DENSITY-MATCHED POWER-OF-TWO QUANTIZATION: ALLOCATING LOGARITHMIC LEVELS FOR MULTIPLIER-FREE LLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) inference spends most of its compute in the multiply-accumulate (MAC) operations of linear projections. Post-training quantization (PTQ) reduces this cost with low-bit integer weights and activations, leaving a multiplier in every MAC. Power-of-Two (PoT) weight quantization replaces that multiplier with a shifter but spaces its levels a factor of two apart, degrading accuracy at low bit-widths. Finer fixed-ratio grids narrow the gap but remain suboptimal, since they spend the same number of levels per doubling of magnitude, wherever the weights lie. We introduce **Density-Matched Power-of-Two (DMPoT)** quantization, which formulates logarithmic level allocation as a continuous point-density optimization problem. We derive the density that minimizes quantization mean squared error (MSE) and use it to guide the construction of a multi-piece family of fractional PoT grids whose deployable members support folded shift-add hardware. Three results follow. (i) The derived density concentrates quantization resolution on the upper flank of the weight distribution while leaving the bottom of the range sparse. (ii) The optimal grids at successive bit-widths nest as sets of levels into one chain, and two bit-width laws fix the operating regime at four to five bits: a one-adder-per-weight bound and a width-minimality result at INT-matched MSE. (iii) Every deployed 4-bit level is an integer shift of 1 or of 3, so a folded datapath computes the single addition once per activation, shared by all output channels. Across seven models and two configurations, DMPoT matches the uniform integer baseline in all fourteen cells, never more than 0.09 perplexity above it, with mean gaps of -0.008 and $+0.038$. The synthesized MAC array with 4-bit weights and activations occupies $0.702\times$ the area of its integer counterpart.

1 INTRODUCTION

Deploying Large Language Models (LLMs) (Touvron et al., 2023; Grattafiori et al., 2024; Yang et al., 2024; Jiang et al., 2023) on resource-constrained devices remains a challenge due to their costly inference. The cost of LLM inference is concentrated in the multiply-accumulate (MAC) operations of the Transformer (Vaswani et al., 2017) linear projections, which also hold most of the parameters (Dettmers et al., 2022). Post-training quantization (PTQ) lowers this cost by rounding weights and activations onto a low-bit integer grid, and at 8 bits it is now essentially lossless (Xiao et al., 2023). Recent methods (Ashkboos et al., 2024; Liu et al., 2025; Hu et al., 2025; Lin et al., 2024a) have pushed the frontier to 4-bit weights, activations, and key-value (KV) cache in uniform integer (INT) format.[1]

Although cheaper than its full-precision counterpart, an INT datapath still pays for a multiplier in every MAC. Power-of-Two (PoT) quantization, or logarithmic quantization (Elhoushi et al., 2021; Li et al., 2020), removes the multiplier entirely. Constraining weight magnitudes to 2^n turns every MAC into a shift and an integer addition. A shift-only MAC array is smaller than the integer multiplier array it replaces, lowering the hardware cost of the MAC operations (§5). PoT quantization has been studied extensively for Convolutional Neural Networks (CNNs) (Elhoushi et al., 2021; Li et al., 2020; 2023) under quantization-aware training (QAT), but far less for LLMs, where model scale

[1]Throughout, WxAyKVz denotes x-bit weights, y-bit activations, and a z-bit KV cache (here, W4A4KV4).

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.

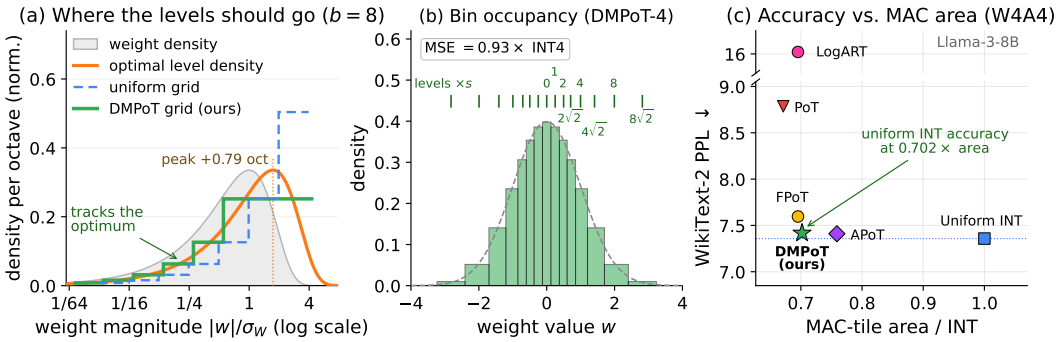

Figure 1: (a) The log-magnitude density of Gaussian weights (gray) and its MSE-optimal level density (orange), which peaks $+0.79$ octaves above σ_W. At b=8, uniform INT (blue) puts half its levels in the top octave, past that peak, while DMPoT (green) tracks the optimum. (b) The exact 4-bit DMPoT grid and its bin occupancy on a unit Gaussian. (c) W4A4 WikiText-2 perplexity against MAC-tile area (Llama-3-8B, 3 seeds). DMPoT matches uniform INT at $0.702\times$ its area.

favors PTQ over QAT. Basic PoT spaces its levels a factor of two apart (one octave), so the spacing between neighboring levels, and the rounding error with it, doubles from one level to the next. Under LLM PTQ, basic PoT therefore falls short of the uniform INT baseline in both perplexity (PPL) and zero-shot accuracy ($+1.43$ PPL on Llama-3-8B, §6.2).

A natural remedy for the accuracy loss is a finer grid. Fractional PoT (FPoT) (Anonymous, 2026) subdivides each octave into k equal-ratio steps and identifies k=2 (base-$\sqrt{2}$) as the near-optimal fraction, which holds accuracy close to uniform INT on a shift-add datapath. That optimum, however, is a single fraction shared by every octave. Each octave receives the same number of levels wherever the weights lie, and a wider code extends the range at the same density rather than refining it. The fixed-ratio, log-uniform spacing itself is inherited from PoT, not derived from the weight distribution. This paper asks two questions: given a fixed budget of levels, the $2^{b-1}-1$ magnitudes a b-bit code can represent, how many should each range of weight magnitudes receive, and which grids let the hardware that realizes them be shared across an array? The first decides accuracy at fixed hardware, and the second decides area at fixed accuracy.

We introduce **Density-Matched Power-of-Two (DMPoT)**, a weight grid whose level allocation follows the mean-squared-error (MSE)-optimal level density of the preprocessed weight distribution, constrained to remain foldable in hardware. DMPoT is applied to the linear-projection weights only, leaving activations and the KV cache in uniform INT. In this configuration, DMPoT stays near-lossless against uniform INT. We define *near-lossless* as a perplexity increase of at most 0.5 and a drop of at most 1.0%p in mean zero-shot accuracy over nine downstream tasks, both relative to the uniform INT baseline, as in FPoT (Anonymous, 2026).

Our contributions are as follows.

- **Derivation-guided grid design**: The MSE-optimal spacing is not log-uniform but places resolution on the upper flank and leaves the bottom of the range sparse. The grids this selects cut distortion 19% below a matched fixed-ratio control at 4-bit and nest, as sets of levels, into a single chain across bit-widths (§4.1, Table 4, Appendix B.3).

- **Hardware feasibility as a design principle**: A one-addition-per-weight budget admits a single deployable form of the derived grid and bounds the chain's one-adder deployment at $b\leq 5$. Every 4-bit level is an integer shift of 1 or of 3, an odd core of $\{1, 3\}$, and no narrower core reaches INT MSE. Its single addition is shared by all output channels, folding the MAC array to 7% less area than the closest accuracy-competitive alternative (§4.2, §5).

- **Near-lossless, multiplier-free LLM inference**: Across seven models at W4A4KV4 and W4A8KV8 (fourteen cells) and along the chain from four to eight bits on Llama-3-8B, accuracy is near-lossless against uniform INT. At four and five bits the weight datapath contains no multiplier, and the 4-bit member has lower perplexity than FPoT at near-identical hardware (§6.2, §6.3).

2 Related Work

In this section, we position DMPoT against three lines of work: PTQ preprocessing for LLMs (§2.1), shift-based grids (§2.2), and non-uniform number formats (§2.3).

2.1 Post-Training Quantization (PTQ) for LLMs

Uniform INT weight quantization methods such as GPTQ (Frantar et al., 2023) and AWQ (Lin et al., 2024b) reach competitive 4-bit accuracy via Hessian-weighted error compensation and activation-aware scaling. For activation quantization, the difficulty lies in outliers that concentrate in a few channels (Dettmers et al., 2022) and dominate the per-tensor scale. SmoothQuant (Xiao et al., 2023) migrates the difficulty to the weights via per-channel scaling. Rotation-based methods redistribute the outliers through randomized Hadamard rotations (QuaRot (Ashkboos et al., 2024)), learned rotations (SpinQuant (Liu et al., 2025)), jointly trained rotation and smooth scales (OSTQuant (Hu et al., 2025)), or dual block-diagonal transforms with channel permutation (DuQuant (Lin et al., 2024a)), enabling W4A4KV4 uniform quantization.

These methods, however, all leave the grid uniform. The preprocessing changes the distribution that reaches the quantizer, not the grid it is quantized to. We keep that preprocessing and replace only the weight grid, training the transform through it (§3, §6.2).

2.2 Power-of-Two (PoT) and Shift-based Quantization

PoT quantization (Elhoushi et al., 2021) constrains weights to $\pm 2^n$ values and replaces multipliers with barrel shifters. The CNN line (DeepShift (Elhoushi et al., 2021), APoT (Li et al., 2020), DenseShift (Li et al., 2023)) relies on QAT and does not transfer to PTQ at LLM scale. For LLMs, PoT-PTQ (Wang et al., 2025) reaches 2–3-bit base-2 weights via data-agnostic scale initialization and residual fine-tuning. ShiftAddLLM (You et al., 2024) rewrites each weight matrix as binary matrices with power-of-two group scales, so multiplications become shifts, additions, and lookups. LogART (Xu et al., 2026) uses learnable rounding and dynamic base allocation between base-2 and base-$\sqrt{2}$. All three retain FP16 activations, with an FP16 accumulator that no integer shift-add unit can absorb. Table 1 carries LogART's weight quantizer on INT activations (§6.2).

Additive PoT. The CNN line's grids outlive its training recipe. APoT (Li et al., 2020) writes each weight as a sum of n power-of-two terms drawn from disjoint sub-alphabets. The additive construction gives it *more* placement freedom than any fixed-ratio geometric grid. At $b{=}4$ with $n{=}2$ its non-zero magnitudes are $\{1, 2, 3, 4, 6, 8, 10\}$. APoT is a baseline in every main-results table.

Fractional PoT. FPoT (Anonymous, 2026) pairs a fixed half-octave (base-$\sqrt{2}$) weight grid with uniform INT activations and realizes the odd-exponent coefficient with a single adder ($x + x{\gg}1 = 1.5x$), yielding a fully integer shift-add processing element (PE) at 4-bit. Its two calibration-side techniques, approximate-grid quantization (§3) and weight grid-alignment (§3), are adopted here unchanged. In the family studied here, FPoT is the log-uniform member at two levels per octave. Its *constant* density over the entire range is the assumption the density analysis of §4.1 rejects.

2.3 Non-uniform Grids and Number Formats

There have been works that attempted to utilize non-uniform grids and number formats. For example, tapered-precision formats (Gustafson & Yonemoto, 2017; Luo et al., 2024; Rouhani et al., 2023; Guo et al., 2022) and block-scaled 4-bit floating point (Open Compute Project, 2023; NVIDIA et al., 2026) also vary the levels per octave. However, they made the variation by convention, not by a distortion criterion, and the latter fixes the E2M1 lattice on both operands. Our 4-bit magnitudes coincide with the E2M1 lattice up to a power of two (§4.2). Our contribution is the allocation and the log-weight, INT-activation datapath it admits. Floating-point engines lie outside the comparison class of §5 (Appendix B.6). NF4 (Dettmers et al., 2023) and SqueezeLLM (Kim et al., 2024) place LLM weights on Gaussian quantiles and a per-channel codebook. Neither factors into an odd core, so neither folds (Appendix B.9).

3 DENSITY-MATCHED POWER-OF-TWO QUANTIZATION

DMPoT quantization allocates a fixed budget of logarithmic levels across octaves where the weight distribution warrants them. We quantize the weights of every linear projection in each Transformer block (four attention, three MLP) with DMPoT, while activations and the KV cache remain uniform INT throughout. Confining the logarithmic grid to the static weights lets weight calibration absorb its distortion, and leaves the dynamic tensors in the format that an integer shift-accumulate datapath consumes directly (Appendix D measures the grid's per-projection tolerance).

Weight distributions are first redistributed by state-of-the-art LLM PTQ preprocessing, which applies a learned rotation and per-channel scaling to spread channel-concentrated outliers (online cost $\approx 0.26\%$ of Llama-3-8B's weight MACs, Appendix A.2). We use this procedure unchanged. The preprocessed weights are then calibrated with GPTQ (Frantar et al., 2023) against the DMPoT level set. A calibration-time grid-alignment, a phase-shift search inherited from FPoT (Anonymous, 2026), refines the clip-searched per-channel scale at zero inference cost (protocol in Appendix A.4). The transform is trained with straight-through gradients through the exact DMPoT grid, and the deployed levels enter at calibration (Appendix H.3).

Grid family and formulation. A logarithmic grid is characterized by how many levels it spends per octave. We therefore parameterize grids by a *schedule* of octave densities: a schedule assigns k_i levels per octave to J_i consecutive octaves, giving the multi-piece exponent map

$$\phi(e) \;=\; \sum_i \frac{\min\big((e - C_{i-1})^+,\, k_i J_i\big)}{k_i}, \quad C_i = \textstyle\sum_{j\le i} k_j J_j,\; C_0 = 0, \quad e \in \big\{0, 1, \ldots, 2^{\,b-1}{-}2\big\}, \tag{1}$$

where b is the weight bit-width. The sign bit and the reserved zero code leave $2^{\,b-1}{-}1$ non-zero magnitudes (7 at $b{=}4$), the level budget a schedule distributes across octaves (Fig. 1a). Densities are restricted to powers of two, $k_i \in \{1, 2, 4, \ldots\}$, so that the in-octave index stays a bit-slice of the stored code, every level of a density up to $k_i{=}4$ stays on the one-adder alphabet (§4.2), and the per-width optima nest (§4.1). For a per-output-channel scale $s > 0$, the representable set and the quantizer are

$$\mathcal{L}_\phi(s) = \{0\} \cup \big\{\pm\, s\, 2^{\,\phi(e)}\big\}_e, \qquad Q_{\mathrm{DMPoT}}(w) = \arg\min_{\ell \in \mathcal{L}_\phi(s)} (w - \ell)^2. \tag{2}$$

We write a schedule as the list of its octave densities, abbreviating repeats with superscripts (so $[1, 2^3]$ holds one level in the bottom octave and two per octave over the next three). The log-uniform (constant-density) schedules recover the prior PoT grids: $[1, 1, \ldots]$ is base-2 PoT and $[2, 2, \ldots]$ (base $\sqrt{2}$) is the FPoT grid, the $J{=}0$ member of $[1^J, 2, \ldots]$, which spends its bottom J octaves at one level each (Appendix A.3 expands the notation).

DMPoT's schedule is selected in §4. The log-domain point density that minimizes distortion is stationary on the *upper flank* of the weight log-magnitude density rather than at its mode, which predicts a sparse bottom octave. An exhaustive search over the family independently returns $[1, 2^3]$ at $b{=}4$. We write **DMPoT-4** for that member ($J{=}1$). Its exact magnitudes span 3.5 octaves.

