, §2, §3.2–§3.4, §5.1
Appendix J	Limitations	§6

A EXTENDED RELATED WORK

This appendix supports §1 and §2. Appendix A.3 also backs the statement of §5.2 that no prior work holds the KV cache in our storage format at our residual multiplication ratio.

A.1 STACKED PLANES, TERNARY PTQ BASELINES AND CONTEMPORANEOUS WORK

Stacked planes. Network Sketching (Guo et al., 2017) and ABC-Net (Lin et al., 2017) sum scaled binary planes, each with its own scaling factor. BiLLM (Huang et al., 2024) and ARB-LLM (Li et al., 2025) add a residual binary plane on salient columns of LLM weights. DBQ (Dbouk et al., 2020), by QAT, and PTQTP (Xiao et al., 2025), by PTQ, use two dense ternary planes with a scaling factor each. PB-LLM (Shang et al., 2024) keeps salient weights in 8 bits and marks them in a bitmap of one bit per weight. SpQR (Dettmers et al., 2024) and SqueezeLLM (Kim et al., 2024) keep sparse outliers in FP16, indexed in the compressed sparse row (CSR) format, beside a dense 3–4-bit grid. SliM-LLM (Huang et al., 2025b) gives each 128-column group $b-1$, b or $b+1$ bits.

Where ESPTQ-w differs. ESPTQ-w marks elements without a second plane by $T_2 = 0$ in the joint symbol instead of a mask (§3.1). Their positions then cost the entropy of T_2, $\mathcal{H}(\rho) = h_{\mathrm{b}}(\rho) + \rho$ bits per weight, with ρ the second-plane assignment ratio and h_{b} the binary entropy (Appendix D). This equals an ideally entropy-coded bitmap plus one sign bit per second-plane element, so the joint symbol removes the separate index structure, not the information that a mask carries.

Precursors and baselines. TWN (Li et al., 2016) ternarizes weights by a threshold rule of thumb. BitNet b1.58 (Ma et al., 2024), ParetoQ (Liu et al., 2025b) and Spectra (Kaushal et al., 2025) obtain ternary LLMs by QAT, so they are not PTQ baselines. Of our four PTQ baselines, PT2-LLM (Yan et al., 2026) reorders columns (SSR) and fits an offset and scaling factor per row and 128-column group with its asymmetric ternary quantizer (ATQ), which starts from TWN's rule and is reused by ESPTQ-w (§3.3). CAT-Q (Wang et al., 2026) learns an offset-free αT per group of 128 weights by gradient-based training and releases weights but no code (Appendix G). TWLA (Zhao et al., 2026) combines the quantizer E2M-ATQ, the rotation KOTMS (Appendix A.2) and mixed-precision activations, and declares 1.58 bits. Its public code also stores a second plane on salient columns and a per-weight partition index, for 5.458 effective bits per weight on LLaMA-2-7B (Appendix H).

Contemporaneous work. Five recent works are not rows of Table 1. Tequila (Huang et al., 2025a) and Sherry (Huang et al., 2026) train by QAT, Sherry with 3:4-sparse ternary weights at 1.25 bits before scaling factors. Tied Trit-Planes (Grella, 2026) is PTQTP with its two scaling factors tied at our $r = 1/3$ (§3.1). The technical report of Malik et al. (2026) runs TWLA's weight pipeline on Qwen3-8B without a new quantizer, and its 1.64 bits per weight count only the ternary symbols (Eq. (8) of that report). Qift (Huang & Tsai, 2026) is 2-bit, not ternary.

A.2 STRUCTURES OF THE ROTATION

Proposals for the rotation R of §2 differ in structure and objective. QuIP (Chee et al., 2023) uses a Kronecker product of random orthogonal matrices, and QuIP# (Tseng et al., 2024) and QuaRot (Ashkboos et al., 2024) use randomized Hadamard transforms. SpinQuant (Liu et al., 2025a) learns rotations by Cayley SGD on the final loss of the quantized network. ButterflyQuant (Xu et al., 2025) learns a butterfly of Givens rotations over the whole hidden dimension. Its objective is the layer-output reconstruction error under symmetric uniform quantization, plus a term that evens out the occupancy of the activations' quantization bins (Eqs. (16)–(19) of Xu et al. (2025)). KOTMS (Zhao et al., 2026) restricts R to a Kronecker product $R_1 \otimes R_2$ of Cayley-parameterized factors and fits the rotated weights to a symmetric three-component Gaussian mixture. The TWLA paper balances the two factor sizes, whereas the released KOTMS of TWLA's public code, which §3.2 compares with, sizes them from each layer's shape (Appendix C). PeRQ (Sanjeet et al., 2026) shows that, in the deterministic worst case, outliers after a block-Hadamard rotation are smallest when the pre-rotation ℓ_1 norm is spread evenly across blocks, and permutes channels before rotation accordingly.

A.3 KV-CACHE QUANTIZATION AND WHERE ITS MULTIPLICATIONS REMAIN

Table A1: Prior KV-cache methods as evaluated in their papers, and ESPTQ-kv. Ratios are general multiplications per multiply-accumulate (MAC); the last column omits the storage-format condition.

Method	Stored format	Scales vary along	General multiplications as evaluated (ratio per MAC)	Condition of §4.1 still unmet with bit-plane codes
KIVI	2-bit asymmetric integers; latest $\leq$ 128 tokens in full precision	K: channel, per 32 tokens; V: token	dequantization fused into both products: every MAC	accumulator output; budget (K scales into Q: $\geq 1/32$)
KVQuant	2–4-bit non-uniform indices; about 1% outliers in FP16; K before RoPE	K: channel, offline; V: token, online	table lookup, then FP16 per element: every MAC	budget (RoPE at every read: 200% of $QK^\top$ MACs on LLaMA-2-7B); accumulator output (V)
QuaRot	4-bit asymmetric integers; K Hadamard-rotated after RoPE	token and head	dequantization, then FP16 dot products: every MAC	accumulator output (V scales pre-multiplied into P)
KVTQ	residual ternary digits, 4 for K and 3 for V; FP16 weights	digit plane, token and head	$QK^\top$: per plane and score ($4/D$); PV: scales pre-multiplied into P^a ($3/D$)	budget ($4/D$ for every $D < 256$); accumulator output (PV)
NoMAD-Attention	4-bit product-quantization codes of K; V unchanged	codebook per head; table per query	table building (dot products); every MAC of PV	budget (every MAC of PV)
Shift-Accumulate	K: sign, power-of-two exponent; V: INT8; FP16 weights	K: token; V: token, power of two	$QK^\top$: two per score ($2/D$); PV: shifts only	budget at its $D = 64$ ($2/D$ meets it only for $D \geq 128$)
Signed-digit K/V cache	per-block number of signed-digit planes at power-of-two positions; open V block also in BF16	K: token, channel block; V: channel, token interval	one rescale per block after lookup and reduction (PV: $1/32$ at 32-token blocks)	budget; 64-token blocks meet it but re-encode a BF16 block at every append (§4.1)
ESPTQ-kv	two ternary planes ($r = 0.65$), base-9 packed; INT8 $\hat{\alpha}_t$ for K; static tables	K: token and KV head; V: static per channel	at the accumulator output only: $QK^\top$ $1/D$, PV $1/N$, write path $1/D$	none

[a] Our reading. The KVTQ paper says only that the quantization step is multiplied into the accumulated result.

Low-bit caches. KIVI (Liu et al., 2024), KVQuant (Hooper et al., 2024) and QuaRot (Ashkboos et al., 2024) hold K and V at 2–4 bits. Their per-token scaling factors of V, and the per-channel ones of K in KIVI and KVQuant, vary along the reduction dimension of PV or $QK^\top$, so they multiply every product or are pre-multiplied into P or Q (§4.1, §4.2; Table A1). The obstacle is where the scaling factors vary, not the integer codes, which bit planes make multiplication-free (§1). BitNet a4.8 (Wang et al., 2024), WKVQuant (Yue et al., 2024) and AsymKV (Tao et al., 2025) also scale V per token at run time (for BitNet a4.8, by our reading), so Table A1 omits them.

Multiplier-free attention. KVTQ (Kang et al., 2023), NoMAD-Attention (Zhang et al., 2024), Shift-Accumulate Attention (Ahmed et al., 2026) and the signed-digit K/V cache (Duan et al., 2026) remove multiplications from attention. Table A1 omits BinaryAttention (Xiao et al., 2026), which targets vision and diffusion Transformers, with no KV cache written during generation. The signed-digit cache, to our knowledge the closest to ours, runs linear layers, $QK^\top$ and PV on one datapath with digits in $\{-1, 0, +1\}$. ESPTQ-kv differs in five respects: signed adder trees instead of table lookups; two planes tied by $r = 0.65$ instead of a per-block number of planes at power-of-two positions; a static per-channel V table instead of scales per token interval; no BF16 copy of the open V block (§4.3); and symmetric INT8 scaling factors with reference values instead of FP16 scales.

No prior method meets all four conditions. Our claim is the conjunction of the four conditions of §4.1, cache writes included. No prior method in Table A1 shares our storage format, and each exceeds the budget as evaluated and still fails the condition in the last column with bit-plane codes. These comparisons back the qualified claims of §1 and §5.2 and the weight-only control of Table 2.

B Storage Format, Grid and Weight-Side Accounting

This appendix supports §3.1. It also backs the bit-plane remark of §1, the quantization thresholds of §4.1 and the matrix products counted in §4.4.

B.1 Counted matrix products and weight-side multipliers

Counted matrix products. Per Transformer layer, seven matrix products take a weight matrix as the right operand (W_q, W_k, W_v, W_o and the MLP's $W_{\text{gate}}, W_{\text{up}}, W_{\text{down}}$), and two take the KV cache ($QK^\top$ and PV, with P the softmax output). The multiplication-free claim of §1 and the conclusion of §4.4 concern these nine.

Weight side. In Eq. (1), a product with a ternary value is a sign flip or a skip, so both plane sums come from signed adder trees, and r is a design-time constant. Only α_{ig} and μ_{ig} multiply, twice per 128 multiply-accumulates (1.5625%), with $\sum_{j\in g} x_j$ shared by all rows. On both sides (Appendix F), we do not count products with design-time constants such as the reference values ($2/i_c = 0.049\%$ at $i_c = 4096$ if counted), operations outside the reduction datapath (softmax, RoPE, RMSNorm, SiLU, SwiGLU gating) or table lookups.

Integer bit planes. Split into k binary bit planes, an INT-k code is also multiplication-free, with one adder tree per plane, a shift per place value and, as for our α, μ, two multiplications per 128-element group. INT2, however, gets four levels from two adder trees, whereas our two signed adder trees give nine unevenly spaced levels. This counts adder trees, not circuit area (Appendix F).

B.2 The nine-level grid: inter-plane ratio and thresholds

Deriving 0.658. Rounding weights $w \sim \mathcal{N}(0, \sigma_w^2)$ to the nearest point of the nine-level grid with a free second-plane scaling factor α_2 gives the squared error

$$\mathcal{E}(\alpha, \alpha_2) = \mathbb{E}_w\Big[\min_{a,b\in\{-1,0,+1\}} \big(w - \alpha a - \alpha_2 b\big)^2\Big]. \tag{B1}$$

Minimizing Eq. (B1) with exact truncated moments gives $\alpha = 1.3801\,\sigma_w$ and $\alpha_2 = 0.9077\,\sigma_w$, a ratio of 0.6577 and an error of $0.02902\,\sigma_w^2$. This is 4.2% above the nine-level Lloyd–Max quantizer (Lloyd, 1982; Max, 1960) and 6.6 times below the best single ternary plane ($0.19017\,\sigma_w^2$).

Why 0.65, and why fixed. We took $r = 0.65$ from a coarser computation of this optimum; re-optimizing α at $r = 0.65$ costs 0.17%. A fixed r needs no stored α_2, and a least-squares r per linear layer, the one estimate we tried, gave worse perplexity at all four ρ tried (design stage, Qwen2.5-0.5B; Appendix I). The derivation assumes a dense grid; we did not re-derive r for ESPTQ-w, which gives nine levels only to the fraction ρ and fits α on three levels (§3.3).

Quantization thresholds. The midpoints of the levels $0, 0.35, 0.65, 1, 1.65$ give $\tau = \{0.175, 0.5, 0.825, 1.325\}$. As the grid is symmetric about 0, comparing a magnitude with τ and attaching the sign yields the nearest level, and $\Lambda(-T_1, -T_2) = -\Lambda(T_1, T_2)$ puts the sign in the joint symbol. Elements without a second plane round at ± 0.5.

B.3 Stored items, reference values and effective bits

Stored items. Table B1 lists what ESPTQ-w writes per linear layer; unlike stacked planes (Appendix A), it holds no second-plane scaling factor. A directory of shapes, offsets and the constants r, G counts only in the file size.

Table B1: What ESPTQ-w stores per linear layer, in this order (row-major), with fixed-length costs on LLaMA-2-7B (element-weighted). The total is the fixed-length accounting value.

Item	Per	Stored as	Bits per element
joint symbol $3(T_1+1)+(T_2+1)$	element	1 byte, zlib	3.170 ($\log_2 9$)
INT8 $\hat{\alpha}_{ig}+127$, $\hat{\mu}_{ig}+127$	row and group	1 byte, same stream	0.125 ($16/G$)
reference values $\sigma_\alpha, \sigma_\mu$	linear layer	FP16	1.1×10^{-6} ($32/o_c i_c$)
rotation angles (448 per group)	group	FP16	0.010 ($56/o_c$)
total (fixed-length value)			3.305

Reference values. The reference values of §3.1 are rounded to FP16, set on the prepass grid and fixed through the main loop, where a refitted α or μ beyond 127 counts saturates. After each calibration stage, they are recomputed and α, μ are re-projected, with no 1.25 margin, unlike on the KV side (Appendix F). This two-level scheme corresponds to QLoRA's double quantization (Dettmers et al., 2023), which, however, subtracts the mean, quantizes FP32 constants to FP8 in blocks of 256 and keeps FP32 second-level constants; we use symmetric INT8 and one FP16 reference value each for α and μ per linear layer.

Effective bits and the baselines' writing rule. One harness writes every method in its own storage format (§1). Each discrete code (symbol, integer zero point, partition index or mask bit) takes one byte, compressed with zlib at level 9 per linear layer, while continuous quantities stay in FP16 and column permutations in 32-bit integers, uncompressed. Four exceptions apply: (a) ARB-LLM (Li et al., 2025) writes its normally empty indices of zero signs into zlib as uint32; (b) QuIP (Chee et al., 2023) stores an uncompressed 8-byte seed per layer; (c) AWQ (Lin et al., 2024) keeps group zero points in FP16, as its public code does; (d) PT2-LLM (Yan et al., 2026) stores in FP16 any row of a block with more than three levels, an anomaly we did not check for in Table 1 that could only raise its #W*. CAT-Q's codes are read from its public weights. ESPTQ-w is written after both calibration stages with FP16 angles, whereas the evaluated models keep them in FP32; we did not evaluate the model with FP16 angles.

Table B2: Storage of LLaMA-2-7B in bits per quantized element, and file size (GB $= 10^9$ bytes). esp-ρ is ESPTQ-w at ρ; gsp-0.2 is the control of §3.3; its fixed-length value charges positions per column by a mask, its entropy lower bound per element. All rows are quantized as in Table 1.

	Fixed-length accounting value	Entropy lower bound	#W* (zlib)	k-gram code length	File (GB)
esp-0.02	3.305	1.794	**2.097**	1.761	2.22
esp-0.12	3.305	2.248	**2.644**	2.205	2.67
gsp-0.2 (control)	1.882	2.036	**2.397**	1.994	2.47

Reading Table B2. The entropy lower bound counts each plane's symbols at their zeroth-order entropy per linear layer, and the k-gram code length codes blocks of up to four symbols along rows; both include frequency tables and the metadata of Table B1. #W* lies 1.208 and 0.661 bits below the fixed-length value for esp-0.02 and esp-0.12. At $\rho = 0.02$, 1.424 of the 1.511 bits between the fixed-length value and the entropy lower bound are the sparse T_2, charged $\log_2 3$ bits against its entropy $\mathcal{H}(0.02) = 0.161$ bits (Appendix D). zlib stays 0.303 bits above the entropy lower bound, and the k-gram length lies below it in every row, so a coder using the dependence between planes could store less. Conversely, zlib exceeds even $\log_2 3$ on near-uniform codes (1.821 bits on CAT-Q's; Appendix H), penalizing dense ternary planes.

C THE *blockfly* ROTATION

This appendix supports §3.2. It also backs the PeRQ sentence of §2, the rotation ratio of §4.4 and the measurement behind Figure 2(a).

C.1 CONSTRUCTION, OBJECTIVE AND SETTINGS

Construction. We store quantized WR and compute xR at run time, since $xW^\top = (xR)(WR)^\top$ for orthogonal R. In each group of $G = 128$ input channels, stage $k = 0, \ldots, 6$ applies 64 Givens

rotations side by side, pairing coordinate i with $i+2^k$ as in the butterfly of the fast Fourier transform (Dao et al., 2019). The group's rotation $R_g = R_0 R_1 \cdots R_6$ is in general a dense orthogonal matrix, but we store only its $(G/2)\log_2 G = 448$ angles and apply the stages one by one. A linear layer's rotation is the block-diagonal $R = \bigoplus_g R_g$. All angles at 0 give $R_g = I$. All angles at $\pi/4$ give $R_g = E\,\mathrm{Had}_{128}/\sqrt{128}$, with E a diagonal sign matrix and Had_{128} the Sylvester Hadamard matrix; we call this unlearned setting *fixed Hadamard*. Unlike the randomized Hadamard transform of QuaRot (Ashkboos et al., 2024), it is deterministic and stays within the group.

Objective. For each row w_i, let $z_i = w_i R - \mathrm{mean}(w_i R)$ and $\kappa_i = \mathrm{mean}_j |z_{ij}|$, both taken over the whole row. The angles minimize

$$\begin{aligned}\mathcal{L} = &-\frac{1}{o_c i_c}\sum_{i=1}^{o_c}\sum_{j=1}^{i_c}\log\Big[\tfrac{1}{3}\sum_{c\in\{-1,0,1\}}\mathcal{N}\big(z_{ij};\, c\,\kappa_i, (0.25\kappa_i)^2\big)\Big] \\ &+ 0.1\sum_{s\in\{-1,0,1\}}\big(\bar{q}_s - \tfrac{1}{3}\big)^2 + 1.0\ \mathrm{mean}_j\big(\tilde{h}_j - 1\big)^2.\end{aligned} \tag{C1}$$

The first term is the negative log-likelihood of the equal-weight trimodal Gaussian mixture (TriGMM); κ_i and the width $0.25\kappa_i$ are recomputed at every iteration without gradient. In the occupancy-balancing term, $\bar{q}_s$ is the posterior of component s averaged over all entries of the layer. In the activation-flattening term, $\tilde{h}_j$ is the j-th entry of $\mathrm{diag}(R_g H_g R_g^\top)$ divided by its mean over group g, where H_g is the diagonal block of GPTQ's Hessian $H = X^\top X$ (Frantar et al., 2023). The equal weights, the width and the occupancy term with weight 0.1 follow the released KOTMS, the KOTMS of TWLA's public code, not the TWLA paper (Zhao et al., 2026); the flattening term is ours. The row-wide z_i and κ_i make the objective a per-row proxy for ESPTQ-w's per-group grid (§3.1); a per-group variant was not measured.

Orientation of the flattening term. Since the channel energies of the rotated input xR_g are $\mathrm{diag}(R_g^\top H_g R_g)$, the implemented term of Eq. (C1) evens out those of the inversely rotated input $xR_g^\top$, a mismatch found after the experiments that leaves the inference rotation, GPTQ's Hessian and the variants without the term unaffected, and its effect under quantized activations is not measured (Appendix I).

Settings. Adam (Kingma & Ba, 2015) with learning rate 0.01 runs 100 iterations from all angles at $\pi/4$ and moves all groups of a linear layer together. The angles are learned once per linear layer, inside the layer-wise loop before that layer is ternarized, from the Hessian already collected for GPTQ, so no extra data or forward pass is needed; GPTQ then uses $R^\top H R$ (Algorithm 1). This setting is arm A3 of the ablation in Appendix I, which also defines A1, A3h and A3w (named in Figure C1); A3w is the variant that §3.2 compares with KOTMS. In that ablation, A3 gives 1.59% higher WikiText-2 perplexity than A3w; the final configuration was fixed before the ablation, and Tables 1 and 2 were not re-run with A3w.

C.2 Group confinement and cost

Why groups. Within a group, the seven stages move data only among 128 elements, the columns that carry α and μ and that GPTQ quantizes at a time. A butterfly across 4096 inputs, as in ButterflyQuant (Xu et al., 2025), needs 12 stages, five of which pair coordinates 128 to 2048 apart across groups, and $96/o_c$ instead of $56/o_c$ angle bits per weight. Widening the span (arm A3g) did not lower perplexity beyond the decision margin (Appendix I). The price is that R preserves each group's norm, so no rotation of this form can move an outlier's mass out of its group. PeRQ (Sanjeet et al., 2026) analyzes this limit for block-confined Hadamard rotations of activations and permutes channels to balance the blocks' ℓ_1 norms; its analysis covers our $\pi/4$ initialization but not the learned angles directly.

Storage and multiplications (Table C1). The 448 FP16 angles of a group are shared by all o_c rows, so they cost $56/o_c$ bits per weight. The released KOTMS, whose code sizes its Kronecker

factors from each layer's shape, stores $16.4\times$ as much on LLaMA-2-7B. A Givens rotation takes four multiplications for two coordinates, so *blockfly* needs 14 per input element, whereas a Kronecker product of $n_1 \times n_1$ and $n_2 \times n_2$ factors needs $n_1 + n_2$ (Zhao et al., 2026). Against the factors balanced as in the TWLA paper (last row), *blockfly* stores more but needs far fewer multiplications. We count the direct form (lifting would give 10.5) and one rotation per linear layer, since Q, K and V learn separate angles for their shared input.

Table C1: Cost of five rotations on LLaMA-2-7B at input widths 4096 / 11008 (arms of Appendix I). Bits per element are measured in the first four rows; all other entries are analytical.

Rotation	Structure ($i_c = 4096$ / 11008)	Stored values	Bits per element	Mult. per input element	Ratio to GEMM (%)
fixed Hadamard (A1)	7 stages in 128 columns	0	0	0 / 0[c]	0
blockfly (ours; A3w)	7 stages in 128 columns	3,985,408	0.009847	14 / 14	0.246
global butterfly (A3g)	12 stages in 4096 columns[a]	5,951,488	0.014704	24 / 14	0.368
released KOTMS (A2)	$(8, 512)$ / $(16, 688)$	65,499,136	0.161826	520 / 704	10.1
balanced KOTMS[b]	$(64, 64)$ / $(128, 86)$	2,333,824	0.005766	128 / 214	2.72

[a] The 11008-wide down projection stays 128-column *blockfly*. [b] Analytical, not measured.
[c] No fully general multiplier: additions, subtractions and one constant scaling by $1/\sqrt{128}$.

Ratio to the GEMM. The rotation lies outside the reduction datapath (Appendix B). Rotating one input of length i_c takes $14\,i_c$ multiplications against the GEMM's $o_c i_c$ multiply-accumulates, a ratio of $14/o_c$, because all output rows share one xR. Weighted by multiply-accumulates over a decoder layer's seven linear layers, $14 \sum i_c / \sum o_c i_c$ is 0.246% on LLaMA-2-7B and 0.96% on Qwen2.5-0.5B (Qwen Team, 2024), whose output widths are smaller.

Unrotated layers. Linear layers whose input width is not a multiple of 128 stay unrotated: in Table 1 only SmolLM-135M's six projections of width 576 (all but down), and in Appendix H the width-960 layers of SmolLM2-360M.

C.3 Measurement behind Figure 2(a)

Conditions. Figure 2(a) and Figure C1 use the public copy NousResearch/Llama-2-7b-hf of LLaMA-2-7B, not checked for bit identity with Table 1's copy. The Hessians come from 128 windows of 2048 WikiText-2 training tokens (seed 0) passed through unquantized preceding layers in BF16, on the RTX 5070 Laptop machine of Appendix G rather than Table 1's. The *blockfly* angles are therefore learned anew with A3's settings, not taken from Table 1. KOTMS is the released code applied to the weights alone from identity, with the same width, occupancy weight and optimization budget and no flattening term.

Statistics. The kurt of Figure 2(a) is the Pearson kurtosis of each output row (3 for a Gaussian), averaged over rows. The err is $\|W - (\alpha T + \mu)\|_F^2 / \|W\|_F^2$ for one ternary plane T with one (α, μ) per row, fitted by at most 10 iterations of the iterative ternary fitting of PT2-LLM (Yan et al., 2026); it is not the error of ESPTQ-w's per-group grid. err locates where the rotation acts but does not rank rotations: over all 224 weight matrices, fixed Hadamard lowers it by only 1.92%, yet removing it raises WikiText-2 perplexity by 116.05% (Appendix I).

Unimodal with weak shoulders (Figure C1). Of six rotations applied to the Q, K, V and down projections of layer 24, only the released KOTMS splits the distribution, with dips near $|z/\kappa| = 0.5$ between the central mode and outer modes near ± 0.85. The density ratio $d(1)/d(0.5)$, where $d(a)$ sums the densities at $\pm a$ read from a 240-bin histogram of $[-4, 4]$, is 1.244 for KOTMS and 0.878–0.883 for the three *blockfly* variants. Folded and smoothed over five bins, the *blockfly* histograms keep a single peak at 0 with no local maximum for $0.3 < |z/\kappa| < 1.1$; the only departure from a monotone decay is a shoulder, flattest near 0.62, and the density at $|z/\kappa| = 1$ lies about 11% below it. Figure 2(a) shows the Q projection alone, so its printed values differ from these four-matrix means. Over all 224 matrices, fixed Hadamard removes 94.8% of the excess row kurtosis but only 36.4% of the err reduction of the learned angles, whose err exceeds neither in any matrix. The three *blockfly* variants are indistinguishable in these statistics although their perplexities spread by 1.59%

(Appendix I). Which of KOTMS's differences (span, storage or Cayley parameterization) splits the modes was not tested.

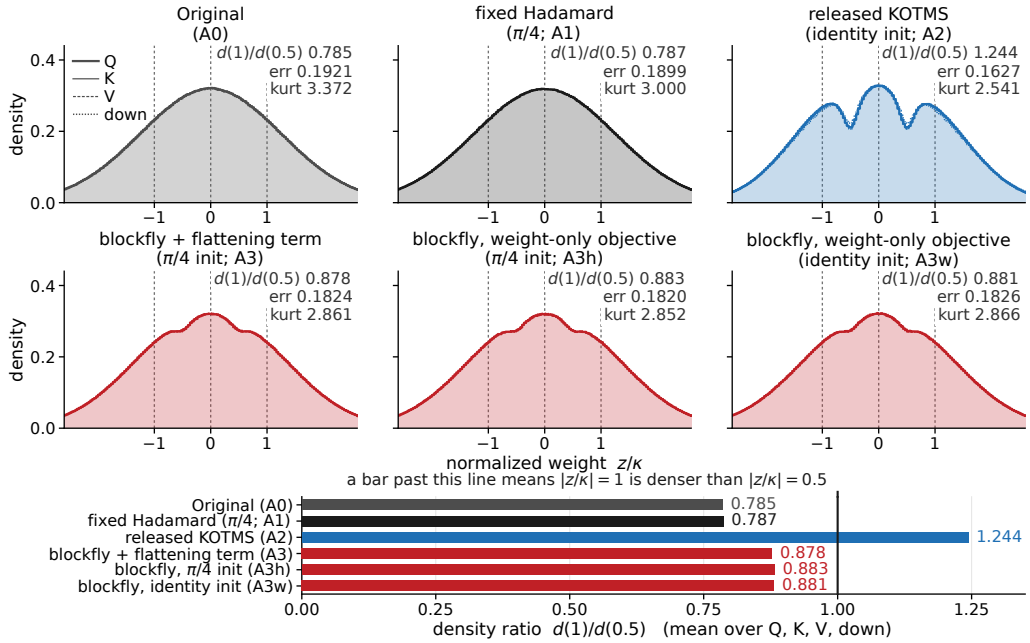


Figure C1: Distributions of z/κ for the Q (filled), K, V and down projections of layer 24 of LLaMA-2-7B under six rotations (arm symbols of Appendix I; dashed lines: ternary points). The printed kurt, err and $d(1)/d(0.5)$ are means over the four matrices; the bottom row compares the density ratio.

D THE PREPASS

This appendix supports §3.3 and the two points of Figure 2(b) that §5.1 cites.

D.1 ATQ: ITERATIVE TERNARY FITTING AND ACTIVATION-AWARE GRID ALIGNMENT

Initialization. ATQ, the asymmetric ternary quantizer of PT2-LLM (Yan et al., 2026), fits each row i of a group of $G = 128$ input channels. It subtracts the row mean μ_i from the weights w_i and applies the threshold rule of TWN (Li et al., 2016) to the centered weights $\tilde{w}_i$, giving the ternary row $t_{ij} = \operatorname{sign}(\tilde{w}_{ij})\,\mathbb{I}[\,|\tilde{w}_{ij}| > 0.75\,\overline{|\tilde{w}_i|}\,]$, where $\overline{|\tilde{w}_i|}$ is the mean magnitude; α_i starts as the mean of $|\tilde{w}_{ij}|$ over the nonzero t_{ij}. The coefficient 0.75 is PT2-LLM's, from the revised TWN; the original TWN uses 0.7.

$$\begin{pmatrix} \sum_j h_j t_{ij}^2 & \sum_j h_j t_{ij} \\ \sum_j h_j t_{ij} & \sum_j h_j \end{pmatrix} \begin{pmatrix} \alpha_i \\ \mu_i \end{pmatrix} = \begin{pmatrix} \sum_j h_j w_{ij} t_{ij} \\ \sum_j h_j w_{ij} \end{pmatrix}. \tag{D1}$$

ITF and AGA. Iterative ternary fitting (ITF) alternates two steps until t_i stops changing, at most 10 times: it solves Eq. (D1) with $h_j \equiv 1$, and it rounds each $(w_{ij} - \mu_i)/\alpha_i$ to the nearest of $-1, 0, +1$. Activation-aware grid alignment (AGA) keeps t_i and solves the same equations with $h_j = H_{jj}$ (§3.3), minimizing the layer-output error for a diagonal H_g; PT2-LLM's code uses the full H_g, and we have not measured what this approximation changes.

D.2 RATE-DISTORTION SELECTION AND THE ENTROPY OF T_2

Gain and code length. In body Eq. (2), $\varepsilon^{(3)} = \varepsilon$ and $\varepsilon^{(9)}$ are the errors to the nearest points of $\mu + \alpha a$ and $\mu + \alpha(a + rb)$, with $a, b \in \{-1, 0, +1\}$ and $r = 0.65$; the nine-level point may change T_1 too. Since the nine-level grid contains the three-level one, $0 \le \text{gain}_{ij} \le s_{ij}$. With $\ell(k) = -\log_2 \hat{p}(k)$ for joint symbol k, $\Delta\ell_{ij} = \ell(T_1^{(9)}, T_2^{(9)}) - \ell(T_1^{(3)}, 0)$.

Setting λ. One λ serves a whole linear layer, so groups with larger residuals receive more second planes. Elements with $\mathrm{gain} > 0$ and $\Delta\ell \leq 0$ are always selected; the rest of the budget $\mathrm{round}(\rho\, o_c i_c)$ is filled in decreasing order of $\mathrm{gain}/\Delta\ell$, and the last ratio admitted is λ. If fewer elements than the budget have a positive gain, $\lambda = 0$ (Appendix D.3). If the always-selected elements alone reach the budget, $\lambda = \infty$ and the ratio can exceed ρ, which never occurred in the 22 LLaMA-2-7B quantizations whose per-layer λ we record.

Prior and two passes. The first pass uses the product prior $p(T_1{=}\pm1) = 0.235$, $p(T_1{=}0) = 0.53$, $p(T_2{=}\pm1) = 0.015$, $p(T_2{=}0) = 0.97$, which approximates an earlier design version's symbols. Its normalized joint-symbol counts become $\hat{p}$ for a second pass that restarts from the same rotated W and H and fixes the final mask; we have not measured the gain of this pass.

Entropy of T_2. With the second-plane elements split evenly between $T_2 = \pm1$, T_2 costs per weight

$$\mathcal{H}(\rho) = -(1-\rho)\log_2(1-\rho) - \rho\log_2\frac{\rho}{2} = h_{\mathrm{b}}(\rho) + \rho, \tag{D2}$$

with h_{b} the binary entropy: at most $\log_2 3 = 1.585$ bits ($\rho = 2/3$), and 0.161 bits at $\rho = 0.02$. Strictly, ρ in Eq. (D2) is the share with $T_2 \neq 0$, which the main loop keeps slightly below the realized ratio (1.947% against 2.000% at $\rho = 0.02$ in the sweep).