Approximate-grid quantization. Above the sparse bottom octave the exact DMPoT-4 grid places levels at half-octave ratios $2^{1/2} \approx 1.414$. A one-adder budget admits only 1.5 near the half-octave point (§4.2), mapping $\{1, 2, 2\sqrt{2}, 4, 4\sqrt{2}, 8, 8\sqrt{2}\}$ to $\{1, 2, 3, 4, 6, 8, 12\}$ at a 6.1% ratio error. As in FPoT (Anonymous, 2026), weights are quantized directly to the *actual hardware level set*, $\mathcal{L}^{\mathrm{dep}}(s) = \{0\} \cup \{\pm s \cdot \{1, 2, 3, 4, 6, 8, 12\}\}$. Calibration therefore absorbs the substitution rather than leaving it as an uncompensated dequantization error at inference (calibration stages in Appendix A.4, empirical check in Appendix H.3 and Table 19, hardware cost of the alternatives in §5). The $b{\ge}5$ schedules deploy to integer sets of the same shape (Table 5).

Stored format. Every deployed DMPoT-4 magnitude factors as $c \cdot 2^t$ with a two-element coefficient alphabet $c \in \{1, 1.5\}$ (e.g. $3 = 1.5 \cdot 2^1$, $12 = 1.5 \cdot 2^3$), that is, an integer shift of an odd core in $\mathcal{Q}{=}\{1, 3\}$. The stored word is therefore a sign bit, a coefficient bit selecting c, and a two-bit shift field t, with the unused pattern $(c, t){=}(1.5, 0)$ as the zero code (Appendix A.6). The hardware consequence, one shared $3x$ adder per activation lane serving all output channels, is developed in §5.

Zero threshold and decision rule. Rounding is nearest-level projection in the linear domain, the minimum-squared-error rule for the deployed set, which is irregular in $\log_2$. Weights with $|w| < s/2$ map to zero (τ=0.5). Every grid comparison in §4 uses this rule (boundary placement, log-domain reading, sweep, and zero fractions in Appendix H.4).

Implementation. Every DMPoT-4 level, exact or deployed, lies on the $\{2^t,\ \mu \cdot 2^t\}$ lattice (μ=$\sqrt{2}$, 1.5) of the FPoT grid, so one fused Triton kernel serves both grids' straight-through quantizers, while APoT's off-lattice level $10 = 1.25 \cdot 2^3$ requires a general nearest-level search (Appendix G).

4 Analysis

We derive where a fixed level budget should place its levels (§4.1) and how much of that allocation a shift-add datapath can realize (§4.2). The hardware cost of the grids selected is measured in §5.

4.1 Grid-Matching Analysis

Optimal point density. Let $u = \log_2 |w|$ with density f_U, and characterize a log-structured grid by its *point density* $\lambda(u)$: levels per octave, normalized so that $\int \lambda\, du = 1$. Under the high-resolution approximation the distortion of a symmetric N-magnitude grid and its minimizing point density are

$$D \;=\; \frac{(\ln 2)^2}{12N^2} \int f_U(u)\, \frac{2^{2u}}{\lambda(u)^2}\, du, \qquad \lambda^\star(u) \;\propto\; \left[f_U(u)\, 2^{2u}\right]^{1/3}, \tag{3}$$

the classical point-density result (Panter & Dite, 1951; Gersho, 1979) transported to the log axis (derivation in Appendix B.1). The factor $2^{2u/3}$ carries the design consequence. A fixed relative error costs more in absolute terms at large magnitude, so $\lambda^\star$ is stationary not at the mode of f_U but on its *upper flank*, $+0.79$ octaves above the weight standard deviation σ_W for a Gaussian source ($+0.795$ measured on Llama-3-8B, Fig. 1a, Appendices B.2 and C). The bottom of the grid should therefore be sparse and the top extended, and no single step ratio complies, so a log-uniform grid is mismatched in *shape* at any k.

Search and chain. An exhaustive search over the schedule family at b=4, each grid at its MSE-optimal scale on $W \sim \mathcal{N}(0,1)$, returns $[1, 2^3]$ (Fig. 1b) at $0.929\times$ the INT4 MSE. Its one-adder image $\{1, 2, 3, 4, 6, 8, 12\}$, the deployed DMPoT-4 grid, reaches $0.984\times$ (best of 5999 grids on its lattice) against APoT's $0.939\times$, and $[1, 2^3]$ cuts distortion 19% below a fixed-ratio control matched to it in span and zero fraction. "Density-matched" buys level *allocation*, not range (Table 3, Appendices B.3 and B.9). Within the near-INT band these figures rank grids rather than predict accuracy (Appendix B.10). Repeating the search at every bit-width gives optima that nest into a chain of level sets with peak density $2^{\,b-3}$ through $[1, 2^3]$ (Table 4), along which the proxy MSE relative to INT_b falls monotonically ($0.929 \to 0.569$, Fig. 4) while the absolute error removed vanishes (Table 4).

4.2 The Adder-Budget Bound

The family is realized by approximating each in-octave ratio with shift-adds on the activation. Under a budget of n adders per weight the realizable ratios in $[1, 2)$ are the signed sums of n+1 powers of two. For n=1 the set has a closed form and holds exactly one element, 1.5, in the middle third, so nearest projection of the geometric targets $2^{j/k}$ is injective only up to k=5, with worst-case ratio error 6.1% (Appendix B.7). Set against the chain's peak density $2^{\,b-3}$, this is the first of two bit-width laws: a one-adder budget supports $k \le 5$, hence $b \le 5$. Beyond b=5 the in-octave multiply needs a small multiplier that no single adder realizes, a different design point studied for accuracy and area (§6.3, Appendix F). The budget also fixes the deployed shape. One adder admits the ratio set $\{1 + m/4\}$, so $[1, 2^3]$ lands on $\{1, 2, 3, 4, 6, 8, 12\}$ and the log-uniform member on $\{1, 1.5, 2, 3, 4, 6, 8\}$ (Appendices B.7 and B.9). The budget counts adders *per weight*, not the shared products a folded lane precomputes, which is what keeps the five-bit alphabet $\mathcal{Q}$=$\{1, 3, 5, 7\}$ one-adder realizable (Appendix E). The second law fixes MSE instead of the adder budget. Among schedules whose deployed grid reaches MSE $\le$ MSE(INT_b), the cheapest datapath is the chain member's, an odd core w=b−2 bits wide (Table 4, last row), and no narrower core suffices at any b searched (one bit narrower gives $\ge 1.98\times$ INT_b, Appendix B.5).

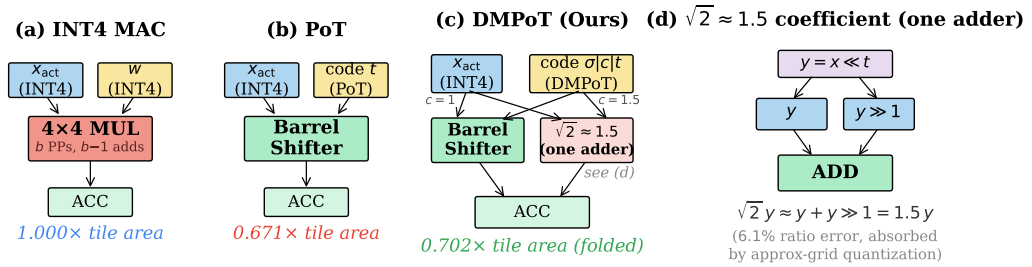

Figure 2: Per-PE weight datapaths: (a) uniform INT multiplier, (b) shift-only PoT, (c) DMPoT with the $\sqrt{2}\approx1.5$ coefficient path, (d) its one-adder realization. Areas are relative to the uniform INT tile (W4A4, Table 10), for (c) the folded tile, which forms $3x_i$ per lane at the array edge (Fig. 8b).

5 Hardware Cost

This section prices the grids selected in §4 as synthesized MAC-tile area. Figure 2 shows the datapaths.

From $\sqrt{2}$ to a multiplier-free element. The exact grid's in-octave ratio $\sqrt{2}$ has no shift-add form, but its one-adder image $1.5 = 1 + 2^{-1}$ does (Fig. 2d). Quantizing to the deployed set lets calibration absorb the 6.1% substitution (§3). Once its adder is folded out of the array (below), a DMPoT processing element is a barrel shifter, a 2:1 coefficient multiplexer and a signed accumulate (Fig. 8b), where the INT element carries a multiplier (Fig. 2a).

Folding the adder out of the array. Writing $w_{ji} = \sigma_{ji}\, s_j\, c_{ji}\, 2^{t_{ji}}$ for the weight linking activation lane i to output channel j (sign $\sigma_{ji} \in \{\pm1\}$, coefficient $c = 1 + \beta/2$, $\beta \in \{0, 1\}$, §3), the projection is

$$y_j \;=\; \sum_{i=1}^{N} w_{ji}\, x_i \;=\; s_j \sum_{i=1}^{N} \sigma_{ji}\, 2^{t_{ji}-\beta_{ji}}\, \underbrace{\big((1+2\beta_{ji})\, x_i\big)}_{\in\,\{\,x_i,\;3x_i\,\}}, \qquad 3x_i = x_i + (x_i \ll 1). \tag{4}$$

The coefficient bit selects x_i or the single sum $3x_i$, the latter shifted by $t-1$ rather than t. The sum $3x_i$ depends on the activation lane i and not on the output channel j, so it is formed once at the array edge and shared by all M output channels. The adder count falls from $N \cdot M$ to N. The saving over the order-only tile is positive by $M{=}4$ and saturates by $M{=}32$, and the ratio to INT stays flat in N (Table 13, Fig. 8). The same fold applies to the FPoT grid, so the two members are priced at near-identical silicon and compared on accuracy (§6.2).

Tile area. We synthesize complete $N \times M$ tiles with Yosys 0.68 (Wolf & Glaser, 2013) and ABC against the open sky130 library (SkyWater Technology & Google, 2020), reporting unrouted, untimed combinational gate-equivalent area relative to the uniform INT tile (a behavioral multiplier) at the same (N, M). At $(32, 32)$ the folded DMPoT tile is, with FPoT's, the cheapest datapath among those whose accuracy is near-lossless against uniform INT. Its area is $0.702\times$ and $0.618\times$ of INT at W4A4 and W4A8, below APoT's cheapest implementation (0.759/0.679). Only shift-only PoT is smaller, and its accuracy is near-lossless in none of the fourteen cells (§6.2). Every element but shift-only PoT's is SAT-proved equivalent to its golden level-times-activation product, so each integer core is bit-exact and the tiles are compared on area alone (Table 10, activation interface in Appendix A.1).

6 Experimental Results

We report results on seven models (§6.2), the bit-width sweep (§6.3), and the ablations (§6.4).

6.1 Experimental Setup

For the main results we use the OSTQuant framework (Hu et al., 2025). Rotation parameters are jointly optimized with smooth scaling factors via Riemannian gradient descent for 400 steps, with straight-through estimator (STE) gradients taken through the weight grid (for DMPoT, its exact form), and final weights obtained via GPTQ (Frantar et al., 2023) with the deployed grid as the target.

Table 1: W4A4KV4 and W4A8KV8 results on seven models (mean±std over three calibration seeds, DMPoT row in bold). Every grid except LogART is co-designed with its own preprocessing. LogART (†) is a post-training transplant on the uniform INT preprocessing (§6.2). PPL: WikiText-2 (↓). 0-shot[9]: nine-task mean (↑, %). Wider bit-widths are in Table 14.

		W4A4KV4		W4A8KV8	
Model	Weight Quant	PPL↓	0-shot[9]↑	PPL↓	0-shot[9]↑
Llama-3-8B	Uniform	$7.358_{\pm.074}$	$65.22_{\pm.37}$	$6.601_{\pm.011}$	$67.50_{\pm.10}$
	PoT	$8.786_{\pm.037}$	$59.55_{\pm.58}$	$7.642_{\pm.016}$	$63.95_{\pm.70}$
	APoT	$7.410_{\pm.027}$	$65.30_{\pm1.10}$	$6.673_{\pm.010}$	$67.76_{\pm.49}$
	LogART†	$16.05_{\pm.50}$	$50.02_{\pm1.91}$	$11.34_{\pm.66}$	$57.72_{\pm1.19}$
	FPoT	$7.596_{\pm.020}$	$64.78_{\pm.55}$	$6.849_{\pm.007}$	$67.30_{\pm.46}$
	DMPoT (Ours)	$\mathbf{7.417}_{\pm.041}$	$\mathbf{65.09}_{\pm.17}$	$\mathbf{6.672}_{\pm.013}$	$\mathbf{67.31}_{\pm.34}$
Llama-2-7B	Uniform	$5.971_{\pm.009}$	$62.11_{\pm.29}$	$5.630_{\pm.035}$	$63.84_{\pm.22}$
	PoT	$6.930_{\pm.004}$	$58.01_{\pm.12}$	$6.273_{\pm.026}$	$61.42_{\pm.67}$
	APoT	$6.012_{\pm.016}$	$61.81_{\pm.18}$	$5.666_{\pm.013}$	$64.06_{\pm.09}$
	LogART†	$17.81_{\pm5.19}$	$46.41_{\pm2.87}$	$11.13_{\pm2.21}$	$51.45_{\pm3.13}$
	FPoT	$6.114_{\pm.042}$	$61.89_{\pm.21}$	$5.751_{\pm.019}$	$63.75_{\pm.22}$
	DMPoT (Ours)	$\mathbf{6.042}_{\pm.013}$	$\mathbf{62.06}_{\pm.21}$	$\mathbf{5.677}_{\pm.012}$	$\mathbf{64.02}_{\pm.39}$
Llama-2-13B	Uniform	$5.260_{\pm.007}$	$65.61_{\pm.18}$	$5.032_{\pm.005}$	$67.08_{\pm.31}$
	PoT	$5.904_{\pm.016}$	$62.72_{\pm.46}$	$5.480_{\pm.005}$	$65.21_{\pm.27}$
	APoT	$5.291_{\pm.004}$	$65.26_{\pm.40}$	$5.037_{\pm.016}$	$66.64_{\pm.08}$
	LogART†	$9.019_{\pm.595}$	$49.86_{\pm2.44}$	$7.037_{\pm.516}$	$59.45_{\pm2.08}$
	FPoT	$5.366_{\pm.017}$	$65.22_{\pm.13}$	$5.106_{\pm.017}$	$66.59_{\pm.05}$
	DMPoT (Ours)	$\mathbf{5.305}_{\pm.016}$	$\mathbf{65.16}_{\pm.21}$	$\mathbf{5.065}_{\pm.023}$	$\mathbf{66.94}_{\pm.26}$
Qwen2.5-1.5B	Uniform	$11.10_{\pm.14}$	$38.74_{\pm.39}$	$9.753_{\pm.028}$	$42.74_{\pm.59}$
	PoT	$13.23_{\pm.11}$	$37.20_{\pm.81}$	$11.35_{\pm.09}$	$39.28_{\pm.41}$
	APoT	$11.03_{\pm.03}$	$39.95_{\pm.52}$	$9.859_{\pm.033}$	$42.84_{\pm.78}$
	LogART†	$12.16_{\pm.12}$	$38.58_{\pm.62}$	$10.30_{\pm.06}$	$41.74_{\pm.71}$
	FPoT	$11.23_{\pm.03}$	$39.89_{\pm.42}$	$10.00_{\pm.02}$	$42.40_{\pm.95}$
	DMPoT (Ours)	$\mathbf{10.95}_{\pm.01}$	$\mathbf{39.35}_{\pm.71}$	$\mathbf{9.840}_{\pm.026}$	$\mathbf{43.15}_{\pm.38}$
Qwen2.5-7B	Uniform	$7.559_{\pm.007}$	$66.40_{\pm.49}$	$7.043_{\pm.007}$	$68.06_{\pm.13}$
	PoT	$8.224_{\pm.013}$	$62.87_{\pm.60}$	$7.562_{\pm.021}$	$65.51_{\pm.19}$
	APoT	$7.548_{\pm.007}$	$66.17_{\pm.15}$	$7.065_{\pm.007}$	$68.53_{\pm.18}$
	LogART†	$7.936_{\pm.040}$	$64.14_{\pm.36}$	$7.303_{\pm.017}$	$67.26_{\pm.47}$
	FPoT	$7.616_{\pm.023}$	$66.34_{\pm.31}$	$7.140_{\pm.010}$	$68.60_{\pm.55}$
	DMPoT (Ours)	$\mathbf{7.518}_{\pm.007}$	$\mathbf{66.49}_{\pm.13}$	$\mathbf{7.047}_{\pm.008}$	$\mathbf{68.16}_{\pm.05}$
Qwen2.5-14B	Uniform	$6.330_{\pm.012}$	$68.67_{\pm.35}$	$5.649_{\pm.005}$	$71.11_{\pm.37}$
	PoT	$7.211_{\pm.048}$	$64.80_{\pm1.14}$	$6.476_{\pm.046}$	$68.75_{\pm.32}$
	APoT	$6.301_{\pm.001}$	$69.02_{\pm.40}$	$5.657_{\pm.007}$	$71.62_{\pm.23}$
	LogART†	$7.100_{\pm.011}$	$65.82_{\pm.84}$	$6.294_{\pm.025}$	$68.31_{\pm1.20}$
	FPoT	$6.380_{\pm.040}$	$68.93_{\pm.34}$	$5.845_{\pm.077}$	$71.28_{\pm.76}$
	DMPoT (Ours)	$\mathbf{6.280}_{\pm.015}$	$\mathbf{69.38}_{\pm.33}$	$\mathbf{5.662}_{\pm.005}$	$\mathbf{71.68}_{\pm.33}$
Mistral-7B	Uniform	$5.616_{\pm.008}$	$66.54_{\pm.26}$	$5.405_{\pm.015}$	$67.82_{\pm.12}$
	PoT	$6.000_{\pm.031}$	$64.57_{\pm.40}$	$5.694_{\pm.015}$	$65.46_{\pm.69}$
	APoT	$5.624_{\pm.019}$	$66.69_{\pm.09}$	$5.415_{\pm.011}$	$67.81_{\pm.26}$
	LogART†	$6.108_{\pm.045}$	$63.72_{\pm.08}$	$5.649_{\pm.033}$	$66.14_{\pm.84}$
	FPoT	$5.708_{\pm.048}$	$66.06_{\pm.52}$	$5.486_{\pm.038}$	$67.31_{\pm.41}$
	DMPoT (Ours)	$\mathbf{5.626}_{\pm.005}$	$\mathbf{66.86}_{\pm.19}$	$\mathbf{5.417}_{\pm.016}$	$\mathbf{67.70}_{\pm.21}$