D.3 The control gsp-p and the granularity sweep of Figure 2(b)

Construction of gsp-p. The control ranks each linear layer's 128-column groups by their single-plane ATQ error $\sum_i \sum_{j\in g} h_j \varepsilon_{ij}^2$ and gives the top $\mathrm{round}(p \times \text{number of groups})$ nine levels on the 64 columns of largest $h_j \sum_i \varepsilon_{ij}^2$, with per-row INT8 (α, α_2, μ) fitted by an 8-step h_j-weighted Lloyd iteration. It skips the rate-distortion criterion and the second pass but shares rotation, compensation and calibration with ESPTQ-w. Like ESPTQ-w it writes no mask, yet its fixed-length accounting value charges one mask bit per column.

Conditions. Table D1 lists the 19 points of Figure 2(b) on LLaMA-2-7B, run on one RTX A5000 of another machine (Appendix G) with WikiText-2-only calibration (128 windows, seed 0; 24 logit windows, seed 101), the teacher on the CPU, no zero-shot tasks and the NousResearch copy of the weights, not checked to be bit-identical to Table 1's. The values therefore differ from Table 1 (7.44 against 7.51 at $\rho = 0.02$).

Table D1: The granularity sweep of Figure 2(b) on LLaMA-2-7B: (a) ESPTQ-w at ρ; (b) gsp-p ($p = 0$: no second plane). #W* in bits; perplexity at length 2048; rounded half up.

(a) esp: ρ	0.02	0.05	0.10	0.12	0.15	0.20	0.30	0.50	0.70	1.00
#W*	2.097	2.253	2.532	2.642	2.782	2.969	3.243	3.543	3.601	3.601
Wiki	7.44	6.93	6.66	6.60	6.52	6.37	6.19	5.87	5.78	5.78
C4	11.98	10.89	10.25	10.01	9.81	9.49	8.91	8.20	8.02	8.02
(b) gsp: p	0.00	0.05	0.10	0.20	0.30	0.40	0.50	0.70	1.00	
#W*	1.909	2.107	2.200	2.397	2.610	2.757	2.891	3.120	3.361	
Wiki	8.23	8.07	8.04	7.79	7.59	7.46	7.25	6.93	6.58	
C4	14.09	13.71	13.57	13.13	12.55	12.23	11.74	10.94	10.17	

What the sweep shows. At equal #W* (linear interpolation), element granularity has lower WikiText-2 and C4 perplexity than group granularity, although the gap also contains the selection rule and free α_2 of gsp-p. §5.1 cites $p = 0$ (8.23 at 1.91 bits) and $\rho = 0.02$ (7.44 at 2.10 bits), which stores about 0.30 bits less than gsp-0.2, here as in Table 1; gsp-p matches $\rho = 0.12$ only at $p = 1.00$, with 0.719 bits more. These gaps hold for #W* in our zlib format, which ignores the column structure of gsp-p: without entropy coding, gsp-0.2's fixed-length value (1.882 bits) is below ESPTQ-w's entropy-free layout at $\rho = 0.02$ (at most 1.99 bits; Appendix H), while the perplexity order holds. For $\rho \geq 0.70$, $\lambda = 0$ in all 224 linear layers, and $\rho = 0.70$ and 1.00 select the same 60.556% of the weights.

E COMPENSATION, CALIBRATION AND THE FULL ESPTQ-W PROCEDURE

This appendix supports §3.4 and the calibration set that §5 states.

E.1 GPTQ COMPENSATION

GPTQ (Frantar et al., 2023) quantizes the columns of W from left to right and cancels each column's error on the columns not yet quantized. Let U be the upper Cholesky factor of H_{d}^{-1}, with H_{d} the matrix H damped by 1% of its mean diagonal, and let group g span columns $i_1, \ldots, i_2 - 1$. After column j is rounded to q_j, with entries $\alpha_{ig}(T_{1,ij} + rT_{2,ij}) + \mu_{ig}$, the first part of Eq. (E1) updates the rest of the group; at the group's end, the second part updates all columns to its right, with $\mathcal{E}_g = [e_{i_1}, \ldots, e_{i_2-1}]$:

$$e_j = \frac{W_{:,j} - q_j}{U_{jj}}, \quad W_{:,k} \leftarrow W_{:,k} - e_j\, U_{jk} \;\; (j < k < i_2), \qquad W_{:,\,i_2:} \leftarrow W_{:,\,i_2:} - \mathcal{E}_g\, U_{i_1:i_2,\,i_2:}. \tag{E1}$$

Columns are quantized in their natural order, without act-order (Appendix I). For stability, a channel never excited by the calibration set gets diagonal 1 with its weights kept, and a failed Cholesky decomposition is retried on the CPU and with tenfold and hundredfold damping before a diagonal-only fallback; in the sweep of Appendix D, one layer per pass needed the CPU retry and none needed more.

E.2 TWO-STAGE CALIBRATION

Both stages move only ϕ_α, ϕ_μ of §3.4, starting from 0, with the ternary planes and the mask frozen. Table E1 gives their settings. Logit distillation differs in objective from EfficientQAT's end-to-end training of quantization step sizes on the next-token loss (Chen et al., 2025).

Table E1: Settings of the two calibration stages. Both run Adam (learning rate 10^{-3}, moments $(0.9, 0.999)$, no weight decay) for a fixed number of steps, without early stopping.

Item	Block-wise calibration	End-to-end logit distillation
Runs	after each decoder layer is quantized	once, after all decoder layers
Loss	mean squared error to the FP16 layer's output on the same input, the output of the quantized preceding layers	KL to the teacher over 257 categories (its top 256 tokens and the rest) plus the anchor $0.1\,\overline{\phi^2}$
Parameters	ϕ_α, ϕ_μ of the decoder layer	all ϕ_α, ϕ_μ; rotation angles frozen
Data and steps	the 128 calibration windows; 256 steps of one window	16 learning windows; 128 steps, each on one 512-token segment of one window
Afterwards	re-project to INT8 with new reference values; restore the layer if the error on the first 4 windows did not fall	re-project likewise; restore all weights if the KL on the 8 verification windows did not fall
In the sweep	80 of 608 layer calibrations restored	never rolled back

Calibration set. The 128 windows of 2048 tokens that feed the Hessians, the prepass and block-wise calibration are 96 from the WikiText-2 training split (Merity et al., 2017) and 32 from the English C4 training split (Raffel et al., 2020), at uniformly random offsets drawn with seeds 0 and 1. The 24 logit windows come from the same splits with seeds 101 (18 WikiText-2 windows) and 102 (6 C4 windows), and a permutation with seed 101 splits them into 16 learning and 8 verification windows, so that both sets contain C4. No WikiText-2 test text enters any stage; overlap between the logit windows and the 128 windows is not checked.

Teacher and numerics. The teacher is the unquantized model in its released 16-bit format (BF16, and FP16 for LLaMA-2), and its top-256 log-probabilities are computed once before training. Logit distillation holds ϕ and the KL in FP32, scales the loss by 1024 and skips steps with non-finite values; $\overline{\phi^2}$ is the mean square of each raw-parameter tensor, averaged over tensors. The teacher runs on the GPU when free memory suffices and on the CPU otherwise, and because the devices can sum in different orders, this placement can change the result. Table 1's runs used this automatic placement; the sweep of Appendix D and the ablation of Appendix I fix the CPU.

Guards and effect. The block-wise guard is measured in-sample; in the sweep it restored 1 to 8 of the 32 decoder layers per run, all within the first 14, for reasons we have not identified. At $\rho = 0.02$ in the sweep, logit distillation lowered the verification KL from 0.662 to 0.322 nats (0.330 after re-projection). We have not ablated the two stages in the final configuration. In design-stage runs with earlier code, block-wise calibration lowered WikiText-2 perplexity on all three models tested but raised LLaMA-3.2-3B's C4 perplexity by 10.5%, and logit distillation raised the seven-task zero-shot average by 1.47 points over five models at unchanged bits.

E.3 The full ESPTQ-w procedure

Algorithm 1 lists the procedure in the order of our code, and Figure E1 shows when each quantity becomes fixed. Beyond Figure 1, each linear layer runs the prepass and main loop twice from the same rotated weights (Appendix D), and block-wise calibration of decoder layer u ends before the Hessians of layer $u + 1$ are accumulated. Every Hessian therefore comes from quantized and calibrated preceding layers. Within a decoder layer, all Hessians and targets come from one forward pass with unquantized weights (no true-sequential option). Without the activation-flattening term of the rotation objective (arms A3h and A3w of Appendix I), the angles need no H and are all learned before the loop.

Algorithm 1 The full ESPTQ-w procedure for one model at a fixed assignment ratio ρ (Table 1 configuration).

Require: decoder layers $u = 1, \ldots, L$; 128 calibration windows $\mathcal{C}$; 16 learning and 8 verification logit windows; ρ
Ensure: per linear layer: joint symbols, INT8 $\hat{\alpha}, \hat{\mu}$, FP16 reference values $\sigma_\alpha, \sigma_\mu$ and angles

for $u = 1, \ldots, L$ **do** ▷ Z: embedding outputs on $\mathcal{C}$ for $u = 1$
  run layer u unquantized on Z; accumulate each linear layer's $H = X^\top X$; keep the outputs Y
  for each linear layer W of layer u **do**
    learn and fix the angles (§3.2); $W \leftarrow WR$, $H \leftarrow R^\top HR$; $\hat{p} \leftarrow$ prior
    for pass $= 1, 2$ **do** ▷ each pass starts from the rotated W
      prepass: ATQ per group (Appendix D) gives $\alpha, \mu, \varepsilon, \sigma_\alpha, \sigma_\mu$; fix M by Eq. (2)
      for groups g from left to right **do**
        ATQ on the compensated W_g; INT8 with $\sigma_\alpha, \sigma_\mu$; round by M; compensate by Eq. (E1)
      if pass $= 1$ **then** $\hat{p} \leftarrow$ empirical joint-symbol distribution
  block-wise calibration of layer u on (Z, Y); re-project; restore if the error did not fall
  $Z \leftarrow$ outputs of the quantized layer u
logit distillation of all ϕ; re-project; restore if the verification KL did not fall
write one zlib stream of joint symbols and INT8 codes per linear layer, the rest in FP16

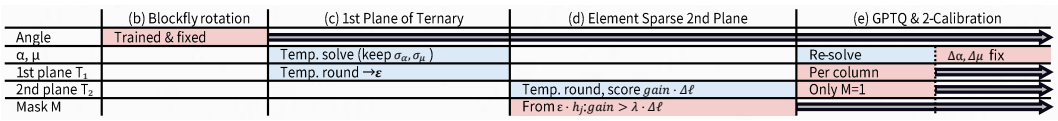


Figure E1: Stages (b)–(e) of Figure 1 at which each quantity of one linear layer is provisional (blue, Temp.) or fixed (red); Re-solve is the main loop's per-group re-solve and 2-Calibration the two-stage calibration. The second pass (Appendix D) repeats the stages from the same rotated weights and is not drawn.

F Ternary KV Cache: Derivations and Settings

This appendix supports §4.1–§4.4 and the KV-side calibration and re-rounding on which §5 and §5.2 rely. L is the number of layers, H_q and H_{kv} are the numbers of query and KV heads, and a channel of the KV cache is an index along the head dimension D.

F.1 HEAD DIMENSIONS AND THE FOUR CONDITIONS

Head dimensions. From each model's released config.json, $D = 64$ for SmolLM-135M, Qwen2.5-0.5B and LLaMA-3.2-1B, and $D = 128$ for the other ten models of Table 1 (all 13 are in Appendix H); in Table 2, only SmolLM-135M has $D = 64$. A per-token scaling factor and offset on K would cost $2/D$, twice the budget at $D = 64$, and a design that fits only $D = 128$ would be model-specific, so one design serves all D.

Why the four conditions. Without the budget condition, the comparison has no axis: we ask which design is the most accurate under the weight side's multiplier budget, not how few bits keep the accuracy. Without the storage-format condition, attention would need a second datapath beside the signed adder trees of the linear layers (§3.1), even for a format that is multiplication-free on its own, such as integer bit planes. Without the accumulator-output condition, an equal multiplier count would not suffice, because a multiplier that varies along the reduction dimension cannot be factored out of the adder tree, unless, like c, it is token-independent and absorbed into the weights (Appendix F.3). Without the write-side condition, only reading would be multiplication-free, although one token is written at every generation step; this condition forces the ternary reference of stage ③ and the threshold comparison of stage ⑤.

The offset. The weight side's offset, one μ per row and 128-column group of the reduction dimension, does not carry over. On V, it would be shared across tokens, and a per-token μ_t would need a multiplication in each term of $\sum_t P_t \mu_t$. On K, it becomes one offset per token, which the budget excludes at $D = 64$, where $\hat{\alpha}_t$ uses the one multiplication per reduction; at $D = 128$, such an INT8 offset would fit exactly ($2/128$), and we did not measure it.

Why only two sources. Sharing scaling factors across 128 tokens would hold up to 127 tokens of K in FP16, whose reads alone take 6.2% of the multiply-accumulates of $QK^\top$ at $N = 2048$, about four times the budget, and requantizing a written K_t would rewrite the cache.

F.2 CALIBRATION OF THE STATIC TABLES

The six static quantities of §4.1 are fixed after the ternary weights are frozen, by the stages of Table F1 (part (i) of Algorithm 2). The learning stages round on the deployed grid and pass only the choice of level straight through (STE; Bengio et al., 2013), so the scaling factor $\mathrm{amax}/1.65$ of K, taken at $\nu = 1$, keeps the gradient with respect to m from cancelling. c, $\hat{s}_v$ and γ are learned in the log domain, c within $[1/8, 8]$ with one variable per RoPE pair, and scaling factors stay unrounded until the reference values are set.

Teacher, headroom and windows. The block-wise inputs come from passes without KV quantization, so errors of earlier layers reach later layers only in end-to-end logit distillation, whose teacher keeps the ternary weights and the model's 16-bit KV format (FP16 for LLaMA-2-7B, bfloat16 otherwise). On all ten pairs of Table 2 whose logs we checked (all but Qwen2.5-3B), it lowered the verification KL, on LLaMA-2-7B at $\rho = 0.02$ from 0.1117 to 0.0662 nats per token. The initial windows coincide with the first eight WikiText-2 windows of the weight side (Appendix E): for the models of Table 2, we checked that both tokenizers return the same token sequence and, except on SmolLM-135M, that the start positions agree. No stage sees the WikiText-2 test split; we did not check text overlap among the windows or sweep learning rates, step counts, window counts or $\nu = 0.85$.

F.3 CHANNEL-WISE SKEW OF K AND ABSORBING c

Constant component. Figure F1 backs the skew stated in §4.1. We measured it on the released bfloat16 weights over the initial windows of Table F1, on the machine with an RTX 5070 Laptop GPU (Appendix G). With E the sum of the per-channel mean squares over a layer's 128 channels, $\|m\|^2/E$ has a median of 50.9% over the 24 layers, a maximum of 99.5% at layer 8 and a minimum of 29.3% at layer 11. The largest channel's share falls at layer 8 from 56.8% to 3.2% after subtracting m and 3.1% after dividing by c (initial value, bound 8), and its median over the layers from 15.2%

Table F1: The KV-side calibration in execution order; every run of Table 2 uses these values. All windows come from training splits.

Stage	Data	Setting
Initial values	8×2048 WikiText-2 tokens (seed 0), KV cache unquantized	m, m_v: channel means of K after RoPE and of V. c: channel absolute maxima of K, equalized within RoPE pairs, geometric mean 1 per head, clipped to $[1/\bar{c}, \bar{c}]$ with $\bar{c} \in \{1.5, 2, 3, 4, 6, 8, 12, 16\}$ chosen per head by the error of K on one 512-token window. $\hat{s}_v$: channel maxima of $\lvert V - m_v \rvert$ over the block-wise windows, divided by 1.65, times the best of $\{1.0, 0.8, 0.65, 0.5, 0.4, 0.3, 0.22, 0.16\}$ per KV head. $\gamma = 1$, $\delta = 0$
Block-wise calibration	first 512 tokens of the first four initial windows	per decoder layer, mean squared error of the attention output before W_o to the unquantized one; moves $m, c, m_v, \hat{s}_v$; Adam (Kingma & Ba, 2015), learning rate 0.02, moments $(0.9, 0.999)$, 300 steps on all four windows, on the CPU
End-to-end logit distillation	12×1024 tokens: nine WikiText-2 (seed 202), three C4 (seed 203); eight learning and four verification windows (shuffle seed 202)	KL over 257 categories to the ternary-weight model with an FP16 KV cache; moves all six; Adam, learning rate 0.001, at most 50 steps; verification every 5 steps, stop after 3 without improvement, keep the best tables (the block-wise ones if none improves)
Reference values	2×2048 WikiText-2 tokens (seed 0), quantization on	largest grid step times $1.25/127$; σ per layer and KV head; one per layer for each of $\hat{s}_v$ and $\hat{s}_r$
Absorbing c	–	rows and bias of W_q times c, of W_k times $1/c$; their reference values re-measured and α, μ re-rounded to INT8; $m \leftarrow m/c$; σ kept

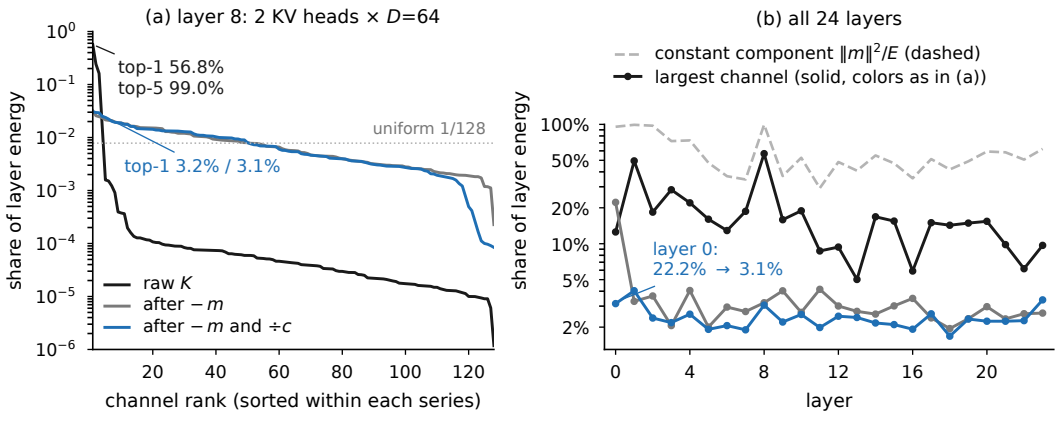


Figure F1: Channel-wise energy (per-channel mean square) of K after RoPE in the unquantized Qwen2.5-0.5B over 8×2048 WikiText-2 training tokens: raw (black), after subtracting m (gray) and after dividing by the initial c (blue). (a) Layer 8, channel shares in descending order, uniform share $1/128$ dotted; (b) per layer, the largest channel's share (solid) and the constant component $\|m\|^2/E$ (dashed).

to 2.8% and 2.3%; c lowers it in 20 of the 24 layers. Formed before m is subtracted, c also shrinks four channels of layer 8 to about $1/80$ of the uniform share (effect on accuracy not measured).

Cause. The bias of W_k carries the constant component at layers 0–2 and 8 ($\|b_k\|^2$ is 99.4–102.4% of the energy before RoPE) and survives RoPE in the lowest-frequency pairs, which turn by at most 0.0049 rad over 2048 positions. Of our models, only the Qwen2.5 models and Qwen1.5-4B have this bias; we did not measure the others.

Absorbing c and the commutation condition. With u the layer input, $\big(\sum_j W_k[d,j]u_j + b_k[d]\big)/c_d = \sum_j (W_k[d,j]/c_d)\, u_j + b_k[d]/c_d$, so dividing row d of W_k and its bias by c_d makes W_k output K/c, and the rows of W_q get the c of their KV head. Because c enters before RoPE and K is quantized after it, $\mathrm{diag}(c)$ must commute with the 2×2 rotation of each rotate-half pair $(d,\, d + D/2)$, which holds only if $c_d = c_{d+D/2}$; calibration equalizes each pair initially and then learns one variable per pair. RoPE would rotate a bias $-m$ with the position, so m/c stays a table subtracted at write time. Nothing between W_k and RoPE may mix channels or normalize a head; Qwen3's per-head RMSNorm on keys would need c in its weight, which our code does not support, and no model of Table 2 has one.

Tables left at run time. After absorption, c becomes 1, m becomes m/c and the reference values are kept. The run-time tables are m/c and m_v (FP16) and $\hat{s}_v$ (INT8) per KV head and channel, $\hat{s}_r = \hat{s}_v\gamma$ (INT8) per query head and channel, and the bias of W_o, which absorbs $m_v + \delta$ in the circuit model; the evaluated code adds $m_v + \delta$ to the input of W_o, which is algebraically identical.

Re-rounding. Scaling a row of the ternary W_q or W_k moves its α and μ off the INT8 grid, so we re-measure their reference values and re-round them; the ternary planes, the fixed-length accounting value and the multiplier count do not change. On SmolLM-135M at $\rho = 0.02$ with seed 0, absorbing c changed WikiText-2 perplexity by -0.083% and re-rounding by a further $+0.146\%$, $+0.062\%$ in total, below 1% of the 16.7% gap between its ESPTQ-w and ESPTQ-wkv rows at $\rho = 0.02$ in Table 2. This single-seed run used the RTX 5070 Laptop machine and a KV-side configuration that may differ from the final one, and we did not measure $D = 128$. Table 2's #W* is recorded before absorption; re-rounding keeps the number of INT8 codes and could change #W* by at most about 0.021 bits.

Relation to prior work. SmoothQuant (Xiao et al., 2023) absorbs per-channel factors into adjacent weights, QServe's SmoothAttention (Lin et al., 2025) (their Section 4.2) already absorbs such factors into W_q and W_k with equal factors on RoPE pairs, and RotateKV (Su et al., 2025) finds that such a transform alone does not keep accuracy at 2 bits (their Table 1). The absorption and the commutation condition are therefore not ours; we add learning c in the two-stage calibration and measuring the cost of re-rounding it into ternary weights.

F.4 Read and write paths

Read path. Table F2 lists the multipliers. For PV, one decode step at $N = 2048$ gives $1/N = 0.049\%$, and a 2048-token prefill gives $2/(N+1) = 0.098\%$. Absorbing σ into W_q would leave $\hat{\alpha}_t$ as the multiplicand and scale every $J(\nu)$ by σ^2, leaving ν^* unchanged. The evaluated code absorbs neither σ nor $m_v + \delta$ and computes attention in floating point from the reconstructions of Eq. (3), which the plane splits only reorder.

No per-token scaling factor on V. KIVI (Liu et al., 2024) quantizes V per token (Appendix A). A per-token factor α_t on V would give

$$O_{q,d} = \sum_t P_{q,t}\, \alpha_t\, \Lambda^v_{t,d}. \tag{F1}$$

In Eq. (F1), α_t varies along the reduction and cannot leave the sum, as Duan et al. (2026) also note, so it must be pre-multiplied into P. These N multiplications per query and head fit the budget by count ($1/D$) but sit before the adder tree, not at the accumulator output where the K side and the weight side multiply; the static $\hat{s}_r$ and m_v sit there instead.

Write path. Algorithm 2 gives the procedure. Stage ③ builds y_{ref} from $s_1\Lambda_1$ in two passes; pass i takes $\beta_i = \max_d |e[d]|$ of the residual $e = x - y_{\text{ref}}$ and adds $0.675\beta_i T_{i+2}$ with $T_{i+2}[d] = \text{sign}(e[d])\,[\,|e[d]| > 0.35\beta_i\,]$. The values 0.35 and 0.675 and the two planes were chosen at design time by a comparison of attention-distribution KL that we do not report. Since $y_1 = s_1\Lambda_1$ is the first term of y_{ref} and $s_\nu = \nu\,\text{amax}/1.65$, with A_η and B_η as in §4.3,

$$\begin{aligned} Q'_\eta \cdot (y_1 - y_{\text{ref}}) &= -B_\eta, \\ Q'_\eta \cdot (y_{0.85} - y_{\text{ref}}) &= \text{amax}\, A_\eta - B_\eta, \\ J(0.85) - J(1) &= \textstyle\sum_\eta \big[(\text{amax}\, A_\eta - B_\eta)^2 - B_\eta^2\big] = \text{amax} \sum_\eta A_\eta \big(\text{amax}\, A_\eta - 2B_\eta\big). \end{aligned} \tag{F2}$$

As $\text{amax} > 0$ for any nonzero token, Eq. (F2) makes the test of §4.3 equal to $J(0.85) < J(1)$; a tie gives $\nu^* = 1$, and the evaluated code compares the two J values directly. The four reductions run on six signed adder trees and need four run-time products, which keeps the ratio at $1/D$ but adds $4H_q/H_{kv}$ reductions per token and KV head (28 on Qwen2.5-7B). Stage ③ places the candidates with the unrounded step $s_\nu\tau$, so near a threshold, ⑤ can store another point. The writing token's own query stands in for the future queries; how well they agree was not measured.

Why two candidates, $\nu \in \{1.00, 0.85\}$. The value 0.85 was chosen at design time; we report no sweep of it alone. Tightening every token collapsed in an earlier configuration, which also quantized V per token, lacked the end-to-end stage, γ, δ and stage ③, multiplied K by c instead of absorbing it and used older weight code: there $\nu = 0.9$ raised the WikiText-2 perplexity of Qwen2.5-0.5B from 29.16 to 19,404 (seed 0). Its cause is not established: block-wise calibration itself failed to lower its loss at $\nu = 0.8$, and we did not test which change prevents the collapse. In design-stage runs with older KV code and seed 0, adding stage ③ changed WikiText-2 perplexity by -2.25% to -4.12% on five models, comparable to a spread of 2.75–2.82% over three seeds; a four-candidate variant was not adopted, because its gains changed sign across seeds.

Stages ⑤ and ⑥. The evaluated code divides and picks the nearest level, which equals ⑤ except on a threshold or within one rounding step of it, a fraction we did not measure. The thresholds $\hat{s}_v\tau$ of V are formed once per model and not stored, and ⑥ discards $T_3, T_4, \beta_1, \beta_2, \text{amax}$ and ν^*. Part (iii) of Algorithm 2 shows the circuit model, with σ absorbed into W_q.

Algorithm 2 The full ESPTQ-kv procedure: (i) before deployment (settings in Table F1), (ii) the write of token t and (iii) the read for one query.

(i) Before deployment (weights already ternary, Algorithm 1)
initialize $m, m_v, c, \hat{s}_v$ as in Table F1; $\gamma \leftarrow 1$, $\delta \leftarrow 0$
block-wise calibration of $m, c, m_v, \hat{s}_v$, one decoder layer at a time
end-to-end logit distillation of all six; keep the tables with the lowest verification KL
reference values: largest grid step over two windows, times $1.25/127$
absorb c into W_q, W_k and re-round their α, μ to INT8; $m \leftarrow m/c$
(ii) Write token t (per layer and KV head; $Q'_\eta = Q_{t,\eta} \odot c$)
① $x \leftarrow K_t/c - m/c$ ▷ K_t/c: output of W_k after RoPE
② $\text{amax} \leftarrow \max_d |x[d]|$
③ $\nu^* \leftarrow 0.85$ if $\sum_\eta A_\eta(\text{amax}\, A_\eta - 2B_\eta) < 0$, else $\nu^* \leftarrow 1$ ▷ Eq. (F2)
④ $\hat{\alpha}_t \leftarrow \text{round_int8}\big(\text{amax}\,\nu^*/(1.65\,\sigma)\big)$
⑤ find Λ_t^k by comparing $|x[d]|$ with $\hat{\alpha}_t\sigma\tau$; ⑥ write $T_1, T_2, \hat{\alpha}_t$
for V: ① $x \leftarrow V_t - m_v$; ⑤ compare with $\hat{s}_v\tau$; ⑥ write T_1, T_2
(iii) Read (per query head; $\tilde{Q} = \sigma\, Q \odot c$)
$S_t \leftarrow \hat{\alpha}_t\big(\tilde{Q} \cdot T_{1,t} + r\,\tilde{Q} \cdot T_{2,t}\big)$ for each cached token t ▷ $Q \cdot m$ cancels
$P \leftarrow \text{softmax}(S/\sqrt{D})$
$O \leftarrow \hat{s}_r \odot \big(\sum_t P_t T_{1,t} + r \sum_t P_t T_{2,t}\big)$; the bias of W_o adds $W_o(m_v + \delta)$

F.5 Multiplier and bit accounting

Table F2 lists every fully general multiplier on the KV side; with no divider on the write path, the KV-side ratio is at most $1/D$ for $N \geq D$. As on the weight side (Appendix B), products with design-time constants, including each datapath's σ and $\sigma\tau$, count as shifts and adds, softmax, RoPE at write

Table F2: Fully general multipliers on the KV side. Length is the number of multiply-accumulates that one reduction sums; "–" marks rows without a reduction.

Path	Length	Multipliers	Ratio	Reason
$QK^\top$, ternary term	D	1 ($\hat{\alpha}_t$)	$1/D$	two signed adder trees; the product with r is shift-and-add
$QK^\top$, $Q \cdot m$	–	0	0	common to a query row, cancels in the softmax
PV, ternary term	N	1 ($\hat{s}_r$)	$1/N$	two signed adder trees
PV, $\sum_t P_t m_v$	–	0	0	equals m_v; absorbed with δ into the bias of W_o
Write ③, four test reductions	D each	4	$1/D$	products with β_1, β_2, amax and A_η
Write ①, ②, y_{ref} of ③, ④, ⑤	–	0	0	subtractions, comparisons and products with constants

time, normalization and calibration lie outside the datapath, and lookups and base-9 unpacking are not multiplications. The product with $\hat{s}_r$, which varies over channels, counts, as the weight side's α and μ do.

The case $N < D$. Early in generation, the per-step ratio $1/N$ of PV exceeds $1/D$, and for $N < 64$ the budget, with no multiplier added; only the denominator $N \cdot D$ is small. At $D = 64$, the queries at positions $i < D$ hold 0.099% of the multiply-accumulates of PV in a 2048-token prefill. The bound of §4.4 therefore assumes $N \geq D$.

Stored bits. Each joint symbol is stored as the index 0–8 of its level in ascending order, five per 2-byte word $\sum_i u_i 9^i$ ($9^5 = 59{,}049 \leq 2^{16}$), so a token and KV head take $\lceil D/5 \rceil$ words (26 bytes at $D = 64$, 52 at $D = 128$), plus 1 byte of $\hat{\alpha}_t$ for K. Per element, K takes 3.375 bits and V 3.25 bits at $D = 64$, and 3.3125 and 3.25 bits at $D = 128$. Table 2's 3.29 bits average these values over the six models of each ESPTQ-wkv row, one with $D = 64$ and five with $D = 128$: $(3.3125 + 5 \times 3.28125)/6 = 3.2865$. Unlike #W*, #KV* counts this fixed-length format, not zlib output.

Bit-exact check. Evaluation holds the nine-level reconstructions in floating point (pseudo-quantization), which saves no memory. We therefore ran the first 512 tokens of the WikiText-2 test split with the cache enabled, packed the selected indices and INT8 $\hat{\alpha}_t$ of every layer, unpacked them and compared the rebuilt K and V bit for bit with the model's cache. They matched in every layer on all 12 ESPTQ-wkv model-ρ pairs of Table 2 and on two SmolLM2-360M pairs. We did not pack the evaluation windows themselves, and the recorded #KV* is computed from the format, not measured.

Static tables. With $E_{kv} = LH_{kv}D$ and $E_q = LH_qD$, the tables of Appendix F.3 take $5E_{kv} + 3E_q$ bytes: FP16 for m/c, m_v and the bias of W_o (length H_qD here), INT8 for $\hat{s}_v$ and $\hat{s}_r$, and nothing for c. On Qwen2.5-0.5B ($L = 24$, $H_{kv} = 2$, $H_q = 14$, $D = 64$), this is 79,872 bytes, or 78.0 KiB, computed analytically and equal to about 31 tokens of packed cache; on LLaMA-2-7B, it is 1,024 KiB. Our result files record a larger size, which counts all six tables in FP16.

Out of scope. Storage is fixed-length, so an element with $T_2 = 0$ takes the same field as any other, and skipping its multiply-accumulates would change only switching activity, which we did not measure. The ratios above are analytical counts in our circuit model; we built no circuit and did not measure latency, area or energy (Appendix J).

G Experimental Setup and Reproducibility

This appendix supports §5 and the §5.1 footnote.

G.1 MACHINES, SOFTWARE, MODELS AND CODE

Machines. Table G1 lists our four machines, named by GPU. The A6000 machine also ran Ubuntu 22.04.5, lm-evaluation-harness 0.4.12 and datasets 5.0.0.

Table G1: Machines, software versions (PyTorch with CUDA build; n.r.: not recorded) and results; transformers is 5.14.1 throughout.

Machine	Python	PyTorch	NumPy	Results
RTX A6000 (48 GB)	3.13.11	2.13.0 (12.6)	2.5.1	Table 1; Figure 4; Table 2's 7B models; Appendix B; parts of Appendices H and I
Two RTX A5000 (24 GB each)	3.13	2.7.1 (11.8)	2.5.1	Figure 2(b); Table 2's SmolLM-135M; parts of Appendices F, H and I
RTX 4500 Ada (24 GB)	3.10	2.13.0 (13.0)	2.2.6	Table 2's Qwen1.5-4B and LLaMA-3.2-3B; part of Appendix F
RTX 5070 Laptop GPU	n.r.	2.13.0 (13.0)	n.r.	Figure 2(a); Appendix C; part of Appendix F

Comparisons stay within one table. We compare values only within one table or figure, and within Table 2 only within one model, because runs differ across machines and run sets: PT2-LLM's WikiText-2 perplexity on SmolLM-135M, for example, moves by -23.2% from Table 1 to Table 2. Table 2 ran SmolLM-135M on the RTX A5000 machine and Qwen1.5-4B and LLaMA-3.2-3B on the RTX 4500 Ada machine. On LLaMA-2-7B and Qwen2.5-7B, its FP16, ESPTQ-w, PT2-LLM and TWLA perplexity and 0-shot[8] are copied from Table 1's runs, whose code version and zero-shot batch size (Appendix G.2) differ from those of the ESPTQ-wkv runs; only their GSM8K was measured anew, with Table 1's settings.