DMPoT is agnostic to this choice of preprocessing. Portability to DuQuant is validated on three of the seven models, within 0.02 PPL of uniform INT and below FPoT in all six cells (Appendix J).

Our default configuration is **DMPoT-4**, the schedule $[1, 2^3]$ deployed in its one-adder image $\{1, 2, 3, 4, 6, 8, 12\}$ with 16-offset (G=16, Appendix A.4) weight grid-alignment (all §3). We report W4A4KV4 and W4A8KV8 on seven models (fourteen cells), and a weight bit-width sweep (§6.3). The models are Llama-3-8B (Grattafiori et al., 2024), Llama-2-7B/13B (Touvron et al., 2023), Qwen2.5-1.5B/7B/14B (Yang et al., 2024) and Mistral-7B (Jiang et al., 2023). Appendix L gives the full settings. GPTQ calibrates on 128 WikiText-2 training windows of $2{,}048$ tokens.

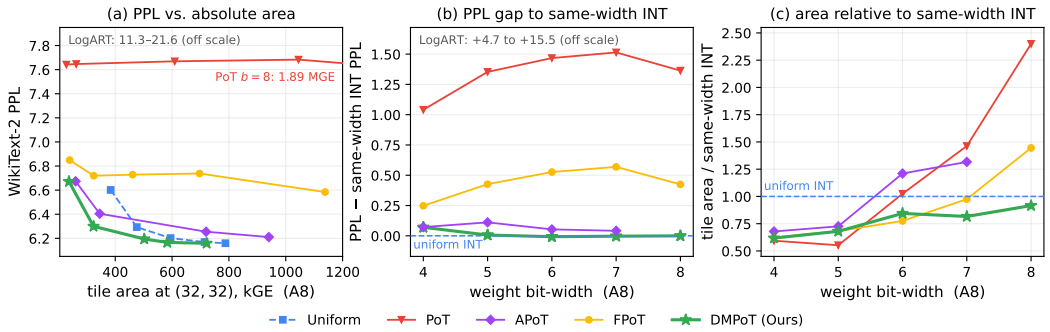

Figure 3: Weight bit-width sweep, Llama-3-8B at A8KV8 (Tables 14, 10 and 15). (a) PPL against area, (b) PPL gap to same-width INT, and (c) area relative to same-width INT. LogART shares FPoT's tile and is off scale in (a) and (b), PoT at b=8 runs off frame, and APoT is undefined at b=8.

We evaluate WikiText-2 (Merity et al., 2017) perplexity (PPL↓) and the unweighted mean of accuracy over the nine zero-shot benchmarks of Appendix L (0-shot[9]↑, in %). All zero-shot evaluations are conducted via the lm-eval-harness framework (Gao et al., 2024). Per-task breakdowns are given in Appendix I. Unless a caption says otherwise, every ± is the *sample* standard deviation over three calibration seeds (42, 0, 123), and gaps are absolute differences of three-seed means.

6.2 Main Results

Table 1 compares DMPoT against the uniform INT baseline, basic PoT, FPoT (the J=0 member of the same family, §3), and APoT (Li et al., 2020). Each grid with a straight-through target is co-designed with its own preprocessing. The learned rounding of LogART (Xu et al., 2026) is too costly to repeat at every straight-through step, so its rows are a post-training transplant on the uniform INT preprocessing, a reference point rather than an upper bound (marked †, protocol in Appendix A.5).

DMPoT reaches uniform INT accuracy. Averaged over the seven models, the perplexity gap between DMPoT and the uniform INT baseline is -0.008 at W4A4KV4 and $+0.038$ at W4A8KV8, with a per-cell range of -0.15 to $+0.09$. Zero-shot accuracy tracks INT as closely, never more than 0.5%p below INT in any cell. Every cell satisfies the near-lossless criterion of the introduction at a weight datapath that contains no multiplier (cell-level readings in Appendix I). Basic PoT, by contrast, costs $+1.43$ PPL and 5.7%p of zero-shot accuracy relative to INT on Llama-3-8B W4A4KV4, where DMPoT reads $7.417_{\pm.041}$, $+0.06$ PPL above INT and 0.1%p below in zero-shot.

The log-uniform member at near-identical hardware. DMPoT and FPoT are two members of the same family with the same number of levels. They compile to the *same folded datapath* and occupy $0.702\times$ and $0.695\times$ INT MAC area at W4A4 (§5, Table 10), yet DMPoT has the lower perplexity in *all fourteen* cells, 7.417 against FPoT's 7.596 on Llama-3-8B W4A4KV4, and leads by 0.14 to 0.35 PPL when both grids share any one of five transforms (Appendix H.2). The one difference, J=1 versus J=0, is whether the bottom of the grid is sparsened to match the point density that §4 predicts. The measured ordering agrees with the grids' unweighted MSE (0.984 versus $1.055\times$ INT4).

The Pareto claim against APoT. Against a co-designed APoT the perplexity comparison is a tie. The mean gap over the fourteen cells is -0.005 PPL in DMPoT's favor (ahead in six, behind in eight), with no cell separated by more than 0.08 PPL. Both grids fold their in-octave products to the array edge, but APoT's additive level ($10 = 2^3+2^1$) gives a three-element in-octave alphabet against DMPoT's two ($\mathcal{Q}$=$\{1,3\}$), and its extra shared lane products consume what the fold saves (Appendix E.2). As a result, the tile is $0.702\times$ INT against $0.759\times$ for APoT's cheapest implementation (Fig. 1c), 7% less MAC area at matched perplexity and zero-shot accuracy (7–10% across the three Yosys builds, 4–10% across the four libraries, Appendix E; generation in Appendix K). Their unweighted MSE leans APoT's way (0.984 against 0.939) by a margin the fourteen cells do not resolve (§4, Appendix C).

6.3 The Five-Bit Boundary of the Bit-Width Sweep

We sweep the weight bit-width from four to eight bits at fixed A8KV8 on Llama-3-8B (Table 14, Appendix F), each width preprocessed and quantized on its own (Appendix B.4 covers serving one quantized model at a lower bit-width). DMPoT converges to uniform INT as the budget grows, from $+0.07$ PPL at $b{=}4$ to 0.00 from $b{=}7$, every bit-width within the near-lossless criterion, while the fixed-density grids stall (FPoT is $+0.25$ PPL over uniform INT at $b{=}4$ and $+0.42$ to $+0.57$ from $b{=}5$, Fig. 3b). Added bits buy them range, not resolution, unless the density grows with the budget, the chain's defining property (the LogART row's wrong-way trend is a transplant artifact, Appendix A.5). On the hardware side the deployed DMPoT datapath stays below the same-width INT tile at every bit-width, with its ratio to the INT tile rising from 0.618 at $b{=}4$ to 0.916 at $b{=}8$ (Fig. 3c). The area margin and the accuracy gained per added bit shrink together, locating the interesting regime at $b{\leq}5$.

The five-bit member. $b{=}5$ is the *deployable boundary* set by the one-adder *per-weight* budget (§4.2). The deployed grid is the chain member $[1, 2, 4^3]$ ($\mathcal{Q}{=}\{1, 3, 5, 7\}$), DMPoT-5, whose folded tile is priced in Table 10 and whose sweep accuracy is PPL $6.30_{\pm.00}$ against uniform INT5's $6.29_{\pm.00}$.

The area boundary. The folded five-bit tile undercuts INT5 at eight-bit activations ($0.680\times$ at W5A8 and $0.682\times$ for the log-uniform $k{=}2$ member at 0.42 higher PPL, FPoT 6.72 against 6.30, Tables 10 and 14), and the five-bit grid ordering is toolchain- and library-dependent, with no claim resting on it (Appendix E). Four bits is the efficiency sweet spot. Five bits is an accuracy–area trade (Fig. 3), and beyond it the chain is the one grid below same-width INT at *every* bit-width, the log-uniform $k{=}2$ grid falling behind at $b{=}8$, and shift-only PoT and APoT from $b{=}6$ (Appendix F.1).

6.4 Additional Ablations

The full ablations are in Appendix H. Both preprocessing components (rotation and scaling) are necessary, with rotation acting as the larger lever (Appendix H.5). The zero threshold, the rounding mode, and the calibration set each move perplexity by at most 0.084 PPL (Appendices H.4 and H.1). The forced $\sqrt{2} \to 1.5$ substitution is validated end to end in Appendix H.3. Approximate-grid quantization stays within 0.024 PPL of the exact-$\sqrt{2}$ reference, while rounding on the exact grid costs $+0.090$ PPL more than approximate-grid quantization. A five-axis generation suite on two instruction-tuned models finds DMPoT never more than 2.2%p behind uniform INT (Appendix K).

7 Conclusion

We introduced DMPoT, the density-matched extension of the fractional power-of-two family: logarithmic weight grids whose level spacing follows the MSE-optimal point density of the weights rather than a fixed ratio, deployable at the four- and five-bit widths a one-adder budget admits. Across seven models at four bits and a four-to-eight-bit sweep on Llama-3-8B, DMPoT is near-lossless against uniform INT. Its 4-bit member runs on a multiplier-free datapath at $0.702\times$ the W4A4 INT MAC-tile area and has lower perplexity than FPoT in all fourteen cells at near-identical hardware. These results move the design question for shift-based LLM inference from "how fine should the logarithmic grid be" to "where should its levels go, and which grids can share the hardware that realizes them", placing DMPoT on the accuracy–area Pareto front rather than strictly above prior grids in accuracy.

Limitations and future work. Hardware cost is standard-cell area from tile synthesis, normalized to the toolchain's behavioral integer multiplier (Appendix E). Routing, timing closure, and power characterization remain outside the synthesis scope (Appendix E.4). Experiments run on a single 80 GB GPU, so the models here span 1.5B to 14B. The density fixes the grid's shape and the discrete search its schedule, both under the Gaussian proxy measured after preprocessing (Appendices B.10 and C). Optimizing a grid directly against a measured density would replace that search. Each sweep member is calibrated at its own bit-width. Calibrating the chain so that a single quantized model serves every bit-width would turn its nesting into progressive precision (Appendix B.4).

AI USE STATEMENT

We used generative AI tools in the following ways. *Writing*: drafting parts of the text and polishing sentences for readability. *Retrieval and discovery*: an initial survey of related work. *Research execution*: automating experiment runs and collating their results. *Mathematical claims*: assisting the authors by verifying their derivations. All AI-assisted text and code were reviewed by the authors, and we take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Multiplier-free LLM inference lowers the arithmetic cost of large-scale deployment and can make capable models practical on constrained hardware; we measure this as silicon area only, and do not claim a corresponding reduction in power. Cheaper inference may equally lower the barrier to harmful applications. This work introduces no new capability: it applies to existing public models and, at best, preserves their behavior at slightly degraded accuracy. It studies post-training quantization of publicly available large language models on public benchmarks, and involves no new data collection and no human subjects.

REPRODUCIBILITY STATEMENT

The experimental protocol (models, bit-widths, calibration data, seeds, and hyperparameters) is described in §6 and Appendix L, with the per-model and per-task results in Table 1 and Appendices I and J; formal statements and proofs are in Appendix B, and the synthesis flow, the cost model with its assumptions, and the build and library cross-checks are in Appendix E. The supplementary material contains the anonymized source code, the synthesis harness with its committed per-cell measurements, and the analysis scripts named in Appendices B and E, which recompute their tables as CSV, together with the anonymized FPoT manuscript cited as Anonymous (2026). The accuracy tables are transcribed from the run logs the code writes.

REFERENCES

Anonymous. Fractional power-of-two quantization for efficient and effective multiplier-free LLM inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026. To appear. Anonymized submission version included in the supplementary material.

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)*, volume 37, pp. 100213–100240, 2024.

Liza Babaoglu, Shuangyi Chen, and Ashish Khisti. Multi-bitwidth quantization for LLMs using additive codebooks, 2026. arXiv:2606.12876.

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*, volume 34, pp. 7432–7439, 2020.

Hugh Brunk and Nariman Farvardin. Fixed-rate successively refinable scalar quantizers. In *Proceedings of Data Compression Conference (DCC)*, pp. 250–259, 1996.

Brian Chmiel, Liad Ben-Uri, Moran Shkolnik, Elad Hoffer, Ron Banner, and Daniel Soudry. Neural gradients are near-lognormal: Improved quantized and sparse training. In *International Conference on Learning Representations (ICLR)*, 2021.

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 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 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.

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)*, volume 35, pp. 30318–30332, 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.

Mostafa Elhoushi, Zihao Chen, Farhan Shafiq, Ye Henry Tian, and Joey Yiwei Li. DeepShift: Towards multiplication-less neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 2359–2368, 2021.

William H. R. Equitz and Thomas M. Cover. Successive refinement of information. *IEEE Transactions on Information Theory*, 37(2):269–275, 1991. doi: 10.1109/18.75242.

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

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. The Language Model Evaluation Harness, 7 2024. URL `https://zenodo.org/records/12608602`.

Allen Gersho. Asymptotically optimal block quantization. *IEEE Transactions on Information Theory*, 25(4):373–380, 1979. doi: 10.1109/TIT.1979.1056067.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Cong Guo, Chen Zhang, Jingwen Leng, Zihan Liu, Fan Yang, Yunxin Liu, Minyi Guo, and Yuhao Zhu. ANT: Exploiting adaptive numerical data type for low-bit deep neural network quantization. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 1414–1433, 2022.

John L. Gustafson and Isaac T. Yonemoto. Beating floating point at its own game: Posit arithmetic. *Supercomputing Frontiers and Innovations*, 4(2):71–86, 2017. doi: 10.14529/jsfi170206.

Xing Hu, Yuan Cheng, Dawei Yang, Zukang Xu, Sifan Zhou, Jiangyong Yu, Chen Xu, Zhixuan Chen, Zhe Jiang, and Zhihang Yuan. OSTQuant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting. In *International Conference on Learning Representations (ICLR)*, 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

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.