Numerical settings. The harness runs PyTorch and cuDNN deterministically without TF32 and sets `CUBLAS_WORKSPACE_CONFIG` as each public code does: `:16:8` for TWLA and the default-equivalent `:4096:8` otherwise. In a float16 development run on SmolLM-135M, the other value would lower WikiText-2 perplexity by 17.8% for PT2-LLM and 5.3% for TWLA. Its effect on Table 1's models is unmeasured, so ESPTQ's ($\rho = 0.12$) WikiText-2 leads of 0.69–5.20% over TWLA on the LLaMA-2 and the 7B and 14B Qwen2.5 models are not established under TWLA's other value.

Models and code. Models are the Hugging Face base checkpoints (Qwen3 as `-Base`; Qwen1.5-4B, Bai et al., 2023; SmolLM-135M, Ben Allal et al., 2024; SmolLM2-360M, Ben Allal et al., 2025) in their released precision, float16 for LLaMA-2 and bfloat16 otherwise; QuIP, our PTQTP reproduction and the activation comparison of Appendix I run in float16. Every method uses the same tokenizer, checked by hash.

G.2 EVALUATION PROTOCOL

Perplexity. WikiText-2 perplexity (Merity et al., 2017) joins the raw test split with blank lines, tokenizes it once and cuts it into non-overlapping 2048-token windows, dropping the remainder; transformers (Wolf et al., 2020) computes the loss in FP32, and window losses are summed in double precision. C4 perplexity (Raffel et al., 2020) follows GPTQ's code (Frantar et al., 2023): 256 windows of 2048 tokens from the first 1100 documents of the first English validation shard.

Zero-shot tasks. lm-evaluation-harness (Gao et al., 2023; Biderman et al., 2024) evaluates every item with zero shots. ARC-e, ARC-c (Clark et al., 2018), PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019) and OpenBookQA (Mihaylov et al., 2018) report acc_norm, the accuracy after dividing each choice's log-likelihood by its length in characters; WinoGrande (XL; Sakaguchi et al., 2020), BoolQ (Clark et al., 2019) and LAMBADA (`lambada_openai`; Paperno et al., 2016) report acc. 0-shot[8] is the unweighted mean of the eight recorded accuracies (percent, two decimals).

GSM8K. The task `gsm8k_cot` (Cobbe et al., 2021) prepends eight fixed chain-of-thought examples (Wei et al., 2022), decodes greedily up to 256 tokens and scores the number N in "The answer is N" by exact match.

Activations, KV cache and batch size. Activations stay in 16-bit floating point, and only the ESPTQ-wkv rows quantize the KV cache. The zero-shot batch size was automatic in Table 1's runs and 4 in the runs made for Table 2. Its effect is unmeasured, so on the two 7B models the 0-shot[8] cost of the KV stage in §5.2 contains a batch-size effect of unknown size; perplexity, evaluated one window at a time, is unaffected.

G.3 Default settings, calibration and seeds

Table G2 lists the defaults of every ESPTQ-w result unless stated otherwise; ESPTQ-kv's are in Appendix F.

Table G2: Default settings of ESPTQ-w and the appendix that defines each.

Setting	Default	App.
Group size G	128 input channels	B
Inter-plane ratio r	0.65, a design-time constant	B
Assignment ratio ρ	0.02 and 0.12	D
Scaling factors, offsets	symmetric INT8 per row and group; FP16 reference value per linear layer	B
blockfly	7 stages per 128 columns from angles $\pi/4$; Adam, learning rate 0.01, 100 iterations	C
blockfly objective	TriGMM NLL (width 0.25κ) $+ 0.1\times$ occupancy $+ 1.0\times$ flattening	C
Prepass	ATQ (at most 10 ITF iterations); λ per linear layer meeting ρ; two passes	D
GPTQ	damping 1% of the mean diagonal of H; original column order	E
Calibration	Adam, learning rate 10^{-3}; 256 block-wise and 128 logit-distillation steps; anchor 0.1	E
Teacher placement	GPU if its free memory exceeds $1.1\times$ the teacher plus 2 GB, else CPU	E
Calibration windows	$96 + 32$ and, for logit distillation, $18 + 6$ WikiText-2 + C4 windows of 2048 tokens	§5
Seeds	base seed 0 (C4 windows $+1$; logit-distillation windows $+101$, $+102$)	G.3

Calibration sets. The baselines keep their papers' sets (Table G3), so WikiText-2 perplexity compares methods calibrated with and without WikiText-2 text, and our set also contains C4. Only Appendix I matches the calibration set, for ESPTQ and TWLA on WikiText-2 alone.

Seeds, decision margin and noise. Every result is a single run with base seed 0. We measured seed variation only under the ablation conditions of Appendix I (LLaMA-2-7B, one RTX A5000, calibration on WikiText-2 alone, $\rho = 0.12$): base seeds 0, 1 and 2 give WikiText-2 perplexities of 6.6040, 6.6210 and 6.6376, a standard deviation σ of 0.254% of the mean (C4: 0.296%), with #W* within 0.0004 bits. The difference of two runs then has standard deviation $\sqrt{2}\,\sigma$, and the decision margin of §5.1 is twice that, 0.72%. With three seeds it is not robust: the t quantile 4.30 (two degrees of freedom) in place of 2 would give 1.54%. We did not measure the seed variation of zero-shot accuracy; as lower bounds on its noise, reruns of identical settings agree to four decimals in perplexity and #W* yet differ by up to 0.20 points per task, and the same weights evaluated under PyTorch 2.5.1 and 2.13.0 differ by up to 0.40 points per task.

G.4 Baselines: settings, reproduction and reported values

Settings. Each baseline runs its authors' public code (Table G3) with the settings its paper declares, changed only for our harness's interface, newer libraries and one in-place Hessian modification in SliM-LLM. The one deliberate deviation, GPTQ's static groups, lowered the reconstruction error of all 12 projections we checked on SmolLM-135M by 25.2% in total.

Skipped conditions. As §5 states, conditions the upstream code does not support stay empty instead of counting as failures; QuaRot's code silently mis-rotates models with tied embeddings, so our harness refuses them. PB-LLM's #W of 1.70 is the value later work reports (Huang et al., 2024; Li et al., 2025; Yan et al., 2026); its own count, with a bitmap of one bit per weight for its example of 10% INT8 weights, is at most 2.7 bits (Shang et al., 2024).

Table G3: Baseline settings. #W is the nominal bit width (Table 1, or the appendix for AWQ, QuaRot and SliM-LLM); calibration uses 128 training windows of 2048 tokens, except [a] 128 Pile (Gao et al., 2020) validation documents of at most 512 tokens cut into 512-token blocks.

Method (#W)	Storage format	Settings; models not supported	Calibration
GPTQ (2.00)	asymmetric 2-bit, group 128	act-order, true-sequential, static groups (off upstream), damping 1%	C4
QuIP (2.00)	2-bit LDLQ after incoherence processing	no greedy passes (not QuIP-RG); float16	C4
AWQ (2.25)	2-bit, group 128	upstream supports neither SmolLM nor Qwen3	Pile[a]
QuaRot (2.00)	asymmetric 2-bit per channel after randomized Hadamard; GPTQ	runs only on LLaMA-2-7B, LLaMA-2-13B and LLaMA-3.1-8B	WikiText-2
SliM-LLM (2.00)	salience-allocated bits per 128 columns	in-place Hessian update fixed; Cholesky fails on Qwen2.5-7B	WikiText-2
PB-LLM (1.70)	binary; 10% salient weights in INT8	not on SmolLM-135M (576 is no multiple of its block 128)	C4
BiLLM (1.08)	binary, residual on salient columns	Hessian salience; block 128	C4
ARB-LLM (1.08)	ARB-LLM$_{\mathrm{RC}}$	Hessian salience; block 128	C4
PT2-LLM (1.58)	asymmetric ternary (ATQ); SSR	ATQ with SSR as its README runs it; damping 1%	WikiText-2
TWLA (1.58)	ternary (E2M-ATQ), second plane on salient columns; KOTMS	its defaults; activations not quantized	WikiText-2
PTQTP (1.58)	two dense ternary planes	no public code; our reimplementation, float16	none
CAT-Q (1.58)	ternary, one FP16 scale per 128 weights	no quantization code; public LLaMA-2-7B weights	–

Reported values. On LLaMA-2-7B, with FP16 equal to the papers', our PT2-LLM and TWLA reach WikiText-2 perplexities of 11.73 and 7.02 against their reported 11.56 and 6.97. PT2-LLM's table reports GPTQ at 52.22 and QuIP at 39.73, where ours give 29.23 and 82.92, for reasons we have not found. Our PB-LLM collapses on LLaMA-2-13B (268.36), as in TWLA's table (335.22). TWLA also reports PT2-LLM's seven-task mean on LLaMA-2-13B as 56.54, against our 0-shot[8] of 46.04; its reported accuracy is higher than ours by 7.33 to 17.44 points on each of the six shared tasks, a gap we cannot explain.

PTQTP. PTQTP has no public code, so we reimplemented it and ran it on the A6000 machine. With FP16 equal within 0.01, our WikiText-2 perplexities, 9.03 on Qwen2.5-7B and 14.19 on LLaMA-3.1-8B, exceed the published 8.31 and 8.53, at 3.903 and 3.910 stored bits. Because this reproduction fell short, Table 1's PTQTP row shows only published values (arXiv version 1), for models whose FP16 perplexity in PTQTP is within 0.01 of ours: LLaMA-2-7B and LLaMA-2-13B from Table 9 of PTQTP, and LLaMA-3.2-1B and Qwen2.5-7B from its Table 1.

CAT-Q. CAT-Q releases weights but no quantization code, and only its LLaMA-2-7B weights fit our model set, as its Qwen3 weights are instruction-tuned; exported to 16 bits, every group of their 224 linear layers holds exactly three values. We evaluated them on the A6000 machine with our protocol. On CAT-Q's five tasks, our mean is 57.85 against its reported 57.69, with no task differing by more than 0.75 points; Table 1 shows our 0-shot[8] and WikiText-2 perplexity instead, because that mean covers other tasks.

H Full Results

This appendix supports §5.1 and §5.2, and TWLA's effective bits quoted in §1.

H.1 Weight-only quantization on all 13 models

Setting. Tables H1 and H2 extend Table 1 to all 13 models and add C4 perplexity and MMLU (0-shot). Every value is a single fixed-seed run on the A6000 machine, evaluated as in Appendix G; the FP16 values come from a separate result file that records no environment. ESPTQ$_{0.02}$ and ESPTQ$_{0.12}$ are Table 1's ESPTQ-w rows.

Table H1: Weight-only quantization on 13 models: (a) effective bits #W*, (b) WikiText-2 and (c) C4 perplexity at sequence length 2048. †: published values; ‡: our evaluation of CAT-Q's public weights; ◇: SliM-LLM before a code change; –: no value.

Model	FP16	GPTQ	QuIP	SliM-LLM	PB-LLM	BiLLM	ARB-LLM	PT2-LLM	PTQTP†	CAT-Q‡	TWLA	gsp-0.2	**ESPTQ$_{0.02}$**	**ESPTQ$_{0.12}$**
#W	16	2.00	2.00	2.00	1.70	1.08	1.08	1.58	1.58	1.58	1.58	1.58	1.58	1.58
*(a) Effective bits #W**														
SmolLM-135M	16	2.021	1.885	2.554◇	–	3.419	2.922	2.121	–	–	5.460	2.489	2.168	2.699
Qwen2.5-0.5B	16	1.952	1.872	2.487	2.546	3.329	2.776	2.063	–	–	5.308	2.391	2.152	2.695
LLaMA-3.2-1B	16	1.982	1.866	2.551	2.497	3.341	2.744	2.061	–	–	5.457	2.413	2.112	2.659
Qwen2.5-1.5B	16	1.972	1.864	2.443	2.515	3.348	2.751	2.059	–	–	5.374	2.392	2.127	2.670
Qwen3-1.7B	16	1.987	1.872	2.422	2.507	3.310	2.737	2.059	–	–	5.239	2.427	2.128	2.668
Qwen2.5-3B	16	1.942	1.870	2.366	2.477	3.287	2.705	2.011	–	–	5.274	2.395	2.106	2.641
LLaMA-3.2-3B	16	1.996	1.867	2.450	2.491	3.300	2.728	2.062	–	–	5.437	2.438	2.104	2.652
Qwen3-4B	16	2.002	1.870	2.492	2.476	3.528	2.817	2.050	–	–	5.408	2.413	2.113	2.649
LLaMA-2-7B	16	2.006	1.868	2.552◇	2.475	3.380	2.756	2.062	–	1.946	5.458	2.397	2.097	2.644
Qwen2.5-7B	16	1.982	1.867	–	2.486	3.295	2.719	2.046	–	–	5.439	2.435	2.102	2.636
LLaMA-3.1-8B	16	1.997	1.865	2.533	2.483	3.242	2.706	2.060	–	–	5.441	2.396	2.095	2.642
LLaMA-2-13B	16	2.006	1.861	2.543	2.472	3.418	2.767	2.061	–	–	5.601	2.415	2.096	2.642
Qwen2.5-14B	16	1.973	1.865	2.382	2.468	3.266	2.708	2.031	–	–	5.520	2.414	2.097	2.637
Mean	16	1.986	1.869	2.481◇	2.491	3.343	2.757	2.057	–	–	5.416	2.416	2.115	2.656
(b) WikiText-2 perplexity														
SmolLM-135M	17.56	1.85e4	2.26e3	277.07◇	–	5.98e3	313.91	260.75	–	–	58.93	46.81	43.54	31.08
Qwen2.5-0.5B	13.08	2.10e3	1.08e3	806.96	267.70	3.84e3	462.24	325.59	–	–	26.62	26.27	23.23	19.33
LLaMA-3.2-1B	9.76	1.62e3	1.06e3	374.64	203.10	1.76e3	136.65	78.91	(17.15)	–	19.62	20.30	19.24	15.67
Qwen2.5-1.5B	9.27	575.36	158.24	50.16	344.42	945.72	253.53	247.40	–	–	14.94	15.35	14.74	12.57
Qwen3-1.7B	9.40	150.09	124.47	77.22	420.91	5.12e3	289.02	66.07	–	–	20.42	14.61	14.29	12.33
Qwen2.5-3B	8.03	149.60	54.71	32.63	53.55	166.30	39.94	95.24	(13.13)	–	11.34	12.26	11.88	10.39
LLaMA-3.2-3B	7.82	773.95	213.06	183.65	115.42	163.49	37.26	42.00	(10.24)	–	12.04	13.17	12.56	10.63
Qwen3-4B	7.90	82.78	47.61	16.42	41.85	205.34	25.41	38.82	–	–	10.82	11.19	10.97	9.71
LLaMA-2-7B	5.47	29.23	82.92	32.25◇	74.99	32.10	15.97	11.73	(6.32)	11.70	7.02	7.88	7.51	6.66
Qwen2.5-7B	6.85	23.84	17.17	–	30.71	29.50	17.81	55.39	(8.31)	–	8.60	9.21	9.01	8.18
LLaMA-3.1-8B	6.24	123.41	89.16	33.83	62.55	44.24	25.50	33.48	(8.53)	–	9.33	10.37	9.81	8.51
LLaMA-2-13B	4.88	12.40	15.67	9.09	268.36	22.87	12.11	9.81	(5.29)	–	5.76	6.52	6.29	5.72
Qwen2.5-14B	5.29	22.59	16.73	13.70	28.28	23.38	14.66	18.85	–	–	7.43	8.06	7.98	7.05
(c) C4 perplexity														
SmolLM-135M	22.17	8.04e3	1.17e3	593.38◇	–	7.49e3	296.39	506.66	–	–	118.17	73.14	65.11	44.58
Qwen2.5-0.5B	20.39	1.22e3	1.04e3	1.90e3	241.43	3.88e3	438.64	927.71	–	–	54.90	47.30	41.71	32.57
LLaMA-3.2-1B	14.02	1.04e3	796.85	956.48	237.29	822.88	128.11	275.43	–	–	36.09	36.13	33.72	26.22
Qwen2.5-1.5B	15.13	393.43	144.87	225.93	363.14	900.29	246.07	979.44	–	–	27.57	27.80	26.42	21.56
Qwen3-1.7B	15.42	121.69	138.57	351.17	348.46	4.50e3	248.21	306.67	–	–	41.81	26.59	25.61	21.40
Qwen2.5-3B	13.36	105.48	64.84	95.43	61.16	144.34	53.51	351.04	–	–	20.29	21.84	21.03	17.84
LLaMA-3.2-3B	11.34	193.11	150.86	636.96	107.09	122.04	45.90	167.95	(15.28)	–	21.36	23.21	21.70	17.76
Qwen3-4B	13.31	77.59	53.06	37.27	52.05	156.26	33.37	158.51	–	–	19.53	20.12	19.60	16.87
LLaMA-2-7B	7.26	28.54	67.77	36.86◇	87.34	37.86	19.51	24.77	(8.41)	12.17	10.38	12.18	11.42	9.72
Qwen2.5-7B	11.88	28.77	22.04	–	39.06	34.95	24.61	216.91	–	–	15.31	16.65	16.22	14.37
LLaMA-3.1-8B	9.54	107.16	77.65	209.92	99.83	52.43	33.54	155.04	(13.22)	–	16.90	18.74	17.46	14.68
LLaMA-2-13B	6.73	14.68	18.75	17.92	222.82	25.76	14.40	21.21	(7.34)	–	8.55	10.23	9.73	8.46
Qwen2.5-14B	10.35	25.74	21.49	26.72	30.02	23.73	19.80	75.66	–	–	13.21	14.73	14.17	12.64

Marks and empty cells. SliM-LLM (Huang et al., 2025b), absent from Table 1, allocates 1–3 bits per 128-column group, 2 on average. PTQTP's values come from Tables 1, 7 and 9 of its arXiv v1 (Xiao et al., 2025). PB-LLM cannot run on SmolLM-135M (hidden size 576, not a multiple of 128), and SliM-LLM's Cholesky decomposition fails on Qwen2.5-7B, whose Hessian is not positive definite. The ◇ runs predate our change that stops SliM-LLM's allocation search from adding 0.01 to the Hessian diagonal in place; a LLaMA-2-7B run unaffected by this change gave 16.73 (reported: 16.01), so these cells should not rank SliM-LLM.

Six more models, the same conclusions in perplexity. Over all 13 models, ESPTQ$_{0.02}$ has lower WikiText-2 and C4 perplexity than PT2-LLM and gsp-0.2, and higher 0-shot[8] than PT2-LLM on all 13 and than gsp-0.2 on 10. ESPTQ$_{0.12}$ has lower perplexity than TWLA on all 13, by 1.03% or more except the unresolved 0.69% on LLaMA-2-13B (§5.1). Its 0-shot[8] is higher on nine models, two of them by only 0.02 and 0.03 points, and lower on Qwen2.5-3B, LLaMA-3.1-8B, LLaMA-2-13B and Qwen2.5-14B, by 0.31 to 1.05 points.

Caveats. PT2-LLM's public code, as we ran it, looks degraded on several models, for reasons we have not identified: its C4 perplexity exceeds 2-bit GPTQ's on 8 of the 13, and its 0-shot[8]

on Qwen2.5-7B is 1.20 points above chance. C4, in our calibration set but not in PT2-LLM's and TWLA's (Appendix G), may favor ESPTQ-w in panel (c). PTQTP's published perplexities are lower than ESPTQ$_{0.12}$'s on C4 for all four models with a value and on WikiText-2 for three of seven, from two dense ternary planes (Table H5).

Table H2: (a) 0-shot[8] and (b) MMLU (0-shot) on 13 models (weight-only quantization). 0-shot[8] excludes MMLU; marks as in Table H1.

Model	FP16	GPTQ	QuIP	SliM-LLM	PB-LLM	BiLLM	ARB-LLM	PT2-LLM	CAT-Q‡	TWLA	gsp-0.2	**ESPTQ**$_{0.02}$	**ESPTQ**$_{0.12}$
(a) 0-shot[8] (%)													
SmolLM-135M	47.63	31.00	31.57	31.68$^{\diamond}$	–	32.22	34.98	31.51	–	38.28	36.89	35.72	40.16
Qwen2.5-0.5B	52.19	30.84	31.24	31.86	31.55	30.46	32.29	30.90	–	42.69	40.13	39.95	44.10
LLaMA-3.2-1B	57.63	31.12	31.64	32.15	34.70	31.79	35.93	31.23	–	44.22	42.28	43.91	47.72
Qwen2.5-1.5B	62.25	31.74	32.98	33.55	33.01	32.30	38.59	29.99	–	51.63	46.02	47.59	53.34
Qwen3-1.7B	62.51	33.46	34.45	33.58	33.55	30.50	38.32	31.32	–	47.16	47.20	48.51	53.90
Qwen2.5-3B	66.05	33.25	35.47	36.07	42.95	37.53	45.55	31.20	–	59.43	51.94	53.15	58.38
LLaMA-3.2-3B	65.51	32.55	31.81	30.25	35.74	37.12	44.13	34.20	–	55.82	50.55	51.46	55.84
Qwen3-4B	67.82	34.92	36.77	44.26	44.81	36.67	51.40	34.82	–	59.40	55.51	57.51	60.45
LLaMA-2-7B	67.56	36.06	32.42	39.00$^{\diamond}$	31.54	39.12	51.44	43.73	56.36	61.30	54.69	56.63	61.33
Qwen2.5-7B	70.49	43.57	49.26	–	47.51	48.83	56.24	32.45	–	63.48	61.82	60.32	66.65
LLaMA-3.1-8B	71.98	32.81	34.81	30.13	35.56	37.89	37.72	36.58	–	62.70	55.53	58.02	62.39
LLaMA-2-13B	70.19	48.19	45.42	45.12	31.38	46.55	58.71	46.04	–	66.61	60.53	63.17	66.06
Qwen2.5-14B	72.97	43.43	46.95	51.24	53.76	56.50	60.53	43.69	–	70.72	63.22	66.31	70.26
(b) MMLU, 0-shot (%)													
SmolLM-135M	25.05	25.32	24.50	23.56$^{\diamond}$	–	25.70	23.78	23.52	–	23.69	22.95	23.12	23.00
Qwen2.5-0.5B	47.34	24.47	24.50	24.58	23.17	23.14	23.43	23.67	–	26.82	22.94	22.93	25.71
LLaMA-3.2-1B	37.68	24.36	24.08	23.07	22.95	24.60	22.97	22.92	–	25.04	23.02	23.39	25.36
Qwen2.5-1.5B	59.66	25.18	23.38	23.40	22.91	23.34	22.95	22.92	–	34.05	24.21	26.13	34.54
Qwen3-1.7B	60.59	24.00	24.11	23.39	23.11	24.23	23.05	22.91	–	27.05	23.10	27.74	37.97
Qwen2.5-3B	65.04	25.13	23.83	23.62	22.98	24.71	28.29	22.90	–	49.18	28.86	27.65	46.45
LLaMA-3.2-3B	55.03	23.58	24.07	23.54	22.93	22.88	22.94	22.95	–	34.47	23.02	23.47	23.54
Qwen3-4B	71.07	23.68	23.67	22.97	23.88	23.11	30.59	23.10	–	51.42	35.55	39.52	57.09
LLaMA-2-7B	41.77	23.44	23.12	23.02$^{\diamond}$	22.89	23.01	23.40	23.03	–	29.57	24.88	26.16	28.31
Qwen2.5-7B	71.77	23.78	29.63	–	28.04	29.41	44.95	23.09	–	61.71	51.30	53.02	61.39
LLaMA-3.1-8B	64.02	23.30	23.16	23.07	22.91	24.97	22.87	22.95	–	44.20	29.80	31.66	38.90
LLaMA-2-13B	52.44	24.14	22.93	22.97	24.75	22.97	30.73	23.03	–	42.90	28.28	32.50	42.79
Qwen2.5-14B	77.62	24.84	25.20	24.69	39.61	42.00	51.37	28.16	–	67.66	52.66	55.26	62.33

MMLU. We measured MMLU (Hendrycks et al., 2021) zero-shot but keep it out of 0-shot[8], because it is generally not a zero-shot task: the field reports it 5-shot, as the benchmark's protocol does, so Table H2(b) is not comparable with published values. PTQTP, which reports neither OpenBookQA nor LAMBADA nor the shots of its MMLU values, has no row in Tables H2 and H3, and our CAT-Q runs did not measure MMLU. On MMLU, TWLA is higher than ESPTQ$_{0.12}$ on 9 of the 13 models. Many cells sit at chance: 87 of the 141 quantized cells are below 25%, 37 of them near the 22.95% of always answering A. On LLaMA-3.2-3B, every quantized row but TWLA's is below 24.1% although ESPTQ$_{0.12}$ matches TWLA in 0-shot[8]; we conjecture, untested, that MMLU's lettered options, used by none of the eight tasks, cause this collapse.

Table H3: Accuracy (%) per task of FP16 and the ternary methods on LLaMA-2-7B and LLaMA-2-13B, which Appendix G compares with published values; ‡ as in Table H1.

Method	ARC-e	ARC-c	PIQA	HellaSwag	WinoGrande	BoolQ	OBQA	LAMBADA	0-sh[8]	MMLU
LLaMA-2-7B										
FP16	73.82	45.05	78.78	76.18	69.46	79.36	44.20	73.63	67.56	41.77
PT2-LLM	42.09	25.26	63.11	40.30	56.75	62.54	30.40	29.36	43.73	23.03
CAT-Q‡	59.64	33.62	72.96	61.70	61.33	70.76	35.80	55.04	56.36	–
TWLA	66.37	35.75	74.92	65.39	66.30	73.27	37.20	71.16	61.30	29.57
gsp-0.2	55.26	31.14	71.33	60.11	62.04	63.88	35.80	57.97	54.69	24.88
ESPTQ$_{0.02}$	58.04	32.85	72.47	62.43	62.90	68.81	35.40	60.16	56.63	26.16
ESPTQ$_{0.12}$	64.14	39.08	75.63	68.36	65.27	71.01	38.80	68.31	61.33	28.31
LLaMA-2-13B										
FP16	76.52	48.72	80.30	79.61	72.53	82.11	45.20	76.54	70.19	52.44
PT2-LLM	46.17	27.05	63.44	42.37	55.33	62.17	30.00	41.76	46.04	23.03
TWLA	71.89	42.66	76.39	72.53	69.93	79.76	42.80	76.89	66.61	42.90
gsp-0.2	64.48	36.95	74.97	68.07	65.98	68.07	39.60	66.12	60.53	28.28
ESPTQ$_{0.02}$	67.47	38.05	75.84	69.31	68.19	77.89	39.60	68.99	63.17	32.50
ESPTQ$_{0.12}$	71.51	43.26	77.37	73.33	69.85	77.74	41.60	73.82	66.06	42.79

H.2 QUANTIZATION TIME

Table H4: Quantization time (hours) of Table 1's methods on the A6000 machine, from loading the model to writing the compressed file. [s]: measured while another job shared the GPU; –: PB-LLM cannot run.

Model	Params (B)	GPTQ	QuIP	PB-LLM	BiLLM	ARB-LLM	PT2-LLM	TWLA	gsp-0.2	**ESPTQ**$_{0.02}$	**ESPTQ**$_{0.12}$
SmolLM-135M	0.135	0.03	0.05	–	0.07	0.25	0.04	0.30	0.17	0.23	0.23
Qwen2.5-0.5B	0.494	0.06	0.30	0.06	0.12	0.63	0.06	0.44	0.37	0.47	0.49
LLaMA-3.2-1B	1.236	0.12	0.45	0.12	0.18	1.49	0.10	0.69	0.42	0.57	0.56
Qwen2.5-1.5B	1.544	0.16	0.79	0.16	0.26	2.06	0.13	1.03	0.64	0.85	0.85
Qwen3-1.7B	1.721	0.17	0.54	0.16	0.26	2.17	0.14	1.00	0.65	0.88	0.86
Qwen2.5-3B	3.086	0.30	2.34	0.34	0.46	4.17	0.25	1.89	1.11	1.46	1.46
LLaMA-3.2-3B	3.213	0.29	1.06	0.37	0.39	4.64[s]	1.08[s]	2.35[s]	1.85[s]	2.55[s]	2.53[s]
Qwen3-4B	4.022	0.36	2.04	0.45	0.51	5.36	0.32	2.16	1.32	1.74	1.72
LLaMA-2-7B	6.738	0.64	3.35	1.22	0.68	9.16	0.49	3.10	1.98	2.50	2.42
Qwen2.5-7B	7.616	0.76	4.82	1.76	0.79	9.38	0.56	3.80	2.12	2.68	2.58
LLaMA-3.1-8B	8.030	0.74	3.55	1.75	0.80	9.94	0.55	4.08	2.21	2.84	2.68
LLaMA-2-13B	13.016	1.42	5.16	3.41	1.18	17.73	1.12	6.97	7.39	8.27	8.05
Qwen2.5-14B	14.770	1.51	5.97	4.08	1.44	18.68	1.13	7.63	5.26	6.35	6.05

Timed interval. Table H4 lists the 129 runs behind Figure 4. The interval includes calibration-set preparation but not evaluation, and for ESPTQ-w and gsp-0.2 it includes the FP16 teacher of logit distillation (Appendix E) and two logged 200-question ARC-Easy validation checks. Six LLaMA-3.2-3B runs overlapped another job and took 1.11–4.36× their Qwen2.5-3B times, against 0.45–1.07× for the other four methods, so Figure 4 omits this model.

Further values. Over Figure 4's 12 models, ESPTQ-w at $\rho = 0.02$ takes 4.8–8.2× as long as PT2-LLM and 0.70–1.19× as long as TWLA, and $\rho = 0.12$ takes 0.94–1.04× as long as $\rho = 0.02$. Only ESPTQ-w and gsp-0.2 take longer on LLaMA-2-13B than on Qwen2.5-14B, possibly because of a CPU-placed float16 teacher (untested).

H.3 NOMINAL VERSUS EFFECTIVE BITS OF EACH METHOD

Three counts. Table H5 sets each method's declared bits #W beside #W* (Appendix B) and the fixed-length accounting value, which counts every symbol at a fixed length ($\log_2 3$ bits per ternary value) plus the auxiliary metadata (scaling factors, offsets, indices), without packing or compression.

Table H5: Nominal bits #W, fixed-length accounting value and effective bits #W* on LLaMA-2-7B, with the mean #W* over Table H1's models and its ratio to #W. [a]: counted with the ten FP16 values per row and group that TWLA's code stores; [◇]: as in Table H1; –: not defined or not measured.

		Fixed-length	#W*		
Method	#W	(LLaMA-2-7B)	LLaMA-2-7B	mean (models)	#W*/#W
GPTQ	2.00	2.141	2.006	1.986	0.99
QuIP	2.00	2.003	1.868	1.869	0.93
SliM-LLM	2.00	2.156	2.552[◇]	2.481[◇]	1.24[◇]
PB-LLM	1.70	2.713	2.475	2.491	1.47
BiLLM	1.08	3.095	3.380	3.343	3.10
ARB-LLM	1.08	–	2.756	2.757	2.55
PT2-LLM	1.58	1.835	2.062	2.057	1.30
PTQTP	1.58	3.420	–	–	–
CAT-Q	1.58	1.710	1.946	–	–
TWLA	1.58	6.582[a]	5.458	5.416	3.43
symbols after zlib (by subtraction)			4.046		
FP16 μ and α per row and group			1.250		
FP16 Kronecker factors of KOTMS			0.162		
gsp-0.2	1.58	1.882	2.397	2.416	1.53
ESPTQ$_{0.02}$	1.58	3.305	2.097	2.115	1.34
ESPTQ$_{0.12}$	1.58	3.305	2.644	2.656	1.68

TWLA. TWLA declares 1.58 bits, one ternary value per weight (Zhao et al., 2026). Its public code, run with its defaults, splits each row's 128-column group into four partitions with a per-weight partition index, gives two of them a second ternary plane on salient columns, and keeps ten FP16

values (four μ, six α) per row and group plus the FP16 Kronecker factors of KOTMS. The indented rows of Table H5 split its 5.458 bits on LLaMA-2-7B accordingly; the symbol term is obtained by subtraction and would be 5.170 bits at fixed length.

Other declared values. BiLLM's and ARB-LLM's 1.08 bits count only $1 + r_{\text{salient}}$, with r_{salient} the salient fraction (Huang et al., 2024; Li et al., 2025), not flags or FP16 means and scales; our formula for ARB-LLM covers only that part. PTQTP labels its results 1.58-bit but stores two dense ternary planes, $2\log_2 3 + 2 \cdot 16/128 = 3.420$ bits. From CAT-Q's public LLaMA-2-7B weights, a separate script recovers one ternary plane and one magnitude per row and 128-column group, the coarsest granularity at which every group has a single nonzero magnitude, and counts them in memory: $\log_2 3 + 16/128 = 1.710$ bits at fixed length and 1.946 bits with zlib.