Xinlin Li, Bang Liu, Rui Heng Yang, Vanessa Courville, Chao Xing, and Vahid Partovi Nia. DenseShift: Towards accurate and efficient low-bit power-of-two quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17010–17020, 2023.

Yuhang Li, Xin Dong, and Wei Wang. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. DuQuant: Distributing outliers via dual transformation makes stronger quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pp. 87766–87800, 2024a.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. *Proceedings of Machine Learning and Systems*, 6: 87–100, 2024b.

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)*, 2025.

Yuanyong Luo, Zhongxing Zhang, Richard Wu, Hu Liu, Ying Jin, Kai Zheng, Minmin Wang, Zhanying He, Guipeng Hu, Luyao Chen, Tianchi Hu, Junsong Wang, Minqi Chen, Mikhaylov Dmitry, Korviakov Vladimir, Bobrin Maxim, Yuhao Hu, Guanfu Chen, and Zeyi Huang. Ascend HiFloat8 format for deep learning. *arXiv preprint arXiv:2409.16626*, 2024.

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 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Pranav Ajit Nair, Puranjay Datta, Jeff Dean, Prateek Jain, and Aditya Kusupati. Matryoshka quantization. In *International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45484–45506, 2025.

NVIDIA, Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, Stefania Alborghetti, Michael Andersch, Sivakumar Arayandi, Alexis Bjorlin, Aaron Blakeman, Evan Briones, Ian Buck, Bryan Catanzaro, Muya Chang, Jinhang Choi, Mike Chrzanowski, Eric Chung, Victor Cui, Steve Dai, Bita Darvish Rouhani, Carlo del Mundo, Deena Donia, Burc Eryilmaz, Henry Estela, Abhinav Goel, Oleg Goncharov, Yugi Guvvala, Robert Hesse, Russell Hewett, Herbert Hum, Ujval Kapasi, Brucek Khailany, Mikail Khona, Nick Knight, Alex Kondratenko, Ronny Krashinsky, Ben Lanir, Simon Layton, Michael Lightstone, Daniel Lo, Paulius Micikevicius, Asit Mishra, Tim Moon, Deepak Narayanan, Chao Ni, Abhijit Paithankar, Satish Pasumarthi, Ankit Patel, Mostofa Patwary, Ashwin Poojary, Gargi Prasad, Sweta Priyadarshi, Yigong Qin, Xiaowei Ren, Oleg Rybakov, Charbel Sakr, Sanjeev Satheesh, Stas Sergienko, Pasha Shamis, Kirthi Shankar, Nishant Sharma, Mohammad Shoeybi, Michael Siu, Misha Smelyanskiy, Darko Stosic, Dusan Stosic, Bor-Yiing Su, Frank Sun, Nima Tajbakhsh, Shelby Thomas, Przemek Tredak, Evgeny Tsykunov, Gandhi Vaithilingam, Aditya Vavre, Rangharajan Venkatesan, Roger Waleffe, Qiyu Wan, Hexin Wang, Mengdi Wang, Lizzie Wei, Hao Wu, Evan Wu, Keith Wyss, Ning Xu, Jinze Xue, Charlene Yang, Yujia Zhai, Ruoxi Zhang, Jingyang Zhu, and Zhongbo Zhu. Pretraining large language models with NVFP4, 2026. URL https://arxiv.org/abs/2509.25149.

Open Compute Project. OCP microscaling formats (MX) specification, version 1.0. https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf, 2023.

Philip F. Panter and Ward Dite. Quantization distortion in pulse-count modulation with nonuniform spacing of levels. *Proceedings of the IRE*, 39(1):44–48, 1951. doi: 10.1109/JRPROC.1951.230419.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan 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 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, 2016.

Gunho Park, Jeongin Bae, Beomseok Kwon, Byeongwook Kim, Se Jung Kwon, and Dongsoo Lee. AnyBCQ: Hardware efficient flexible binary-coded quantization for multi-precision LLMs. In *International Conference on Learning Representations (ICLR)*, 2026.

Yeonhong Park, Jake Hyun, SangLyul Cho, Bonggeun Sim, and Jae W. Lee. Any-precision LLM: Low-cost deployment of multiple, different-sized LLMs. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 39682–39701, 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.

Bita Darvish Rouhani, Ritchie Zhao, Ankit More, Mathew Hall, Alireza Khodamoradi, Summer Deng, Dhruv Choudhary, Marius Cornea, Eric Dellinger, Kristof Denolf, Dusan Stosic, Venmugil Elango, Maximilian Golub, Alexander Heinecke, Phil James-Roxby, Dharmesh Jani, Gaurav Kolhe, Martin Langhammer, Ada Li, Levi Melnick, Maral Mesmakhosroshahi, Andres Rodriguez, Michael Schulte, Rasoul Shafipour, Lei Shao, Michael Siu, Pradeep Dubey, Paulius Micikevicius, Maxim Naumov, Colin Verrilli, Ralph Wittig, Doug Burger, and Eric Chung. Microscaling data formats for deep learning, 2023. URL `https://arxiv.org/abs/2310.10537`.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4463–4473, 2019.

SkyWater Technology and Google. SkyWater open source PDK: sky130. `https://github.com/google/skywater-pdk`, 2020.

Philippe Tillet, H. T. Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (MAPL)*, pp. 10–19, 2019. doi: 10.1145/3315508.3329973.

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

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)*, volume 30, 2017.

Xinyu Wang, Vahid Partovi Nia, Peng Lu, Jerry Huang, Xiao-Wen Chang, Boxing Chen, and Yufei Cui. PoTPTQ: A two-step power-of-two post-training for LLMs. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI)*, 2025. arXiv:2507.11959.

Clifford Wolf and Johann Glaser. Yosys – A free Verilog synthesis suite. In *Austrochip Workshop on Microelectronics*, pp. 47–52, 2013.

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)*, pp. 38087–38099. PMLR, 2023.

Jiawei Xu, Yi Zheng, Chenghe Sun, Taiyu Zhou, Zuqi Zhang, Jie Li, Lirong Zheng, and Zhuo Zou. LogART: Pushing the limit of efficient logarithmic post-training quantization. In *International Conference on Learning Representations (ICLR)*, 2026.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Haoran You, Yipin Guo, Yichao Fu, Wei Zhou, Huihong Shi, Xiaofan Zhang, Souvik Kundu, Amir Yazdanbakhsh, and Yingyan (Celine) Lin. ShiftAddLLM: Accelerating pretrained LLMs via post-training multiplication-less reparameterization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

A PRELIMINARIES

This appendix fixes the notation behind §3: uniform quantization as deployed for activations and the KV cache, the preprocessing stack the pipeline reuses, and two expansions of §3, the schedule notation and the weight grid-alignment protocol.

A.1 UNIFORM QUANTIZATION

Standard uniform (linear) quantization maps a floating-point tensor to a finite set of evenly-spaced levels. Activations and the KV cache are quantized dynamically with the asymmetric b-bit rule

$$Q_{\text{lin}}(x) = \big(\text{clamp}(\lfloor x/s \rceil + z,\ 0,\ 2^b{-}1) - z\big)\, s, \qquad s = \frac{\gamma\,(x_{\max} - x_{\min})}{2^b - 1}, \qquad z = \Big\lfloor \frac{-\gamma\, x_{\min}}{s} \Big\rceil, \tag{5}$$

where $x_{\max} \geq 0$ and $x_{\min} \leq 0$ bound one quantization group (its range extended to include zero) and $\gamma \leq 1$ is a clipping ratio ($\gamma{=}1$ for every quantizer other than the two normalization outputs). A group is one token for the normalization outputs and the down-projection input, and 128 channels of one token, which is one attention head in every evaluated model, for the attention output and the key and value caches. In this work, activations and the KV cache use uniform INT quantization in all configurations, while weights use logarithmic (DMPoT) quantization (§3).

Deployment form. The zero-point z is set per group at run time, and the MAC tiles of §5 take the signed code $u_i = q_i - 2^{b-1}$ rather than the unsigned code $q_i = \text{clamp}(\lfloor x_i/s \rceil + z, 0, 2^b{-}1)$ of Eq. (5). The signed code u_i is q_i with its most significant bit inverted, a b-bit two's-complement word. With $k_{ji} = w_{ji}/s_j$ the signed integer weight level of Eq. (4), a tile accumulates $\sum_i k_{ji} u_i$, and since $q_i - z = u_i + (2^{b-1} - z)$ the quantized product is recovered exactly as

$$y_j \;=\; s_j\, s \Big[\sum_i k_{ji}\, u_i \;+\; (2^{b-1} - z)\, K_j \Big], \qquad K_j = \sum_i k_{ji},$$

where K_j is a constant of the quantized weights, one per output channel (one per output channel and group when z is set per group). The correction is one multiply-add per output channel and group, applied with the per-channel scale outside the tile. It takes the same form for every weight grid, uniform INT included, so Appendix E leaves it out of every tile alike. The zero-points of the key and value caches enter the attention arithmetic, which lies outside the tiles (Appendix E.4).

A.2 STATE-OF-THE-ART LLM PTQ PREPROCESSING

Recent LLM PTQ preprocessing methods (QuaRot (Ashkboos et al., 2024), SpinQuant (Liu et al., 2025), OSTQuant (Hu et al., 2025), DuQuant (Lin et al., 2024a), with the taxonomy in §2) rewrite each linear operation as $Y = XW = (XT)(T^{-1}W)$ with an invertible transform T (orthogonal rotation, per-channel scaling, or their composition), which redistributes channel-concentrated outliers and reduces kurtosis before quantization. We primarily build on OSTQuant, which jointly trains orthogonal rotations and per-channel smoothing scales via Riemannian gradient descent on the Stiefel manifold (400 steps with STE). The learned rotations and scales are fused into the weights. What remains at inference is a fixed Hadamard transform, 14,336-dimensional before each down-projection and 128-dimensional on every query and key head, evaluated with fast Hadamard routines. It costs $\approx$0.57 M additions per layer against 218 M weight MACs on Llama-3-8B, $\approx 0.26\%$. Our DMPoT method is compatible with any such preprocessing. We evaluate with OSTQuant and confirm DuQuant portability in Appendix J.

A.3 SCHEDULE NOTATION, EXPANDED

This block expands the schedule notation of §3. The schedule separates *how finely* the grid resolves from *where* it does so. The density list fixes the resolution of each octave, and the position of the dense pieces fixes where that resolution is spent. A sparse prefix spends the bottom of the range coarsely and concentrates resolution on the upper flank of the weight distribution, the regime the derivation of §4 selects. A constant density instead spends it log-uniformly, which recovers the prior PoT and FPoT grids. In octave i the exact grid places its k_i levels at $2^{\,i+j/k_i}$, $j < k_i$, and the deployed

grid at the multiples of $2^i/k_i$ in $[2^i, 2^{i+1})$. Table 5 gives each schedule's level count, odd cores and spacing regimes, and Figure 5 draws the deployed sets themselves.

A.4 WEIGHT GRID-ALIGNMENT PROTOCOL

This block expands the grid-alignment clause of §3. The alignment is a phase-shift search introduced with FPoT (Anonymous, 2026) and used here unchanged. It tries $G{=}16$ offsets $s_{\text{base}}2^{g/G}$, $g = 0, \ldots, G{-}1$, projects each onto the deployed level set, and folds the per-channel winner into the static scale. The deployed array is identical either way. Under approximate-grid quantization (§3), the per-channel scale search, this grid-alignment, and GPTQ's Hessian-weighted error compensation all see the deployed level set. That is what lets calibration absorb the one-adder substitution instead of leaving it as dequantization error.

A.5 BASELINE PROTOCOLS

Co-designed preprocessing. Every grid with a straight-through quantizer (Uniform, PoT, APoT, FPoT, DMPoT) is trained with its *own matched* preprocessing. The rotation and smoothing parameters are optimized with that column's weight grid in the STE loop (the exact grid for DMPoT, while the deployed levels enter at calibration, Appendix H.3). LogART's weight quantizer is itself an optimization, a Hessian-guided per-channel grid search followed by 500 iterations of learned rounding for every projection. One pass took about 12 hours for Llama-3-8B on one A100, against under two hours for a complete co-designed run including its 400-step transform training, and a straight-through loop would have to repeat it at every step.

The LogART rows of Tables 1 and 14 are therefore a transplant (§6.2). LogART (Xu et al., 2026) is designed for FP16 activations, while this paper evaluates every grid under uniform INT activations and an INT KV cache, so the two protocols share no operating point. We carry over LogART's weight grid, its per-channel grid search and its learned rounding and apply them, in place of this paper's scale search, grid alignment and GPTQ pass, to the uniform INT preprocessing checkpoint of the same configuration and seed without further transform training, so the LogART rows differ from the co-designed rows in the preprocessing as well as the weight quantizer. None of LogART's own hyper-parameters is tuned for this setting or re-tuned per bit-width in the sweep. The rows carry an across-seed spread far larger than any other method (up to ± 10.66 at $b{=}8$) and move the wrong way as the budget grows. We read both as properties of the transplant, not of the published method, and mark the rows † accordingly.

A.6 QUANTIZER PSEUDOCODE

Algorithm 1 states the complete DMPoT quantize/dequantize/STE pipeline. The grid is built once from the schedule. At calibration the per-channel scale comes from a clip search on the deployed levels, and each weight takes the nearest deployed level under the arithmetic-midpoint rule of §3. The transform's straight-through forward applies the same rule to the exact levels. The table below maps the level index m to the stored word. The magnitude bits are the coefficient bit and the shift field, the unused pattern $(c, t){=}(1.5, 0)$ is the zero code, and the gate $\neg\big(c \wedge \neg(t_1 \vee t_0)\big)$ suppresses the element's partial product for it. This gating is part of every synthesized tile and SAT-proved element of Appendix E.

Stored DMPoT-4 magnitude code (the sign is the fourth bit, and m is the level index of Algorithm 1)

magnitude bits $(c\,t_1t_0)$	000	001	010	011	100	101	110	111
coefficient c	1	1	1	1	1.5	1.5	1.5	1.5
shift t	0	1	2	3	0	1	2	3
level	1	2	4	8	0 (zero code)	3	6	12
index m	1	2	4	6	0	3	5	7

Algorithm 1 Signed DMPoT Quantization, Dequantization and Datapath (arithmetic-midpoint decision rule)

Require: Weights $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}}}$, bit budget b, schedule $[k_1^{J_1}, k_2^{J_2}, \dots]$ of Eq. (1), clip ratio $\alpha \in (0, 1]$

Ensure: Codes $\mathbf{M} \in \{0, \dots, 2^{b-1}-1\}$, signs $\mathbf{S} \in \{-1, 0, +1\}$, per-channel scales $\mathbf{s}$

// Grid (levels are data-independent; computed once)

$e_{\max} \leftarrow 2^{b-1} - 2$ ▷ b=4 → 6; b=5 → 14

$\mathcal{L} \leftarrow \left[2^{\varphi(e)} \right]_{e=0}^{e_{\max}}$, φ from Eq. (1) ▷ DMPoT-4: $[1, 2^3]$; DMPoT-5: $[1, 2, 4^3]$

$\mathcal{L}^{\text{dep}} \leftarrow$ one-adder projection of $\mathcal{L}$ ▷ DMPoT-4: $\{1, 2, 3, 4, 6, 8, 12\}$ (§3)

$\mathbf{t} \leftarrow$ arithmetic midpoints of $\mathcal{L}^{\text{dep}}$, prepended by $\frac{1}{2}\mathcal{L}_0^{\text{dep}}$ ▷ zero boundary $= s/2$

(log reading: $\tau = 2^{-1/(2k_1)}$; sparse bottom octave k_1=1 ⇒ 0.707; log-uniform k=2 ⇒ 0.841)

// Per-channel scale, clip search on the deployed grid

for each output channel j **do**

  $s_j \leftarrow \alpha_j^* \cdot \max_i |W_{j,i}| \,/\, \mathcal{L}_{e_{\max}}^{\text{dep}}$ ▷ α_j^* minimizes $\sum_i |\hat{w} - w|^{2.4}$ over row j

end for

// Quantize (nearest deployed level, arithmetic midpoints)

for each weight $w = W_{j,i}$ **do**

  $\tilde{w} \leftarrow |w|/s_j$; $e \leftarrow \text{searchsorted}(\mathbf{t}, \tilde{w})$ ▷ e=0 iff $\tilde{w} < \mathbf{t}_0$: zero code

  $m \leftarrow e$; $\sigma \leftarrow \text{sgn}(w) \cdot \mathbf{1}[m \neq 0]$

end for

// Dequantization (calibration)

$\hat{w} = \sigma \cdot s_j \cdot \mathcal{L}_{m-1}^{\text{dep}}$ for $m \geq 1$; $\hat{w} = 0$ for $m = 0$

Transform training (STE forward): the same rule on the exact $\mathcal{L}$, with $s_j = \max_i |W_{j,i}|/\mathcal{L}_{e_{\max}}$

// Inference datapath (odd-core form, §5)

Stored word: sign σ, coefficient bit $c \in \{1, 1.5\}$, shift field t; zero code (c, t)=$(1.5, 0)$; $m \mapsto (c, t)$: Appendix A.6

Per activation lane i (shared over all M output channels): $3x_i = x_i + (x_i \ll 1)$

Per PE: select x_i (c=1) or $3x_i$ (c=1.5), shift by t or $t-1$, sign, accumulate (integer tree)

Per output channel: multiply once by s_j

B Point-Density Derivation and Validation

This appendix derives the optimal log-domain point density used in §4.1, gives the closed forms for a Gaussian source, documents the schedule search (1) and the nested chain it returns, measures what it costs to serve a lower width from the stored eight-bit tensor, enumerates the realizable in-octave ratio sets behind the adder-budget bound of §4.2, and states the scope of the argument. Every proxy number in this appendix and in §4.1 is regenerated by `analysis/pointdensity.py`, `analysis/multipiece.py` and `analysis/gen_tables.py` in the supplementary material. The one end-to-end measurement this appendix reports (Table 6) follows the evaluation protocol of §6, and its runner, collector and per-seed output ship with the supplementary code.

B.1 Derivation of the optimal point density

Let $u = \log_2 |w|$ with density f_U, and let a log-structured grid be characterized by its *point density* $\lambda(u)$: the number of levels per octave, normalized so that $\int \lambda \, du = 1$. A symmetric N-magnitude grid then places $N\lambda(u)$ levels per octave at u. With local log-spacing $\Delta u(u) = 1/(N\lambda(u))$, a weight at log-magnitude u falls in a bin of *absolute* width

$$\Delta w \;=\; |w| \ln 2 \, \Delta u \;=\; \frac{2^u \ln 2}{N\lambda(u)}, \tag{6}$$

because a fixed ratio step becomes a proportionally larger absolute step at larger magnitude. Under the high-resolution approximation each bin contributes $\Delta w^2/12$ to the mean squared error, so integrating against f_U gives Eq. (3),

$$D \;=\; \frac{(\ln 2)^2}{12N^2} \int f_U(u)\, \frac{2^{2u}}{\lambda(u)^2}\, du. \tag{7}$$

Write $h(u) = f_U(u)2^{2u}$. By Hölder's inequality applied to $\int h\lambda^{-2}$ subject to $\int \lambda = 1$,

$$\int h\lambda^{-2}\, du \;\geq\; \Big(\int h^{1/3}\, du\Big)^3, \tag{8}$$

with equality if and only if $\lambda \propto h^{1/3}$, which is the second form of Eq. (3):

$$\lambda^\star(u) \;\propto\; \big[f_U(u)\, 2^{2u}\big]^{1/3} \;=\; f_U(u)^{1/3}\, 2^{2u/3}, \qquad D^\star = \frac{(\ln 2)^2}{12N^2}\Big(\int (f_U 2^{2u})^{1/3} du\Big)^3. \tag{9}$$

This is the classical point-density result of Panter & Dite (1951) and Gersho (1979) transported to the log axis. It is equivalent to the familiar form $\lambda_w \propto p(w)^{1/3}$, since substituting $\lambda_w = \lambda_u/(w \ln 2)$ and $p(w) = f_U/(w \ln 2)$ returns $\lambda_u \propto f_U^{1/3} w^{2/3}$. We derive it directly in the log domain because the design family is parameterized there.

Lemma 1 (Peak condition). *$\lambda^\star$ is stationary where* $\dfrac{d \log_2 f_U}{du} = -2$.

Lemma 1 follows immediately from

$$\frac{d}{du} \ln \lambda^\star \;=\; \frac{1}{3}\frac{d}{du} \ln f_U + \frac{2}{3} \ln 2 \;=\; 0. \tag{10}$$

Two remarks. (i) In magnitude space the same condition reads $S(w) = 3$, where $S = -\,d \log f / d \log w$ is the log-log slope of the value-domain density $f(w)$. The constant differs from the -2 above because f_U carries the Jacobian factor 2^u. (ii) The condition is stationarity, not maximality. It identifies the unique global peak whenever S crosses 3 exactly once, as it does for every source considered in this paper and for the measured density of Appendix C (single crossing, at $u = +0.795$). A multimodal or outlier-contaminated density can cross several times, and the global peak is then not determined locally.

The factor $2^{2u/3}$ is what separates this from "place levels where the mass is". A quantizer minimizing *absolute* error must shift levels toward large magnitudes, because a fixed *relative* error costs more in absolute terms there. The linear-grid mismatch follows from the same factor. A linear grid's log-domain density is $\propto 2^u$, which matches the doubling of $\lambda^\star$ on the lower flank but keeps doubling past the peak, where $\lambda^\star$ turns over.

B.2 Gaussian source

For $W \sim \mathcal{N}(0, \sigma_W^2)$ with $\sigma_W = 1$ without loss of generality, the log-magnitude density is

$$f_U(u) \;=\; 2 \ln 2 \cdot 2^u\, \varphi(2^u), \qquad \varphi = \text{standard normal pdf}, \tag{11}$$

with mode at $u = 0$ (that is, at $|w| = \sigma_W$) and exact moments

$$\mathbb{E}[U] = -\frac{\gamma + \ln 2}{2 \ln 2} = -0.9164 \text{ oct}, \qquad \sigma_U = \frac{\pi}{2\sqrt{2}\, \ln 2} = 1.6024 \text{ oct}, \tag{12}$$

where γ is the Euler–Mascheroni constant. Since $\frac{d}{du} \ln f_U \;=\; \ln 2\,(1 - 2^{2u})$, Lemma 1 gives $\frac{d}{du} \ln \lambda^\star = \frac{\ln 2}{3}(3 - 2^{2u})$, so $2^{2u^\star} = 3$ and

$$u^\star = \tfrac{1}{2} \log_2 3 = +0.7925 \text{ octaves}. \tag{13}$$

Adaptive quadrature of f_U on $u \in [-60, 8]$ reproduces $u^\star = +0.79248$, $\mathbb{E}[U] = -0.91637$ and $\sigma_U = 1.60243$, matching the closed forms to seven decimals. Table 2 reports $\lambda^\star$ relative to its peak.

Level-placement quantiles ($5/25/50/75/95\%$) of $\lambda^\star$ lie at -3.20, -0.86, $+0.22$, $+0.99$, $+1.76$ octaves, a 90% placement span of 4.97 octaves, against a 90% *mass* span of $|W|$ of the same 4.97 octaves (-4.00 to $+0.97$). The placement span is the mass span offset upward by $u^\star$. That is the quantitative form of the two design consequences stated in §4.1: the bottom of the grid should

Table 2: Optimal log-domain point density relative to its peak, Gaussian source. Positions are octaves relative to σ_W. The mode of f_U is at $u{=}0$ and the peak of $\lambda^\star$ at $u^\star{=}{+}0.79$.

u	-3	-2	-1	$+0$	$+0.79$	$+1$	$+2$
$\lambda^\star(u)/\lambda^\star(u^\star)$	0.119	0.236	0.457	0.806	1.000	0.977	0.265

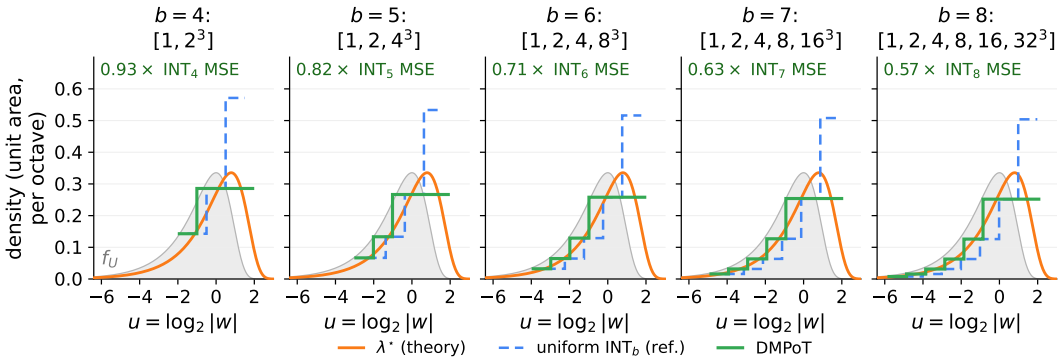

Figure 4: Per-width density matching, $b{=}4$–8: the log-magnitude density f_U of Gaussian weights (gray), the optimal point density $\lambda^\star(u) \propto [f_U\, 2^{2u}]^{1/3}$ (3) (orange), and the per-octave level histograms, at each grid's own MSE-optimal scale, of uniform INT$_b$ (blue, dashed) and the exact grid of the searched DMPoT schedule (green). The number in each panel is that grid's MSE relative to INT$_b$ (Table 4).

be sparse, and the grid should extend further above σ_W than mass-matching suggests. The same asymmetry sets how far $\lambda^\star$ travels. The optimal density varies by $4.36\times$ across a 4-bit support and $16.8\times$ across a 6-bit one, which is why no single step ratio can comply.

B.3 The multi-piece family and the nested chain

This appendix supports §4.1: the search over Eq. (1), the nesting of its per-bit-width optima, what it costs to serve a lower width from the stored eight-bit tensor, and what the restriction to dyadic densities costs. Every proxy number is regenerated by `analysis/multipiece.py`, which scores grids with the same evaluator as Tables 3 and 4 (Gaussian source, arithmetic-midpoint decisions, per-grid optimal scale). The end-to-end cells of Table 6 are model runs, read with the same extraction as Table 14. Figure 4 draws the selected schedule against $\lambda^\star$ and uniform INT$_b$ at every width. Both follow $\lambda^\star$'s doubling lower flank, a linear grid having log-density $\propto 2^u$, and they part company at the peak. The schedule caps its density at 2^{b-3} and holds through the roll-off, whereas the uniform INT grid keeps doubling and spends half its levels in the top octave, where $\lambda^\star$ is already falling. The mismatch, and the schedule's margin of $0.93 \to 0.57\times$ INT$_b$ proxy MSE, grows with b (Table 4).

Search procedure. A schedule is a non-decreasing list of octave densities $k_i \in \{1, 2, 4, \ldots, 2^{b-2}\}$. The grid takes the first $N = 2^{b-1} - 1$ exponents $i + j/k_i$, so the final piece may be partial. Schedules are deduplicated by level set, leaving $3/19/124/608/2637/10867$ distinct grids for $b = 3, \ldots, 8$. Searching this space without constraint yields the chain of Table 4. Imposing instead that each optimum *contain* its predecessor as a level set returns the same winner at every bit-width, so the nesting is free. The unconstrained optima already form a chain (nesting cost 0.00% at all five steps). The chain's doubling prefix is the closed-form lower-flank law of the derivation ($f_U \propto 2^u$ on the lower flank implies $\lambda^\star \propto 2^u$ there, Appendix B.2), not an empirical accident. The restriction to dyadic densities is what buys the bit-slice decode, the one-adder alphabet up to $k{=}4$ (hence $b{\le}5$ on the chain), and nesting by construction.

Table 4 sets out the resulting chain. Its two error rows are the same fact in two units. Along the chain the MSE ratio to INT$_b$ widens in the chain's favor, while the absolute error removed, read against the fixed 4-bit yardstick, vanishes. It is the absolute quantity that a downstream loss sees (tested end to end by the sweep of §6.3). Measured against that fixed 4-bit yardstick, the error the chain removes collapses by $24\times$ over the same range, a vanishing absolute return on added width.

Nesting. The per-width optima nest as level sets, $[1^3] \subset [1, 2^3] \subset [1, 2, 4^3] \subset [1, 2, 4, 8^3] \subset [1, 2, 4, 8, 16^3] \subset [1, 2, 4, 8, 16, 32^3]$, with the $b{=}3$ root the base-2 PoT grid. The subset test is

Table 3: Four-bit grids under the Gaussian proxy (N=7 non-zero magnitudes), each at its own MSE-optimal scale. MSE is relative to uniform INT4 at the same level budget. *Span* is the ratio of the largest to the smallest non-zero magnitude in octaves, and *zero* is the fraction of weights mapped to zero. The fixed-ratio control and the exact DMPoT-4 grid it is matched to are in *italics* (span within 0.01 octaves and zero fraction within 0.002, §4.1).

Grid (magnitudes)	MSE / INT4	span (oct)	zero
PoT, k=1 $\{1, 2, 4, \ldots, 64\}$	2.928	6.00	0.026
APoT n=2 $\{1, 2, 3, 4, 6, 8, 10\}$	0.939	3.32	0.105
FPoT exact, base $\sqrt{2}$	1.007	3.00	0.135
FPoT deployed $\{1, 1.5, 2, 3, 4, 6, 8\}$	1.055	3.00	0.132
fixed-ratio base 1.5 *(control)*	1.142	3.51	0.101
DMPoT-4 exact $[1, 2^3]$	0.929	3.50	0.099
DMPoT-4 deployed $\{1, 2, 3, 4, 6, 8, 12\}$	**0.984**	3.58	0.097

Table 4: The chain and its bit-width laws. Gaussian proxy, with every grid at its own MSE-optimal scale against uniform INT$_b$ at the same level budget. The MSE rows score the exact grids (deployed images in Tables 3 and 5). *Error removed* is MSE(INT$_b$) minus the schedule's MSE, in units of the INT$_4$ MSE. The adder row is the smallest budget whose nearest projection stays injective at the peak density (Table 7), a minimum up to one adder and an upper bound beyond (Appendix B.7). The width row is the narrowest odd core among schedules whose deployed grid reaches the INT$_b$ MSE (Appendix B.5). Search procedure and proofs are in Appendix B.

b	3	4	5	6	7	8
schedule (chain-b)	$[1^3]$	$[1, 2^3]$	$[1, 2, 4^3]$	$[1, 2, 4, 8^3]$	$[1, 2, 4, 8, 16^3]$	$[1, 2, 4, 8, 16, 32^3]$
MSE / INT$_b$	1.015	**0.929**	0.815	0.715	0.633	0.569
error removed (/INT$_4$)	-0.054	**0.071**	0.053	0.024	0.009	0.003
peak density 2^{b-3}	1	2	4	8	16	32
minimum adders	0	**1**	**1**	2	2	3
iso-MSE odd-core width w=$b-2$ (bits)	–	2	3	4	5	6

decided on exact rational exponents, at 0.00% proxy cost and with no training. This is not to be expected a priori. Optimal scalar quantizers are in general not embeddable across rates, and embedded designs pay a measured loss (Brunk & Farvardin, 1996), while refinability at the rate–distortion level for a Gaussian source is a separate, classical statement (Equitz & Cover, 1991). Concurrent multi-precision work instead *learns* the nesting, by a Matryoshka-style objective over additive vector codebooks (Babaoglu et al., 2026) or by MSB-nesting on uniform or binary-coded grids (Nair et al., 2025; Park et al., 2024; 2026). Structural nesting alone is not distinctive (uniform INT nests by LSB truncation, and its truncations are the per-width optima of the uniform family). The content here is that a *density-matched, non-uniform* family nests. The mechanism is the elementary inclusion $\{2^{j/k}\} \subset \{2^{j/mk}\}$ whenever the finer density is an integer multiple m of the coarser one, which the dyadic family meets with m=2 at every added bit, while densities that do not divide break it ($\{2^{j/3}\} \not\subset \{2^{j/5}\}$). Because the in-octave coefficient is linear, the *deployed* grids nest as well ($\{1, 1.5\} \subset \{1, 1.25, 1.5, 1.75\} \subset \cdots$). A b=8 code contains each lower width as an octave-aligned window (dropping coefficient least-significant bits thins the in-octave density, the window selects the range, and the per-channel scale absorbs its shift), with $G_4 \cdot 2^t \subset G_8$ for $t = 0, \ldots, 4$ and the b=4 window the deployed DMPoT-4 set up to a power of two. What nests is the level set, not a model that can be *served* at several widths, which the learned methods buy and this format does not obtain for free (§B.4, Table 6). Figure 5 draws the deployed level sets on one log-magnitude axis, where the sparse bottom octave and the nesting are visible directly. Table 5 gives their odd cores and spacing regimes.