Why #W* can exceed the fixed-length value. On one-byte codes, zlib does not reach the entropy of near-uniform ternary symbols: CAT-Q's codes, 40.8% of them zero, take 1.821 bits each, above their empirical entropy of 1.568 bits and above $\log_2 3$. #W* thus charges dense ternary planes, as in PT2-LLM and CAT-Q, more than ideal packing would.

Table H6: Entropy-free layout of ESPTQ on LLaMA-2-7B (bits per quantized weight), with PT2-LLM under the same packing of T_1. All sizes are analytical; the T_2 entries and ESPTQ's totals are upper bounds.

Item	Size	ESPTQ$_{0.02}$	ESPTQ$_{0.12}$	PT2-LLM
T_1, five per byte, aligned per row and group	26 bytes / 128 weights	1.6250	1.6250	1.6250
Count of nonzero T_2 per row and group	8 bits	0.0625	0.0625	–
Read offset per row	32 bits	0.0067	0.0067	–
Nonzero T_2: in-group position and sign	7 + 1 bits, fraction $\leq \rho$	$\leq$ 0.1600	$\leq$ 0.9600	–
Scale and shift per row and group	INT8 $\times$2 / FP16 $\times$2	0.1250	0.1250	0.2500
Rotation angles	FP16, 448 per group	0.0098	0.0098	–
Reference values	FP16, 2 per linear layer	1.1×10^{-6}	1.1×10^{-6}	–
Column permutation (SSR)	32 bits per input column	–	–	0.0039
Total		$\leq$ 1.9891	$\leq$ 2.7891	1.8789
#W* with zlib (Table H1)		2.0973	2.6443	2.0617
Total $-$ #W*		≤ -0.1082	$\leq +0.1448$	-0.1828

Entropy-free layout. Table H6 itemizes the layout of §5.1, with an 8-bit count of nonzero T_2 per row and group and a 32-bit read offset per row to locate the variable-length T_2 lists. The T_2 bound fails only if a layer's λ is infinite (Appendix D), which never occurred in the 22 LLaMA-2-7B runs of the sweep and the ablation that record λ; it was not checked for Table 1's run. No kernel reads this layout, its offsets suit static weights but not the fixed-length-addressed KV cache, and the sizes hold for LLaMA-2-7B only.

H.4 KV-CACHE QUANTIZATION PER MODEL AND PER TASK

Table H7: Table 2 per model and per task, from the result file behind Table 2 (FP16 activations). §: model in Table 2; HS, WG, LAMB.: HellaSwag, WinoGrande, LAMBADA; block titles give the machine and, with ESPTQ-wkv rows, #KV* and the static tables (§4.4).

Method	#W*	Wiki	C4	ARC-e	ARC-c	PIQA	HS	WG	BoolQ	OBQA	LAMB.	0-sh^8	MMLU	GSM
SmolLM-135M§ (A5000; #KV* 3.3125, static 78.75 KiB)														
FP16	16.000	17.56	22.17	56.10	29.35	67.79	42.71	52.64	60.15	34.20	37.75	47.59	25.40	2.05
PT2-LLM	2.121	200.33	483.49	32.53	22.35	51.74	26.64	49.72	37.89	27.60	0.43	31.11	22.90	0.00
TWLA	5.459	51.98	86.68	38.89	22.10	57.13	31.36	51.85	62.20	26.60	14.05	38.02	22.97	0.76
ESPTQ-w$_{0.02}$	2.168	42.89	63.52	39.52	22.61	57.24	30.43	53.28	61.25	23.80	10.96	37.39	22.87	0.53
ESPTQ-w$_{0.12}$	2.699	30.97	44.55	41.84	24.91	60.72	33.95	53.83	60.46	26.40	15.74	39.73	22.93	1.52
ESPTQ-wkv$_{0.02}$	2.168	50.03	75.47	38.38	23.89	56.75	29.49	51.38	58.50	26.00	8.64	36.63	22.91	0.38
ESPTQ-wkv$_{0.12}$	2.699	35.80	51.96	40.74	25.51	59.25	32.53	51.38	59.48	26.20	13.00	38.51	22.97	1.52
Qwen2.5-3B§ (A5000; #KV* 3.28125, static 261.00 KiB)														
FP16	16.000	8.03	13.36	73.23	47.44	78.73	73.53	68.98	77.40	42.80	66.91	66.13	65.17	59.21
PT2-LLM	2.011	50.34	203.60	33.54	21.42	54.95	30.56	51.78	44.56	26.20	8.09	33.89	23.29	0.99
TWLA	5.282	10.92	19.71	64.14	39.25	72.69	60.78	64.40	64.10	37.40	62.08	58.10	45.25	21.15
ESPTQ-w$_{0.02}$	2.105	11.90	21.01	57.07	32.51	70.73	55.95	61.01	64.68	37.00	48.13	53.39	29.16	6.14
ESPTQ-w$_{0.12}$	2.641	10.38	17.91	59.30	35.15	73.23	62.33	63.46	67.74	39.00	55.27	56.94	42.71	13.65
ESPTQ-wkv$_{0.02}$	2.105	12.64	22.49	57.49	30.97	69.37	53.68	59.91	64.95	34.80	42.93	51.76	27.13	2.88
ESPTQ-wkv$_{0.12}$	2.641	10.99	19.06	58.63	35.07	72.52	60.30	62.27	67.89	36.40	49.21	55.29	37.22	7.58

Method	#W*	Wiki	C4	ARC-e	ARC-c	PIQA	HS	WG	BoolQ	OBQA	LAMB.	0-sh[8]	MMLU	GSM
LLaMA-3.2-3B[§] (4500 Ada; #KV* 3.28125, static 392.00 KiB)														
FP16	16.000	7.82	11.34	72.05	45.99	77.97	74.07	69.06	73.58	40.40	69.75	65.36	54.89	27.22
PT2-LLM	2.062	39.17	170.96	33.00	22.87	55.60	30.43	50.36	61.35	26.20	4.21	35.50	22.94	0.00
TWLA	5.434	11.98	20.83	60.65	35.49	71.60	58.68	63.38	68.41	36.60	55.40	56.28	31.24	4.17
ESPTQ-w$_{0.02}$	2.104	12.59	21.80	52.06	30.80	67.74	53.87	61.48	64.01	33.60	50.28	51.73	27.41	2.12
ESPTQ-w$_{0.12}$	2.652	10.69	17.77	60.06	36.18	72.52	62.50	63.61	67.43	39.60	57.38	57.41	33.94	5.53
ESPTQ-wkv$_{0.02}$	2.104	13.69	24.50	51.22	29.69	66.16	51.24	56.91	63.06	34.80	40.77	49.23	27.10	2.12
ESPTQ-wkv$_{0.12}$	2.652	11.69	20.04	52.61	31.06	69.59	58.26	59.35	64.77	37.00	50.36	52.88	28.19	2.43
Qwen1.5-4B[§] (4500 Ada; #KV* 3.28125, static 800.00 KiB)														
FP16	16.000	8.94	13.56	61.62	39.68	77.26	71.40	63.69	77.77	40.00	66.72	62.27	54.29	55.19
PT2-LLM	2.065	25.90	73.41	38.51	22.61	57.73	33.91	51.93	55.35	27.40	13.68	37.64	23.01	0.99
TWLA	5.514	11.48	18.94	61.11	33.36	71.00	59.92	60.38	71.04	33.20	54.84	55.61	37.07	9.10
ESPTQ-w$_{0.02}$	2.116	12.55	21.10	52.53	30.29	68.28	54.48	59.12	68.93	33.20	48.15	51.87	33.46	4.25
ESPTQ-w$_{0.12}$	2.660	11.10	17.88	59.05	33.28	72.80	60.91	61.80	68.04	36.40	55.40	55.96	36.58	12.28
ESPTQ-wkv$_{0.02}$	2.116	13.33	22.77	50.34	29.10	68.01	52.15	58.48	65.69	31.00	43.08	49.73	29.53	1.82
ESPTQ-wkv$_{0.12}$	2.660	11.81	19.27	57.20	31.74	71.65	58.45	60.46	66.67	36.20	50.09	54.06	34.15	6.67
Qwen3-4B (A5000)														
FP16	16.000	7.90	13.31	76.09	51.79	77.69	73.66	70.48	82.97	41.40	68.97	67.88	71.21	78.24
ESPTQ-w$_{0.02}$	2.113	10.97	19.59	58.59	36.60	70.62	58.23	60.54	68.38	35.00	51.19	54.89	43.38	11.83
ESPTQ-w$_{0.12}$	2.649	9.71	16.83	71.55	44.71	73.88	63.68	64.56	76.88	36.40	57.58	61.16	54.23	41.39
LLaMA-2-7B[§] (A6000; #KV* 3.28125, static 1024.00 KiB)														
FP16	16.000	5.47	7.26	73.82	45.05	78.78	76.18	69.46	79.36	44.20	73.63	67.56	41.77	13.50
PT2-LLM	2.062	11.73	24.77	42.09	25.26	63.11	40.30	56.75	62.54	30.40	29.36	43.73	23.03	1.67
TWLA	5.458	7.02	10.38	66.37	35.75	74.92	65.39	66.30	73.27	37.20	71.16	61.30	29.57	4.09
ESPTQ-w$_{0.02}$	2.097	7.51	11.42	58.04	32.85	72.47	62.43	62.90	68.81	35.40	60.16	56.63	26.16	1.82
ESPTQ-w$_{0.12}$	2.644	6.66	9.72	64.14	39.08	75.63	68.36	65.27	71.01	38.80	68.31	61.33	28.31	3.64
ESPTQ-wkv$_{0.02}$	2.097	8.10	12.62	53.54	31.31	69.59	58.71	60.06	66.88	34.60	52.26	53.37	25.01	1.67
ESPTQ-wkv$_{0.12}$	2.644	7.13	10.83	59.64	35.75	73.94	64.63	62.35	69.66	35.40	63.57	58.12	27.47	2.81
Qwen2.5-7B[§] (A6000; #KV* 3.28125, static 364.00 KiB)														
FP16	16.000	6.85	11.88	77.44	51.11	79.87	78.92	72.93	84.62	47.20	71.86	70.49	71.77	80.36
PT2-LLM	2.046	55.39	216.91	29.97	22.01	53.16	28.80	51.22	42.54	26.40	5.49	32.45	23.09	0.08
TWLA	5.439	8.60	15.31	66.88	41.98	77.20	69.79	70.17	72.08	40.00	69.71	63.48	61.71	54.74
ESPTQ-w$_{0.02}$	2.102	9.01	16.22	66.54	41.72	74.10	66.63	67.40	66.12	38.60	61.48	60.32	53.02	27.52
ESPTQ-w$_{0.12}$	2.636	8.18	14.37	75.97	46.33	76.93	71.24	69.85	82.78	42.80	67.32	66.65	61.39	46.25
ESPTQ-wkv$_{0.02}$	2.102	10.05	18.86	63.68	38.48	73.12	64.10	66.69	69.69	38.00	58.78	59.07	46.99	17.97
ESPTQ-wkv$_{0.12}$	2.636	9.05	16.28	74.75	45.14	75.63	69.52	68.19	80.92	41.40	63.15	64.84	54.49	26.61

Contents. Table H7 adds Qwen3-4B, whose ESPTQ-w rows enter Table 2's #W* averages, to Table 2's six models. PT2-LLM, TWLA and ESPTQ-wkv on Qwen3-4B have no result in Table 2's campaign yet. Table 2 averages ESPTQ-w over seven models and the other quantized rows over six; the 3.29 bits of its caption are 3.2865 bits for both ESPTQ-wkv rows.

The two stages. On each of Table 2's 12 model-ρ pairs, ESPTQ-wkv raises WikiText-2 perplexity and lowers 0-shot[8] relative to ESPTQ-w by less than ESPTQ-w does relative to FP16. In GSM8K, the weights cost more points than the KV stage on all six pairs of Qwen1.5-4B, Qwen2.5-3B and Qwen2.5-7B, yet on Qwen2.5-7B at $\rho = 0.12$ both stages remove the same fraction of the accuracy they start from (42.4% and 42.5%).

Differences between the Table 1 and Table 2 campaigns. Table 2's campaign ran five of Table H7's models on other machines (see the block titles), with the zero-shot batch size fixed at 4. On LLaMA-2-7B and Qwen2.5-7B (A6000), its FP16, PT2-LLM, TWLA and ESPTQ-w values are transcribed from Table 1's runs, except GSM8K, measured on weights quantized again with the same settings (#W* equal to four decimals, bit identity not checked). Shared rows on the other machines differ from Tables H1 and H2, the baselines more than ESPTQ-w: on SmolLM-135M, WikiText-2 perplexity is 200.33 against 260.75 (-23.2%) for PT2-LLM and 51.98 against 58.93 for TWLA, whereas ESPTQ-w moves by at most 1.5%. We have not separated these causes, so we compare rows only within one table: mixing them would make ternarizing SmolLM-135M's KV cache appear to raise 0-shot[8] at $\rho = 0.02$ (35.72 in Table 1, 36.63 for ESPTQ-wkv).

Separate runs and bridge points. Each ESPTQ-wkv row quantizes its weights in a run separate from the ESPTQ-w row; their #W* agree to four decimals except on LLaMA-2-7B at $\rho = 0.12$ (2.6443 against 2.6442 bits), where the weights differ for an unidentified reason. Our bridge points, three LLaMA-2-7B configurations (ESPTQ-w at both ρ and gsp-0.2) also run with Table 1's settings on the RTX A5000 machine, differ from Table 1's runs by at most 0.65% in WikiText-2 perplexity

and 0.78 points in 0-shot[8]. On the 7B models, whose two rows also differ in code version and batch size (§5), the KV stage costs at least 1.26 points of 0-shot[8], less than twice that 0.78.

I Additional Comparisons, Ablations and Negative Results

This appendix supports §1, §2, §3.2–§3.4 and §5.1 with measurements that do not enter Tables 1 and 2; we do not compare results measured on different machines.

I.1 Uniform-grid PTQ at 2 bits

§1 cites the best 2-bit WikiText-2 perplexity of uniform-grid PTQ on LLaMA-2-7B. Table I1 gives the three models on which all four methods run; the public code of AWQ or QuaRot does not cover the other ten models of Appendix H. GPTQ and QuIP come from the runs of Table 1, and AWQ and QuaRot run their public code in our harness with their own calibration sets and settings (Appendix G), QuaRot in the weight-only 2-bit condition of Table 7 of Ashkboos et al. (2024).

Results. QuaRot is the lowest of the four on all three models. On LLaMA-2-7B its 25.66 is $4.7\times$ FP16's 5.47, whereas ESPTQ reaches 7.51 at 2.097 bits ($\rho = 0.02$, Table 1). Our 25.66 is higher than the 22.07 that QuaRot reports under the same condition and the 19.97 that TWLA reports for 2-bit QuaRot (Zhao et al., 2026); QuaRot's own scripts on our machine gave 22.37, still $4.1\times$ FP16, and we have not found why our harness run differs.

Table I1: Uniform-grid PTQ at 2 bits (A6000 machine; Wiki and 0-sh[8] as in Table 1). #W is the declared width (AWQ's includes its FP16 group scale and zero point), and #W* the mean effective bits over the three models.

Method	#W	#W*	LLaMA-2-7B		LLaMA-3.1-8B		LLaMA-2-13B	
			0-sh[8]	Wiki	0-sh[8]	Wiki	0-sh[8]	Wiki
FP16	16	–	67.56	5.47	71.98	6.24	70.19	4.88
GPTQ	2.00	2.003	36.06	29.23	32.81	123.41	48.19	12.40
AWQ	2.25	2.296	30.54	2.23e5	30.53	1.54e6	33.49	1.23e5
QuaRot	2.00	2.173	39.25	25.66	35.18	19.13	49.34	10.21
QuIP	2.00	1.865	32.42	82.92	34.81	89.16	45.42	15.67

I.2 Both methods under one activation quantizer

§2 notes that TWLA is the only prior ternary-weight PTQ work with 4-bit activations, and §3.2 adds the activation-flattening term in view of such activations. Table I2 applies TWLA's activation quantizer to both methods on two models. It quantizes the inputs of the linear layers, not the KV cache. INT8 and INT4 are per-token affine quantizers with one clipping ratio per linear layer. MP is TWLA's inter-layer aware activation mixed precision (ILA-AMP), which assigns 2 to 8 bits to each decoder layer (mean 4); on Qwen2.5-0.5B it chose 4 bits everywhere, so MP equals INT4 there. Both methods calibrate on 128 WikiText-2 windows of 2048 tokens, and each row's four settings share one set of quantized weights.