The shared scale, on the proxy. A single stored scale is not free on the Gaussian proxy, since each b has its own MSE-optimal scale. The stored scale's proxy cost $\min_t \mathrm{MSE}(G_4 \cdot 2^t s_8^\star)/\mathrm{MSE}(G_4 s_4^\star) - 1$ is minimized at t=4, at 1.011 against $0.984\times$ INT$_4$, i.e. $+2.7\%$. The minimization over t is not optional. Reusing $s_8^\star$ without re-centering the window collapses the grid by roughly two orders of magnitude. On the scale of proxy movements this paper has priced end to end (the one-adder projection moves the grid from 0.929 to $0.984\times$ INT$_4$ and costs 0.024 PPL, Appendix H.3), $+2.7\%$ predicts a hundredth of a perplexity point. Whether that is the quantity that matters is a separate question, and §B.4 measures it.

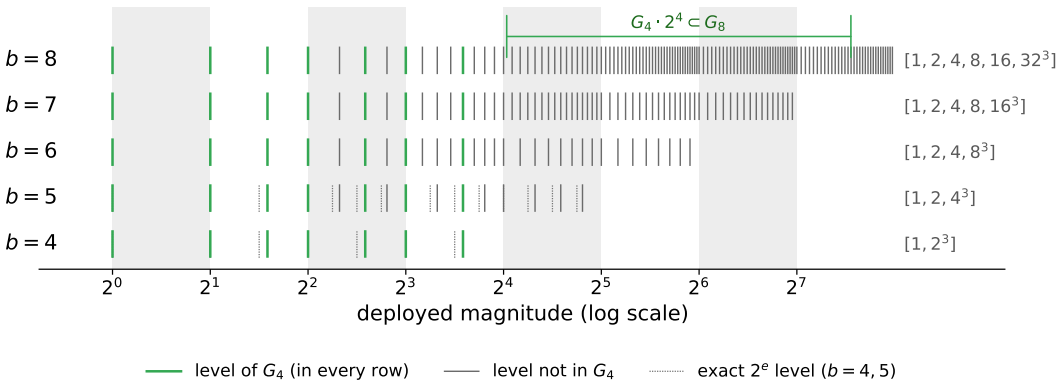

Figure 5: The chain of deployed grids, b=4 to 8, on one log-magnitude axis. Ticks are the levels of the deployed grid G_b, green where a level is already in G_4. Alternate octaves are shaded. Dotted ticks (b=4, 5) are the exact geometric levels 2^e beside their one-adder images, and the bracket marks the octave-aligned window $G_4 \cdot 2^4$ inside G_8, the window served in §B.4.

Cost of the dyadic restriction. Allowing arbitrary integer densities improves the proxy by 4.7, 8.3, 7.2, 7.7 and 9.3% at b=4, . . . , 8 (exhaustive over monotone schedules for $b \leq 5$, then coordinate descent). The margin grows from 4.7% at b=4 to 7–9% from b=5 on, but on the fixed 4-bit yardstick of Table 4 it shrinks from 0.044 of the INT_4 MSE at b=4 to 0.004 at b=6 and 0.0004 at b=8. From b=6 it is at most a fifth of the error the chain itself removes against INT_b, and the sweep of §6.3 already places the chain within 0.01 PPL of INT_b there. At the deployable widths the restriction costs real proxy MSE and is justified by the decoding, the alphabet and the nesting rather than by being cheap. One caveat is worth recording, because it is easy to measure the opposite. The dyadic optima happen to be monotone, so constraining the search to monotone schedules is free for them but not for arbitrary densities. Imposing it on both families reports the b=8 gap as 0.35% instead of 9.3%. The price of leaving the dyadic grid is permanent. The in-octave index stops being a bit-slice, the odd-core set grows from $\{1, 3\}$ to $\{1, 3, 5\}$ or larger, and nesting across widths no longer follows by construction.

Table 5: The chain's deployed grids, one row per bit-width. Each grid is the linear-mantissa image of its schedule (§3, notation in Appendix A.3), a set of integers drawn in Fig. 5. The *cores* column is the odd-core alphabet $\mathcal{Q}$, the odd parts of the levels. The *shift classes* column gives the largest level of each spacing regime, with steps of 1, 2 and 4 (§B.5). MSE is computed for the deployed set at its own optimal scale on $W \sim \mathcal{N}(0, 1)$.

b	schedule	levels	cores $\mathcal{Q}$	MSE/INT_b	shift classes
4	$[1, 2^3]$	7	$\{1, 3\}$	**0.984**	$\leq 4, \leq 8, \leq 12$
5	$[1, 2, 4^3]$	15	$\{1, 3, 5, 7\}$	0.882	$\leq 8, \leq 16, \leq 28$
6	$[1, 2, 4, 8^3]$	31	$\{1, 3, \ldots, 15\}$	0.775	$\leq 16, \leq 32, \leq 60$
7	$[1, 2, 4, 8, 16^3]$	63	$\{1, 3, \ldots, 31\}$	0.685	$\leq 32, \leq 64, \leq 124$
8	$[1, 2, 4, 8, 16, 32^3]$	127	$\{1, 3, \ldots, 63\}$	0.613	$\leq 64, \leq 128, \leq 252$

B.4 What nesting does not buy: serving a lower width

Serving a lower width, measured. We store one b=8 tensor (the checkpoint behind the DMPoT W8A8KV8 cell of Table 14) and serve b=4 from it through the t=4 window on Llama-3-8B at A8KV8, over the same three calibration seeds and with no retraining (Table 6). A control that reserves the stored width without inducing anything reproduces the published perplexity exactly (6.160 against 6.1605), which fixes the resume protocol. The induced b=4 grid then reads $7.719_{\pm.132}$ PPL and $65.1_{\pm.7}$ 0-shot[9], against $6.672_{\pm.013}$ and $67.3_{\pm.3}$ for the chain's independently calibrated b=4 member. The penalty is +1.047 PPL and −2.2 points, and the two axes move together. Grid geometry is not the cause. At t=4 the window is the deployed DMPoT-4 grid times exactly $192/252 = 0.7619$ at every one of its seven levels, so the two runs quantize onto the same level shape and differ only in the scale anchor and in the width at which preprocessing and GPTQ were fit. Nor is it an artifact of where the truncation is applied. Projecting onto the window inside the GPTQ sweep instead of once after it gives $7.726_{\pm.128}$ and $65.0_{\pm.7}$.

Table 6: Serving b=4 from one stored b=8 tensor (Llama-3-8B, A8KV8, mean$_{\pm\text{std}}$ over three calibration seeds, no retraining). *Calibrated at b* is the width at which preprocessing and GPTQ were fit. Rows are described in §B.4. PPL: WikiText-2 (↓). 0-shot[9]: nine-task mean (↑, %).

	stored b	served b	calibrated at b	PPL↓	0-shot[9]↑
Independent b=8 (stored model)	8	8	8	$6.159_{\pm.001}$	$68.1_{\pm.1}$
control, no induction (seed 42)	8	8	8	6.160	68.2
Independent b=4 (chain member)	4	4	4	$6.672_{\pm.013}$	$67.3_{\pm.3}$
Induced b=4, window t=4	8	4	8	$\mathbf{7.719}_{\pm.132}$	$\mathbf{65.1}_{\pm.7}$
diagnostic: window applied inline	8	4	8	$7.726_{\pm.128}$	$65.0_{\pm.7}$
diagnostic: scale renormalized	–	4	8	$7.879_{\pm.049}$	$64.7_{\pm.2}$

Where the penalty sits. Re-anchoring the same window geometry at the stored scale's full range instead of 0.762 of it (no longer a subset of the stored levels, so a diagnostic only) costs *more*, $7.879_{\pm.049}$ and $64.7_{\pm.2}$. Sharing the stored scale is thus worth -0.160 PPL, and the whole $+1.207$ PPL sits with the rotation, smoothing scales and GPTQ compensation having been fit at b=8 rather than b=4, a calibration-width effect the proxy does not see (it charges $+2.7\%$ for the shared scale). We therefore keep two claims apart. The nesting is structural and free ($G_4 \subset \cdots \subset G_8$ holds exactly for the exact and the deployed grids alike, at 0.00% proxy cost), but a stored tensor is not by itself a served model. At b=4 it costs about 1 PPL and two points of zero-shot accuracy unless preprocessing and weight calibration are redone at the serving width, which is what every member of the sweep in §6.3 does. Progressive precision from this format is therefore a re-calibration problem rather than a format problem. The methods cited above buy it by training for it. We do not train for it here, and we make no serving claim beyond Table 6.

B.5 ISO-MSE WIDTH MINIMALITY

This subsection supports the odd-core-width row of Table 4 and the width-minimality law of §4.2. The search behind that row inverts the objective of its MSE rows. It keeps every schedule whose deployed grid satisfies MSE $\leq$ MSE(INT$_b$) and sorts by datapath cost rather than by error. Feasible sets are large ($17/116/631/2880$ schedules at b=$5, \ldots, 8$), yet the minimizer is the chain member itself at every width, and it is minimal in both cost terms at once. No feasible schedule matches its odd-core width with fewer shift classes (spacing regimes, of which the chain member has three: consecutive integers up to 2^{b-2}, every second integer up to 2^{b-1} and every fourth up to $2^b - 4$). The one-bit-narrower core is not merely worse but far outside the constraint (1.98–$2.88\times$ INT$_b$), so $w = b - 2$ is a hard floor for this family rather than the edge of a shallow optimum. At b=4 the statement holds with w=2 (the deployed grid of §4.2 at 0.984), with w=1 being the PoT family. In hardware terms, the chain reaches INT$_b$ proxy MSE with a multiplier two bits narrower than INT$_b$'s, and within this family no narrower one suffices. To our knowledge no prior numeric format states a quantified, exhaustively verified minimal multiplier width at iso-INT MSE. Regenerated by `cd analysis && python multipiece.py ge`.

B.6 RELATION TO 4-BIT FLOATING POINT

The deployed DMPoT-4 magnitudes coincide, up to a power of two, with the element lattice of block-scaled 4-bit floating-point formats (Open Compute Project, 2023; NVIDIA et al., 2026). We read this as convergent design rather than derivation from it. At four bits, three independently constructed log-domain grids (LogART, FPoT, and DMPoT) share the same two-element odd core and hence the same folded datapath (§5). Our analysis explains the convergence. A one-addition budget forces any such grid onto a linear-mantissa lattice (§4.2). The shared lattice does not by itself separate the two designs. An E2M1 (2-bit exponent, 1-bit mantissa) $\times$E2M1 MAC also reduces to integer logic: map the mantissas $\{1, 1.5\}$ to $\{2, 3\}$, form the three products $\{4, 6, 9\}$ by lookup, add the exponents as integers, and accumulate in fixed point. What separates them is the second operand. A symmetric E2M1$\times$E2M1 array carries the exponent sum t_w+t_a into every processing element of the array. The weight exponent varies along each lane and the activation exponent across lanes, so the sum can be formed neither once per lane nor folded into the static weight. The alternative, expanding each activation to fixed point at the array edge, is exactly the asymmetric log$\times$linear datapath this paper deploys, at a wider lane operand. With uniform INT activations the expansion is the identity, the one

Table 7: Realizable in-octave resolution per adder budget. $\mathcal{R}_n$ as in §B.7, with exponents truncated at 2^{-8}. The last column is the largest k for which nearest projection of the targets $2^{j/k}$ onto $\mathcal{R}_n$ is injective, scanned up to k=128.

adder budget n	$\lvert\mathcal{R}_n \cap [1,2)\rvert$	largest collision-free k
1	16	**5**
2	88	20
3	208	66

addition per lane is the shared product $3x_i$, and the processing element keeps only a 2:1 multiplexer, a shifter, and the accumulator (§5). The asymmetry also sites the approximation error. The 6.1% $\sqrt{2}\to1.5$ substitution (Proposition 1) is confined to the static weights, where the scale search and GPTQ absorb it during calibration (§3). Activations quantized onto the same log lattice would carry it per token, with no calibration step to compensate. The area ratios of §5 accordingly compare integer-activation datapaths, and we synthesize no E2M1×E2M1 engine.

B.7 Realizable ratio sets

This appendix expands §4.2. For n=1 the realizable set is closed in form,

$$\mathcal{R}_1 \;=\; \{1\} \;\cup\; \{1+2^{-j}\} \;\cup\; \{2-2^{-j}\}, \qquad j \geq 1, \tag{14}$$

accumulating at the octave ends with exactly one element, 1.5, in the middle third. $\mathcal{R}_n$ is enumerated as the set of signed sums of n+1 powers of two falling in $[1,2)$, with exponents truncated at 2^{-8}, and the scan over k stops at k=128 (Table 7). The three sets have 16, 88 and 208 elements in $[1,2)$ for $n = 1, 2, 3$. The statement used in §4.2 is:

Proposition 1 (Realizable resolution)**.** *Mapping each target $2^{j/k}$ to its nearest element of $\mathcal{R}_1$ is injective if and only if $k \leq 5$. At k=6 the targets $2^{3/6} = 1.414$ and $2^{4/6} = 1.587$ both round to 1.5 and the grid loses a level. For $k \leq 5$ the worst-case ratio error is 6.1% ($k \in \{2,4\}$), 5.5% (k=3) or 5.3% (k=5).*

Nearest projection extends the same limit to $k \leq 20$ and $k \leq 66$ for n=2 and n=3. Those two figures are properties of the construction, not of the budget. Choosing the codebook *jointly* over $\mathcal{R}_n$, instead of projecting each geometric target independently, can place more levels in an octave than injective nearest projection allows, so the $n \geq 2$ budgets in Table 4 should be read as what this construction requires rather than as lower bounds. Only the n=1 row is load-bearing for the $b \leq 5$ bound of §4.2, and it is exact. Eq. (14) is closed-form and its $k \leq 5$ limit is Proposition 1.

The complete dyadic grid $G_M = \{1 + m/2^M : 0 \leq m < 2^M\}$ used in §4.2 is contained in $\mathcal{R}_n$ exactly up to $M = 2n$ ($M = 2, 4, 6$ for $n = 1, 2, 3$), verified by exhaustive enumeration. The inclusion is strict. The set $\mathcal{R}_n$ also holds ratios off G_{2n}, and Table 7 counts all of them at resolution 2^{-8}. For $n = 1$ the boundary case is explicit. The ratios $1.25 = 1 + 2^{-2}$, $1.5 = 1 + 2^{-1}$ and $1.75 = 2 - 2^{-2}$ are all realizable, whereas 1.375 and 1.625 admit no representation $2^a \pm 2^b$ and therefore require a second adder.

B.8 Robustness across sources

The Gaussian enters the design only through f_U, so the derivation of §B.1 runs unchanged for any source. Where the peak of $\lambda^\star$ lands, and whether the b=4 ranking survives, are both governed by the tails.

The peak location is an invariant of a large family. For the symmetric Pearson II/VII family $f(w) \propto (1 + cw^2)^{-m}$, which covers Student-$t$ for every $\nu > 2$ (the Gaussian as $m \to \infty$) and the symmetric Beta densities for every $\beta > 1$, the stationarity condition $3f(w) + wf'(w) = 0$ puts the peak at $w^\star = \sqrt{3}\,\sigma_W$, $u^\star = \frac{1}{2}\log_2 3 = +0.7925$ octaves, independent of c and m (the uniform source attains the same maximum as a corner at the support edge). The invariance is a property of this family, not of all sources. Laplace puts the peak at $+1.085$ octaves and a generalized Gaussian with exponent 4 at $+0.679$, so the measured Llama-3 peak at $+0.795$ (Appendix C) is an empirical fact about preprocessed weights, not a tautology of the analysis. The peak is also flat, with $\lambda^\star$ staying within 1% of its maximum over a 0.29-octave window, so the design is insensitive to misestimating σ_W or the source family.

Table 8: The b=4 grid ranking across sources, bounded to heavy-tailed. Entries are MSE relative to uniform INT4 *on the same source*. PoT, FPoT, APoT and DMPoT are the deployed grids of Table 3, and "family best" is the best exact grid of the dyadic schedule family of §4.1. Lloyd–Max is the unconstrained 15-level optimum. $u^\star$ is the peak of $\lambda^\star$ in octaves above σ_W (†: attained at the support edge, with no interior stationary point).

source	$u^\star$	PoT	FPoT	APoT	DMPoT	family best	Lloyd–Max
Uniform	$+0.79^\dagger$	5.68	1.76	1.34	1.82	1.05	1.00
sym-Beta β=2	+0.79	4.78	1.50	1.20	1.58	1.05	0.97
sym-Beta β=4	+0.79	4.01	1.29	1.08	1.36	1.05	0.92
sym-Beta β=63.5	+0.79	3.00	1.07	0.95	1.00	0.94	0.84
Gaussian	+0.79	2.93	1.06	0.94	0.98	0.93	0.83
Llama-3-8B (measured)	+0.79	2.89	1.05	0.93	0.98	0.92	0.83
Laplace	+1.08	1.40	0.84	0.74	0.67	0.67	0.63
Student-t, ν=4	+0.79	0.72	0.82	0.75	0.66	0.66	0.52

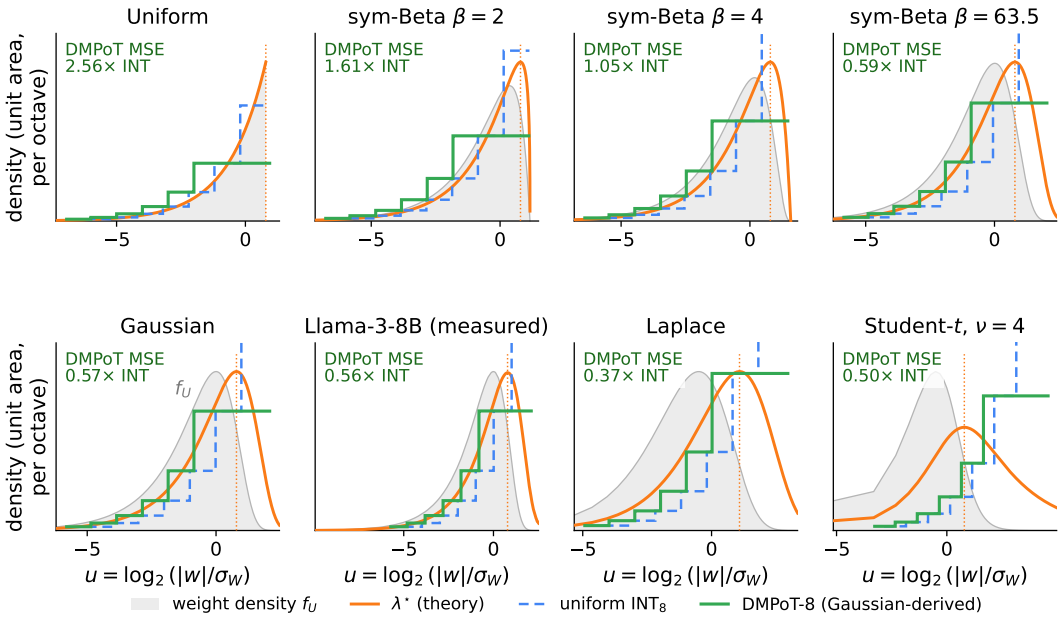

Figure 6: Point-density matching at b=8 on the sources of Table 8. Each panel shows the source's log-magnitude density f_U (gray) and its own optimal point density $\lambda^\star$ (orange, peak dotted), with the per-octave level histograms of uniform INT_8 (blue, dashed) and the exact grid of the Gaussian-derived DMPoT schedule $[1, 2, 4, 8, 16, 32^3]$ (green), each at its own MSE-optimal scale on that source. The number in each panel is that schedule's MSE relative to INT_8 on the same source. Neither grid is re-derived per source.

The ranking is governed by the tails. Table 8 scores the deployed b=4 grids on sources from bounded to heavy-tailed (every number regenerated by `analysis/source_robustness.py`, same evaluator as Table 3). Heavier tails widen the log-grid advantage. On Laplace and Student-t (ν=4) sources DMPoT-4 reaches $0.67\times$ and $0.66\times$ INT4. Strongly bounded sources reverse it. On the uniform source, uniform INT4 *is* the 15-level Lloyd–Max optimum (strict convexity of $\sum_i \Delta_i^3$ at fixed span), every fixed grid in the table scores above $1\times$, and the best member of the dyadic family manages only $1.05\times$ (a non-dyadic density, k=3, reaches $0.94\times$ on sym-Beta β=4, still within a few percent of INT4). The crossover sits where it should. The coordinate marginal of a rotated unit-norm vector in $\mathbb{R}^d$ is sym-Beta with $\beta = (d-1)/2$, and at d=128 (β=63.5) the bounded source is already Gaussian-like (DMPoT $1.00\times$). The grid of this paper is a tool for wide-dynamic-range sources. On strongly bounded ones the same machinery returns the uniform grid. (The paper's use of uniform INT for activations and the KV cache rests on the integer-datapath argument of §5, not on a distributional one. We have not measured activation or KV densities here.)

Figure 6 draws f_U and $\lambda^\star$ on every source of the table against the same two fixed b=8 grids. The Gaussian-derived DMPoT schedule tracks $\lambda^\star$ closely wherever its peak coincides with the source's

own, and the two visibly part company where it does not. Read against uniform INT8 on the same source, the schedule's advantage widens from a tie on sym-Beta β=4 ($1.05\times$) to $0.37\times$ on Laplace, and reverses to $2.56\times$ on the strongly bounded Uniform source, the same crossover Table 8 reports at b=4.

B.9 Relation to the unconstrained (Lloyd–Max) optimum

A codebook quantizer free to place all 15 signed levels satisfies the two Lloyd–Max conditions, nearest-level boundaries and centroid levels. The deployed quantizer meets the first exactly (arithmetic midpoints) and approximates the second with two scalars, the per-channel scale and the grid-alignment offset of §3, instead of seven free magnitudes. The missing freedom costs little. Against the Gaussian 15-level Lloyd–Max optimum ($0.01074\,\sigma_W^2$, `analysis/source_robustness.py`), uniform INT4, APoT, deployed DMPoT-4 and deployed FPoT read 1.200, 1.127, 1.181 and $1.267\times$ and PoT $3.515\times$, so level placement at four bits is worth at most 20% in proxy MSE, one reason these figures rank grids rather than predict accuracy (Appendix B.10) and the span-matched grids' perplexities cluster tightly (§6.2).

The same lens locates the equal-mass rule that §2.3 positions against. NF4's Gaussian-quantile placement (Dettmers et al., 2023) is the $\lambda^\star \propto f_U$ prescription, whereas the optimal density of Eq. (3) is $\propto \left(f_U\, 2^{2u}\right)^{1/3}$. The extra factor $2^{2u/3}$ weights error by magnitude, so equal-mass placement under-resolves the upper flank, where large weights make large errors.

The unconstrained optimum also explains the one place a competitor stays ahead. Its magnitudes, normalized to the smallest, are $\{1.00,\ 2.03,\ 3.11,\ 4.29,\ 5.65,\ 7.33,\ 9.79\}$, near-consecutive integers at the bottom that widen toward the top (precisely the shape of the deployed grids), with a top level of 9.79. APoT's top level (10) sits almost on it. The two-coefficient lattice must choose 8 or 12, and the density argument picks 12. That single level, the only one where the two sets differ, is APoT's entire MSE edge ($1.127\times$ against $1.181\times$ Lloyd), a margin the fourteen cells of Table 1 do not resolve on perplexity (§6.2).

Within its own hardware class, however, the deployed grid is not merely the image of the best schedule but globally optimal. Enumerating all 5999 scale-classes of 7-element subsets of the lattice $\{2^t,\ 1.5\cdot 2^t\}$ (spans up to seven octaves, exact rational deduplication, `analysis/source_robustness.py`), $\{1,2,3,4,6,8,12\}$ ranks first at $0.984\times$ INT4. Exactly one other class beats INT4: $\{1.5,3,4,6,8,12,16\}$ at $0.999\times$. The runner-up shares the two-coefficient in-octave alphabet, so it folds identically. It needs five exponent rungs against four. The tiebreak between the only two INT4-beating members of the class is therefore encoding width, a hardware quantity.

Remark (the deployed grid is not the proxy-optimal one-adder grid). Searching one-adder deployed grids without the schedule structure returns $\{1,2,3,4,5,6,8\}$, which improves the proxy over $\{1,2,3,4,6,8,12\}$ but needs the odd cores $\{1,3,5\}$. That is the trade the hardware section measures. It is $|\mathcal{Q}|$=2 that makes the fold profitable (§5), and the proxy difference sits inside the regime where §B.10(i) says the proxy is weakest. The deployed grid is selected jointly with the datapath, not by proxy MSE alone. The projection of §4.2 is therefore forced only *within* the schedule family, and it is the datapath, not the proxy, that selects the family.

B.10 Scope of the argument

We state the following limitations explicitly rather than in a footnote.

(i) High-resolution regime. Eq. (3) is an $N \to \infty$ asymptotic and is violated at $b = 3$–4. We use the derivation as a *design principle* that predicts a sparse bottom octave and its approximate magnitude. The exact schedule comes from exhaustive search. We do not claim the derivation outputs the design.

(ii) Gaussian proxy. All MSE figures assume a Gaussian source. Rotation-based preprocessing drives weight distributions toward Gaussianity, which is what makes the proxy usable. Appendix C reports the corresponding measurement on real preprocessed weights. Analyses that instead assume a *log-normal* source (for instance Chmiel et al. (2021) for neural gradients) reach different exponent/significand layouts, so the distinction matters and the measurement is what settles it for our

setting. Appendix B.8 quantifies the dependence directly. The peak location is invariant across the symmetric Pearson II/VII family, while the b=4 ranking shifts with tail weight.

(iii) Unweighted MSE. The objective ignores Hessian weighting and the error compensation performed during weight calibration. We therefore use the proxy to *rank* grids within a narrow accuracy band rather than as an absolute predictor of perplexity. Within the near-INT band of Table 3, the measured perplexities order the same way as the proxy figures, which is what the ranking use rests on. The induced-serving measurement of §B.4 shows how far apart the two can be. The proxy prices the shared stored scale at $+2.7\%$ MSE, whereas end to end that constraint is worth -0.160 PPL. The $+1.207$ PPL actually paid comes from fitting the calibration at the stored width, a quantity this objective does not represent at all.

(iv) Fixed-transform analysis. The derivation ranks grids under a *fixed* transform. When the preprocessing transform is co-optimized with the grid, f_U itself adapts and measured gaps narrow relative to the prediction. The analysis therefore predicts grid *order*, not absolute accuracy after co-optimization. It is equally silent on a transform fit at one width and deployed at another, which is one half of what serving an induced grid does. Table 6 prices that mismatch only in combination with the weight calibration it is bundled with.

(v) Representability is necessary, not sufficient. Proposition 1 says a k-level octave *can* be built within an adder budget. It does not say the resulting accuracy is acceptable. The algebra gives hardware feasibility and the schedule search gives accuracy. Only their intersection is a design. In particular b=5 is feasible with one adder and is predicted by Table 4 to beat INT5 on MSE. End to end, the measured W5A8 accuracy is within 0.01 PPL of INT5 (§6.3), consistent with the proxy margin being too small to resolve.

(vi) No area figures. This appendix contains none, by construction. The $b \leq 5$ bound of §4.2 is arithmetic. Where the paper claims area it comes from tile-level synthesis (§5), because per-module PE area does not predict array area. The shared scaled operand is a free input at a module boundary but a derived signal inside a flattened tile (Appendix E).

C THE GAUSSIAN PROXY, MEASURED

The schedule search of §4.1 optimizes against a Gaussian source. That assumption is the one input to the derivation that is not itself derived, and it is not innocuous. Analyses that assume a *log-normal* source instead reach different exponent/significand layouts (Chmiel et al., 2021), so a grid designed under the wrong source would be the wrong grid. This appendix therefore measures f_U on real preprocessed weights rather than assuming it.

Protocol. We take Llama-3-8B, apply the same preprocessing used throughout the paper (layernorm fusion, learned rotation, smoothing), and histogram $u = \log_2 |w|$ over *all* weights of *all* seven linear projections in all 32 blocks (6.98×10^9 values), after normalizing each output channel by its own standard deviation, so that u is measured in octaves relative to σ_W exactly as in Appendix B. From the measured histogram we recompute $\lambda^\star \propto (f_U 2^{2u})^{1/3}$ and its peak, and we evaluate the four deployed 4-bit grids on the measured distribution at each grid's own MSE-optimal scale. As a control, the same histogram is taken of the raw checkpoint weights, before any preprocessing.

Where the proxy is load-bearing. The quantity the design actually consumes is the location of the peak of $\lambda^\star$, because that is what says how far above σ_W the grid should reach and how sparse its bottom should be. Measured, it lands at $u^\star = +0.795$ octaves. The Gaussian closed form is $\frac{1}{2}\log_2 3 = +0.7925$. The first two moments agree as closely, with $\mathbb{E}[U] = -0.918$ measured against -0.916 predicted, and $\sigma_U = 1.600$ against 1.602. The agreement is not a coincidence of scale. Rotation is an orthogonal mixing of many channels, so the central limit theorem is doing the work. What it buys is visible in Figure 7a. Per-channel σ-normalization makes even the *raw* bulk near-Gaussian ($\mathbb{E}[U] = -0.94$, $\sigma_U = 1.62$), but the raw outlier tail reaches $62\,\sigma_W$, with $P[u > 3] \approx 10^{-5}$ against 10^{-10} after rotation, which confines it to $\approx 8\,\sigma_W$, and the tail is precisely what the 2^{2u} weighting amplifies. The proxy is therefore safe only on preprocessed weights, and it is the tail, not the bulk, that preprocessing fixes.

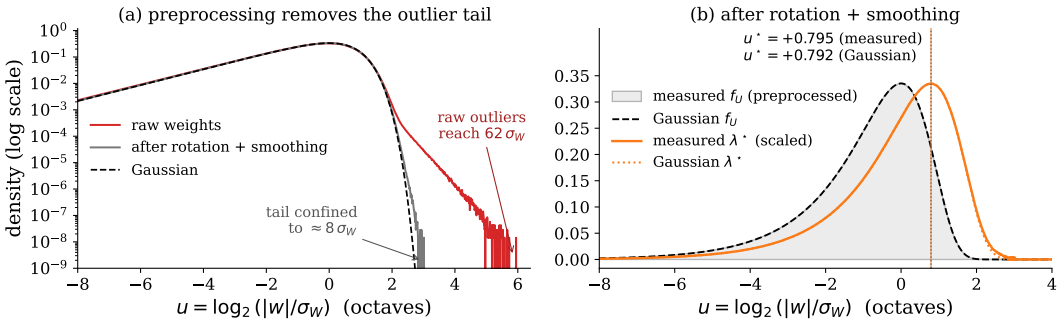

Figure 7: The Gaussian proxy measured on 6.98×10^9 Llama-3-8B weights, before and after preprocessing. **(a)** log-scale density of $u = \log_2(|w|/\sigma_W)$, raw against preprocessed, with the Gaussian closed form dashed. **(b)** after preprocessing, the measured f_U and the induced $\lambda^\star \propto (f_U 2^{2u})^{1/3}$ against their Gaussian closed forms, with vertical lines at the measured and Gaussian peaks $u^\star$.