Results. INT8 costs at most 0.39% in perplexity and 0.42 points in 0-shot[7]. INT4 degrades all six combinations: on Qwen2.5-3B, perplexity rises by 117.3%, 60.2% and 41.0% for ESPTQ$_{0.02}$, ESPTQ$_{0.12}$ and TWLA. On Qwen2.5-0.5B, ESPTQ$_{0.12}$ stays lowest at INT4, as it already is with FP16 activations.

What it does not show. TWLA holds 2.0 to 2.5 times the weight bits of ESPTQ, so the comparison cannot separate the weight method from the weight bits. Every ESPTQ run includes the activation-flattening term (as implemented; Appendix C), whose effect is not isolated: the only comparison with and without it under quantized activations, a design-stage screening on Qwen2.5-0.5B at $\rho = 0.02$, gave perplexity 94.8 against 275.7 under per-token INT4, but changed the $\pi/4$ initialization at the same time. The clipping search often stopped at an edge of its candidate range (216 of 252 ESPTQ layers at INT8 on Qwen2.5-3B), and we did not try a wider range.

Table I2: Both methods under TWLA's activation quantizer (one RTX A5000; float16 weights; 0-sh[7] omits LAMBADA from 0-sh[8]). These runs did not record #W*, so it repeats the runs of Appendix H; the values are not comparable with Tables 1 and 2.

Model	Method	#W*	Wiki				0-sh[7]			
			FP16	INT8	INT4	MP	FP16	INT8	INT4	MP
Qwen2.5-0.5B	ESPTQ$_{0.02}$	2.15	23.64	23.68	37.95	37.95	42.06	41.82	39.48	39.48
	ESPTQ$_{0.12}$	2.70	19.18	19.24	33.98	33.98	44.49	44.43	41.10	41.10
	TWLA	5.31	25.68	25.69	43.31	43.31	43.77	43.72	40.59	40.59
Qwen2.5-3B	ESPTQ$_{0.02}$	2.11	11.77	11.81	25.57	17.99	52.77	52.82	43.46	43.81
	ESPTQ$_{0.12}$	2.64	10.29	10.33	16.48	15.80	57.22	56.80	47.49	47.20
	TWLA	5.27	10.94	10.96	15.42	14.42	58.33	58.45	52.75	52.86

I.3 Rotation ablation

This ablation varies only the rotation of LLaMA-2-7B at $\rho = 0.12$ (§3.2, §5.1).

Arms. A3, the rotation of Table 1, learns *blockfly* from the $\pi/4$ initialization with the activation-flattening term (Appendix C). The other arms of Table I3 change one element at a time: A3h drops the term, A3w starts A3h from identity, A3g widens A3w to a butterfly over all inputs of the six projections of width 4096, A1 fixes the angles at $\pi/4$, A0 removes the rotation, and A2 learns the released KOTMS (Appendix A) under A3w's objective and initialization. A2 stores 0.168 bits more than A3, so the family pair uses A2′ at $\rho = 0.09$, whose #W* is close to that of A3w. A3 with column permutation could not run, because our block-wise calibration and accounting do not handle permuted columns.

Table I3: Arms of the rotation ablation (LLaMA-2-7B, $\rho = 0.12$ unless stated; one RTX A5000; calibration on WikiText-2 alone; A3 with seed 0). Rot. is the rotation's storage per quantized weight; all arms but A3 use the weight-only objective.

Arm	Rotation setting	Rot. (bits)	#W*	Wiki	C4	0-sh[8]
A0	none	0.00000	2.5918	14.4213	10.8606	58.62
A1	Hadamard, fixed at $\pi/4$	0.00000	2.6228	6.6750	10.2573	60.22
A3h	*blockfly*, $\pi/4$ init.	0.00985	2.6429	6.5654	10.0367	61.05
A3w	*blockfly*, identity init.	0.00985	2.6401	6.5170	9.8954	60.57
A3g	4096-column butterfly, identity init.	0.01470	2.6463	6.5048	9.8356	61.09
A2′	released KOTMS, $\rho = 0.09$	0.16183	2.6325	6.5588	9.8460	61.01
A2	released KOTMS	0.16183	2.8107	6.4664	9.6374	62.23
A3	A3h + activation-flattening term	0.00985	2.6423	6.6040	10.0116	61.46

Conditions and decision margin. Measured on both machines, three configurations with Table 1's settings differ by up to 0.65% in WikiText-2 perplexity, so we never tabulate these arms with Table 1. Three seeds of A3 give a standard deviation σ of 0.254% of WikiText-2 perplexity and 0.296% of C4 perplexity (Appendix G), so we judge a difference of two runs, with A3 at its three-seed mean, against the decision margin $2\sqrt{2}\sigma$: 0.72% and 0.84%. This is a working rule, not a significance test, since three seeds leave σ uncertain (the t quantile 4.30 would give 1.54% and 1.80%); zero-shot accuracy has no margin.

Findings by pair (Figure I1). Learning the angles (A1→A3h) lowers perplexity by 1.64% (WikiText-2) and 2.15% (C4) and raises 0-shot[8] by 0.83 points, beyond even the t-based margins. The rotation families are indistinguishable at almost equal bits: *blockfly* (A3w) is 0.64% lower than KOTMS (A2′) in WikiText-2 perplexity and KOTMS 0.50% lower in C4, both within the margins. We adopt *blockfly* for its lower cost (§3.2, Appendix C). Widening the span (A3w→A3g) stays within the margins at 24 instead of 14 multiplications.

Removing the rotation. A1→A0 raises WikiText-2 perplexity by 116.05% but C4 perplexity by only 5.88%, a gap we have not explained. With the second plane fixed, the rotation is thus the element of largest effect in this ablation, far larger on WikiText-2 than on C4.

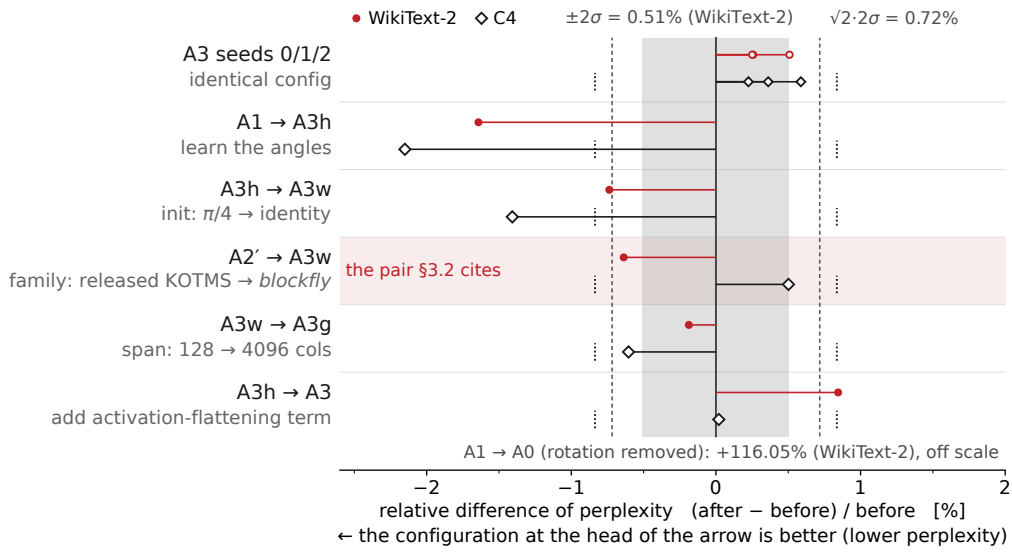


Figure I1: Relative perplexity difference (after − before)/before of each pair of arms that differ in one element (circles: WikiText-2; diamonds: C4; A1→A0 is off scale), with the seed-only differences of A3 in the top row. Gray band: $\pm 2\sigma$; dashed lines and dotted ticks: the decision margins $\pm 0.72\%$ (WikiText-2) and $\pm 0.84\%$ (C4).

Initialization and the activation-flattening term. Identity initialization (A3h→A3w) lowers perplexity by 0.74% (WikiText-2) and 1.41% (C4) but lowers 0-shot[8] by 0.48 points; the WikiText-2 difference barely exceeds the margin, both lie within the t-based margins, and the disagreement is unexplained. Adding the term (A3h→A3) costs 0.84% in WikiText-2 perplexity, slightly beyond the margin but within the t-based one, and 0.02% in C4, so we claim no benefit for it.

The default versus the ablation. Table 1 keeps A3, whose $\pi/4$ initialization entered together with the term at design time; no measurement separated the two, and no arm combines identity initialization with the term. In perplexity A3 is not the best arm: it is 1.59% (WikiText-2) and 1.45% (C4) above A3w, and A3g and A2′ are also lower at #W* within 0.01 bits, although A3 has the highest 0-shot[8] of the four. §3.2 therefore states its KOTMS comparison for A3w, not for A3, which is 0.95% and 1.96% above A2′. Tables 1 and 2 were not re-run with A3w (Appendix J).

I.4 Layer-error minimization, mask placement and column permutation

§3.4 states that none of the six approaches we tried for minimizing the layer error more precisely improved perplexity consistently. Table I4 lists them as rows (1) to (6).

Approaches 1 to 4. These ran on an earlier group-granular design, in which the top fraction p of 128-column groups carry a second plane on all elements and GPTQ rounds greedily (the baseline row). Approach 1 weights errors by OBQ's sensitivity $1/[H^{-1}]_{jj}$ (Frantar et al., 2022) instead of $h_j = H_{jj}$ and also changes how groups are chosen (note a). Approach 2 rounds by beam search under the Hessian metric, lowering the layer output error by 8 to 11% (note b: beam width 8 on SmolLM-135M, unrecorded on LLaMA-2-7B). Approach 3 picks per row the candidate grid with the smaller group output error after GPTQ rounding, and (4a) to (4c) add an earlier grid to the candidates. No approach lowered perplexity in all four cells, and each raised it in at least two; this is the sense of §3.4. On LLaMA-2-7B alone, approach 3 and (4b) improved at both p, but they worsened SmolLM-135M at both.

Approaches 5 and 6. These ran on a later element-wise version (Qwen2.5-0.5B, *blockfly*, no two-stage calibration or rate-distortion criterion), whose control fixes the second-plane positions in the prepass at the fraction ρ of largest salience $h_j \varepsilon_{ij}^2$. Approach 6 fits the inter-plane ratio r per linear layer to the prepass residuals by least squares and is 16.9% to 28.2% worse at equal ρ, one reason

Table I4: Six approaches to more precise layer-error minimization (design-stage measurements: WikiText-2 perplexity, single seed, machine not recorded). Upper block: group-granular stage, bits equal within 0.0015 at each p; lower block: element-wise stage with bits in parentheses (n.c.: outside the control's bit range).

	LLaMA-2-7B		SmolLM-135M	
Approach (design stage)	$p = 0.2$	$p = 0.3$	$p = 0.2$	$p = 0.3$
Baseline grid of that stage	10.47	9.06	145.08	62.80
(1) OBQ inverse-Hessian metric[a]	11.27	9.17	123.12	63.19
(2) Beam search under the Hessian metric[b]	10.62	9.10	125.42	66.23
(3) Candidate grid by group output error	10.35	8.95	152.51	72.19
(4a) Merged candidates, group output error	10.63	8.89	144.27	67.76
(4b) Merged candidates, diagonal-H error	10.41	8.96	146.21	67.91
(4c) 2 merged candidates, diagonal-H error	10.72	8.87	141.10	72.86
Qwen2.5-0.5B	$\rho = 0.01$	$\rho = 0.02$	$\rho = 0.04$	$\rho = 0.08$
Control: prepass mask, fixed $r = 0.65$	64.81 (1.799)	52.03 (1.864)	38.31 (1.974)	29.73 (2.155)
(5) Mask decided in the main loop	73.09 (1.825)	48.42 (1.901)	37.10 (2.027)	25.48 (2.233)
(5) vs. control at equal bits	+22.3%	+2.1%	+3.6%	n.c.
(6) Per-layer least-squares r	81.19 (1.798)	63.89 (1.862)	49.10 (1.970)	34.74 (2.146)

Appendix B fixes r. Approach 5 decides the mask inside the main loop, giving the second plane to the elements whose compensated weights gain most from nine levels, above a per-layer threshold taken from the prepass gains. Against the control interpolated in bits, it is 22.3%, 2.1% and 3.6% worse at $\rho = 0.01$, 0.02 and 0.04. We conjecture that choosing positions from compensated weights couples the selection with GPTQ's compensation path, so §3.3 fixes the mask in the prepass.

Column permutation. PT2-LLM reorders columns (Yan et al., 2026). On the element-wise version above, ordering columns by decreasing salience $h_j \sum_i \varepsilon_{ij}^2$ lowered perplexity in 17 of 40 pairs of model and ρ but raised LLaMA-2-7B's by 21.3% to 34.4%, above even the version with neither rotation nor permutation. On the group-granular stage, GPTQ's act-order likewise helped SmolLM-135M and hurt LLaMA-2-7B at both p. We therefore do not permute, for accuracy.

J LIMITATIONS

This appendix supports §6, grouping limitations of the measurements and of the method.

J.1 SCOPE OF THE MEASUREMENTS

Single runs. Every result is a single run with a fixed seed. The decision margin rests on three seeds of LLaMA-2-7B alone, 0.72% in WikiText-2 perplexity or 1.54% with the t quantile (Appendix G), so differences near it, such as the 0.69% lead over TWLA on LLaMA-2-13B, stay unresolved. Zero-shot accuracy has no margin, and reruns of identical settings differ by up to 0.20 points per task.

Models and contexts. Weight-only results stop at Qwen2.5-14B (§5) and KV-cache results at the 7B models of Table 2. All models are base checkpoints evaluated on at most 2048 tokens (Appendix G), so we make no claim on larger or instruction-tuned models or on longer contexts.

Two measurement campaigns. Tables 1 and 2 ran on different machines and code versions, and re-quantizing with the same settings for Table 2 moved PT2-LLM's WikiText-2 perplexity on SmolLM-135M by -23.2%, against at most 1.5% for ESPTQ-w (Appendices G and H). We therefore compare rows only within one table, and within Table 2 only within one model. On the 7B models, whose ESPTQ-w and ESPTQ-wkv rows also differ in code version and batch size, the KV stage's 0-shot[8] cost of at least 1.26 points is under twice the 0.78-point cross-machine gap of our bridge points (Appendix H).

Analytical hardware claims. The residual multiplication ratios are analytical counts in our circuit model, and the sizes of the entropy-free layout and of the packed KV cache are computed from the

format (Appendices B, F and H). We built no circuit and no kernel that reads the quantized weights or the packed cache, and we measured no latency, area or energy. The evaluated models also keep FP32 rotation angles where the files store FP16 (Appendix B).

PTQTP. Table 1's PTQTP row shows published values, as our reimplementation fell short (9.03 against 8.31 in WikiText-2 perplexity on Qwen2.5-7B; Appendix G), so comparisons with it cross evaluation protocols.

J.2 Limitations of the method

The activation-flattening term. As implemented, the term evens out the channel energies of the inversely rotated input $xR_g^\top$, not of xR_g, which inference computes (Appendix C). Under quantized activations its effect is unmeasured in the final configuration, because the only comparison with and without it, a design-stage screening on Qwen2.5-0.5B, also changed the initialization (Appendix I). With FP16 activations it raised WikiText-2 perplexity by 0.84%, so we claim no benefit for it.

The $\pi/4$ initialization. The ablation does not support this default. Identity initialization lowers WikiText-2 and C4 perplexity by 0.74% and 1.41% but also lowers 0-shot[8] by 0.48 points, a disagreement we cannot explain, and no arm combines it with the activation-flattening term (Appendix I). Tables 1 and 2 were not re-run with the identity-initialized arm A3w, 1.59% lower in WikiText-2 perplexity than Table 1's rotation, so our perplexities may exceed what tested alternatives reach.

Design-stage negative results. The six approaches to more precise layer-error minimization, each tested on one or two models, and column permutation were rejected on earlier versions with one seed and never re-tested in the final configuration (Appendix I), so they are ruled out only for those versions. The gain of stage ③ of the KV write path likewise comes from older code (Appendix F).

Group confinement. *blockfly* mixes coordinates only within 128-column groups, so it cannot move an outlier's mass out of its group, and layers whose input width is not a multiple of 128 stay unrotated (Appendix C). On SmolLM-135M, these are six of the seven projections of every decoder layer. We have not measured how much they contribute to its large remaining gap from FP16 (§5.1).

Multi-step reasoning. GSM8K degrades most (§5.2). The weights cost more points than the KV stage on all four model-ρ pairs of Qwen1.5-4B and Qwen2.5-7B, yet on Qwen2.5-7B at $\rho = 0.12$ the KV stage removes 42.5% of the accuracy it starts from, as large a fraction as the weights (Appendix H), so both stages must improve for multi-step reasoning.