Where it does not hold. Figure 7b shows the measured density is not exactly Gaussian. It carries a heavier left tail (a small excess of near-zero weights) and a slightly lighter right tail. Both deviations are in the region the $2^{2u/3}$ factor discounts most strongly. The lower flank contributes least to absolute error, and the upper tail carries little mass, so $\lambda^\star$, which depends on $f_U^{1/3}$, is insensitive to them. This is the practical content of the derivation's use of a cube root. The design is a function of the *shape* of the log-density near its upper flank, not of its tails.

Grid ranking on measured weights. Evaluating the deployed grids on the measured distribution reproduces the ranking the analysis predicts and the experiments observe. Relative to uniform INT4 at the same level budget, the unweighted MSE is $2.89\times$ for PoT, $1.05\times$ for the FPoT ($J{=}0$) grid, $0.98\times$ for DMPoT-4 and $0.93\times$ for APoT. The corresponding zero fractions are 0.026, 0.133, 0.098 and 0.105. Two things are worth noting. First, DMPoT-4 improves on the $J{=}0$ member by the margin the Gaussian analysis predicts, and the direction and rough size of every gap survive the move from proxy to measurement. Second, APoT remains slightly ahead of DMPoT-4 on this metric. End to end the two tie. Co-designed on both sides, the mean gap over the fourteen model×configuration cells is 0.005 PPL, and it runs the other way (in DMPoT's favor, §6.2). No cell separates them by more than 0.08. So the proxy's small APoT edge is not contradicted by the experiments. It is simply too small to resolve, which is what the analysis says it should be. The additive levels buy a little accuracy. What they cannot buy is the smaller folded tile.

D Per-Projection Sensitivity Analysis

This appendix supports the weights-only scope of §3. It locates where the weight MACs are spent and measures how each linear projection tolerates the DMPoT grid.

D.1 Per-projection MAC breakdown

On Llama-3-8B (32 layers, one token) the FFN projections carry 80.8% of the $6{,}979$ M weight MACs (Gate, Up and Down at 26.9% each) and attention 19.2% (W_Q and W_O at 7.7% each, W_K and W_V at 1.9%).

D.2 Per-projection DMPoT sensitivity

Table 9 reports the per-projection DMPoT sweep on Llama-3-8B (W4A4KV4). Each row applies the weight-only DMPoT-4 grid to the indicated projection(s) while the rest use uniform INT, which isolates how much of the total weight-grid overhead each projection contributes and whether the contributions compose additively.

Every single-projection row lands at or below the All-INT baseline of 7.444, within 0.12 PPL of it, so no projection is individually intolerant of the grid. The projections W_O, Gate and Up are the most tolerant (7.329–7.336) and W_V and Down the least (7.392). The contributions do *not* compose

Table 9: Per-projection DMPoT-4 weight-only sweep, Llama-3-8B W4A4KV4 (400 steps, activations and KV cache uniform INT4). Each row applies DMPoT-4 to the listed projections and uniform INT4 to the rest. "All INT" is the uniform INT baseline. Single seed (42) for a like-for-like comparison. The bold row is the deployed all-projection configuration, whose three-seed mean is 7.417 (Table 1).

DMPoT Applied To	PPL↓	0-shot[9]↑ (%)
All INT (uniform)	7.444	64.98
W_Q only	7.370	64.81
W_K only	7.364	64.51
W_V only	7.392	64.87
W_O only	7.329	65.12
Gate only	7.329	65.38
Up only	7.336	64.91
Down only	7.392	64.86
Attention (W_Q+W_K+W_V+W_O)	7.406	64.37
FFN (Gate+Up+Down)	7.365	65.42
All 7 projections	**7.441**	**65.16**

additively, and not in the direction a sensitivity budget would suggest. The seven single-projection gains do not accumulate, and applying the grid to all seven returns to 7.441, essentially the All-INT baseline. Preprocessing is co-optimized against whichever projections are quantized, so a single quantized projection is partly compensated by the six that are not, which is why the per-projection rows are read here as tolerance evidence rather than as an error budget.

E Hardware Synthesis Details

This appendix documents the synthesis harness behind §5: the per-variant areas and M/N sweeps behind the anchor table (Table 10), the least-squares decomposition, the equivalence checking, and the per-model accuracy–cost trade-offs. The raw measurements are committed as one JSON per synthesized cell (`hw_synthesis/y068c/{tile,dmpotd}_W{4,5}A{4,8}.json` for the Yosys 0.68 sweep the paper reads, sibling directories for the other builds and libraries of Table 11), merged into `hw_synthesis_per_cell.csv`, from which every table is regenerated by `analysis/gen_hw_tables.py`.

E.1 Tile RTL and synthesis flow

A tile is parameterized by the weight width W, the activation width A, the reduction depth N (lanes) and the output-channel count M, and is emitted by `hw_synthesis/tile_gen.py` from the processing-element templates in `pe_modules.v`. It contains

1. the per-lane precompute, which for the folded datapath is one $3x_i = x_i + (x_i \ll 1)$ adder per lane, shared by all M output channels, and is absent in the unfolded engines;
2. $N \times M$ processing elements, each a barrel shifter plus (depending on the engine) a coefficient select, a per-element adder, or an integer multiplier;
3. an explicit 3:2 carry-save reduction tree per output channel (`hw_synthesis/csa_gen.py`), followed by a carry-propagate adder;
4. an accumulator of width $W{+}A{+}8$ ($W{+}A{+}10$ for the wide-range LogART-5 engines, whose integerized levels reach 256 and need the extra range at $N{=}64$).

The $b{=}4$ engine set has eleven members: `int` (a $W \times A$ signed multiplier, the accuracy reference), `pot` (a single barrel shifter), three realizations of each $\{1, 3\}$ grid (DMPoT-4 and FPoT), and three APoT realizations (`apot_base`, `apot_unf_oddcore`, `apot_fold`). The three realizations of each $\{1, 3\}$ grid are `dmpot4_paper`/`fpot_paper` (shift then per-element $1.5\times$ adder), `dmpot4_unf`/`fpot_unf_oddcore` (odd-core operand order, per-element adder) and `dmpot4_fold`/`fpot_fold` (the folded realization of §5). LogART-4 shares the FPoT

Table 10: MAC-tile area at the $(32, 32)$ operating point as a ratio to same-bit uniform INT (INT row in absolute GE). Yosys 0.68, sky130. Each cell is the cheapest implementation of its grid. DMPoT is the folded one-adder tile up to b=5 and the chain datapath from b=6 (Table 15). LogART and FPoT share one tile at every width. APoT defines no b=8 grid ('–').

Grid	W4A4	W4A8	W5A8	W6A8	W7A8	W8A8
Uniform INT (GE)	225 087	383 966	475 852	594 766	714 383	787 659
PoT	0.671	0.594	0.552	1.025	1.463	2.399
APoT	0.759	0.679	0.726	1.210	1.317	–
LogART	0.695	0.624	0.682	0.775	0.975	1.445
FPoT	0.695	0.624	0.682	0.775	0.975	1.445
DMPoT	**0.702**	**0.618**	**0.680**	**0.844**	**0.816**	**0.916**

tiles. At b=5 a separate set of ten takes their place: `int` and `pot` again, two DMPoT-5 realizations (`dmpot5c_unf`, `dmpot5c_fold`), the folded `fpot5_fold`, three APoT-5 realizations (`apot5_base`, `apot5_unf_oddcore`, `apot5_fold`) and two LogART-5 realizations (`logart5_unf_oddcore`, `logart5_fold`). At b=6–8 the sweep prices the chain (`stair`), APoT's three-alphabet PE (`apot6`/`apot7`), and the full-range shift engines `potw`/`fpotw` (§F.1). Every reported family cell is the cheapest synthesized implementation of its grid (at W4A8, for example, APoT's cheapest implementation is its folded form). The INT baseline enters every ratio in one implementation, a behavioral signed multiply lowered by the toolchain. A Booth-recoded multiplier synthesized in the same harness is smaller ($0.848\times$ at W4A4, $0.828\times$ at W4A8). Normalizing to the Booth tile instead divides every ratio against INT by that factor, which raises the folded tiles to $0.828\times$/$0.746\times$ for DMPoT, $0.820\times$/$0.754\times$ for FPoT and $0.895\times$/$0.820\times$ for APoT, and changes none of the ratios among the grids. At these widths the multiplier is small and shares the equalized reduction tree, so the asymmetry of the implementation search is noted rather than corrected.

Synthesis is Yosys 0.68 (Wolf & Glaser, 2013) with `synth -flatten` followed by `abc -liberty` against `sky130_fd_sc_hd__tt_025C_1v80` (SkyWater Technology & Google, 2020), reporting `stat -liberty` chip area. Flattening before technology mapping is what makes the measurement meaningful. It is the step that lets the synthesizer see that $3x_i$ is derived once per lane rather than once per element. Area is converted to gate equivalents by dividing by the `nand2_1` cell area, $3.7536\ \mu\mathrm{m}^2$. Weight codes are runtime inputs, and the per-lane operands are derived inside the tile rather than precomputed outside it.

One toolchain, one number, and what a second one changes. All numbers in this paper come from a single Yosys build and synthesis script, and the ratios are comparable only within one build. Another version moves absolute areas by a few percent, and not uniformly across engines. We therefore re-ran every cell the paper reads (the four-bit M and N sweeps and the five-bit anchors, every engine) under Yosys 0.33 and 0.9 as well (Table 11). The `proc` pass Yosys 0.9 needs before `flatten` moves no tabled ratio by more than 0.009. The ordering among the listed grids is preserved at both four-bit configurations under every build, and the folded tile remains the cheapest listed datapath that is near-lossless against uniform INT (the indented *unfolded* row is a decomposition step of the same grid). Its W4A4 ratio is 0.702, 0.750 and 0.783 under Yosys 0.68, 0.9 and 0.33, and its margin over APoT's cheapest implementation stays between 7.5% and 9.7% across builds and both configurations, never inverting. The other build columns correlate with the paper column at 0.963 (0.33) and 0.939 (0.9). The ordering among the five-bit grids does not survive. DMPoT-5 and LogART-5 are the two cheapest under every build but trade places, and §6.3 rests nothing on that ordering.

A second axis: the standard-cell library. A fold trades adders for multiplexers, which libraries price differently, so we also held the build and script fixed and swapped only the liberty file, re-synthesizing the $(32, 32)$ anchors against IHP SG13G2 (130 nm), the Nangate 45 nm Open Cell Library and the ASAP7 predictive FinFET library (7 nm), the last two the libraries most often reported in the quantization and open-EDA literature (right-hand columns of Table 11, each normalized to the INT tile of its own library). The four-bit ordering is the same in every library at both activation widths except on IHP at W4A8, where the folded tile undercuts shift-only PoT. The folded tile's margin over APoT's cheapest implementation stays between 4.3% (ASAP7, W4A4) and 10.4% (IHP, W4A8), and the library columns correlate with the paper column at 0.977 to 0.993. GF180MCU and the high-speed sky130 flavor behave the same way and are omitted. The five-bit grids reorder

Table 11: Toolchain and library sensitivity of the reported ratios at $(N, M)=(32, 32)$, every cell normalized to the INT tile of the same build and library (§E.1). The three sky130 columns are Yosys 0.68 (the paper's), 0.33 and 0.9. The remaining columns are the paper's build against three further libraries, IHP SG13G2 (130 nm), Nangate45 (45 nm), and ASAP7 (7 nm). The indented DMPoT row is the order-only tile of Table 12. †Yosys 0.9 carries one extra `proc` pass before `flatten`, priced in §E.1.

		sky130 `hd`, three builds			paper build, other libraries		
Config	Grid	0.68 *(paper)*	0.33	0.9†	IHP	NG45	ASAP7
W4A4	shift-only PoT	0.671	0.748	0.713	0.722	0.664	0.711
	APoT (cheapest realization)	0.759	0.867	0.818	0.785	0.735	0.773
	folded DMPoT-4	0.702	0.783	0.750	0.738	0.686	0.741
	unfolded, same operand order	0.746	0.840	0.787	0.789	0.714	0.763
W4A8	shift-only PoT	0.594	0.620	0.613	0.612	0.583	0.609
	APoT (cheapest realization)	0.679	0.715	0.702	0.677	0.645	0.660
	folded DMPoT-4	0.618	0.657	0.640	0.607	0.584	0.612
	unfolded, same operand order	0.692	0.747	0.710	0.688	0.665	0.695
W5A8	shift-only PoT	0.552	0.560	0.532	0.567	0.536	0.575
	APoT-5 (cheapest realization)	0.726	0.744	0.707	0.717	0.687	0.724
	LogART-5	0.682	0.724	0.658	0.683	0.652	0.693
	fixed-ratio k=4 control	0.737	0.789	0.706	0.723	0.686	0.721
	folded DMPoT-5	0.680	0.729	0.701	0.677	0.647	0.683
chain-b vs the same-width INT multiplier tile							
W5A8	chain-5	0.796	0.857	0.779	0.804	0.760	0.811
W6A8	chain-6	0.844	0.869	0.803	0.850	0.820	0.857
W7A8	chain-7	0.816	0.876	0.829	0.818	0.779	0.839
W8A8	chain-8	0.916	0.941	0.886	0.908	0.882	0.915

across libraries as they do across builds, while the chain members stay below same-width INT in every library (0.76–$0.92\times$ at W5–W8A8), the one claim §6.3 rests on.

What the adder budget counts. The budget is counted *per weight*: the adders that realize the one in-octave coefficient a stored weight selects, not those a folded lane precomputes. The two coincide only at b=4, where the alphabet $\mathcal{Q}=\{1, 3\}$ needs the single shared product $3x$. At b=5 the deployed alphabet $\mathcal{Q}=\{1, 3, 5, 7\}$ keeps every coefficient one-adder realizable while the lane precomputes three shared products, amortized over all output channels of a tile. The per-weight count is what bounds the width.

Five-bit synthesis note. The DMPoT-5 tiles ($\mathcal{Q}=\{1, 3, 5, 7\}$) implement the deployed chain member $[1, 2, 4^3]$. In the accuracy diagnostic behind §6.3, the off-chain schedule $[1, 1, 4^3]$, which shares $\mathcal{Q}=\{1, 3, 5, 7\}$ but not this tile, scores $6.31_{\pm.01}$ PPL / $68.2_{\pm.3}$ 0-shot on Llama-3-8B against the deployed chain member's $6.30_{\pm.00}$ / $67.9_{\pm.3}$. A proxy-MSE gap of this size does not resolve end to end. The four-to-five-bit erosion of the fold margin itself is robust. It reproduces under both other builds we synthesized ($0.657 \to 0.729$ under Yosys 0.33 and $0.640 \to 0.701$ under 0.9).

Accumulator-width audit. The equivalence work (`sat_pe.py`) surfaced one width underprovision. With the harness's uniform $W{+}A{+}8$ accumulator, the LogART-5 grid, whose integerized levels reach 256, can wrap on the all-corner input region from N=32. The committed `logart5` cells are re-synthesized at the corrected $W{+}A{+}10$ width (every other engine's per-lane maximum keeps $W{+}A{+}8$ overflow-free at the swept N), which moves the LogART-5 anchor from 0.720 to $0.724\times$ under Yosys 0.33, where both accumulator widths were synthesized, and reorders nothing. The committed `logart5` cells are the corrected width under every build (0.682 canonical, Table 11).

E.2 Per-variant areas and the amortization curve

Table 12 gives the operating point $(N, M)=(32, 32)$ with the two-step (order, fold) decomposition and both APoT implementations. Table 13a sweeps M at N=32 and Table 13b sweeps N at M=32, both up to 64, so the anchor is bracketed on both array dimensions.

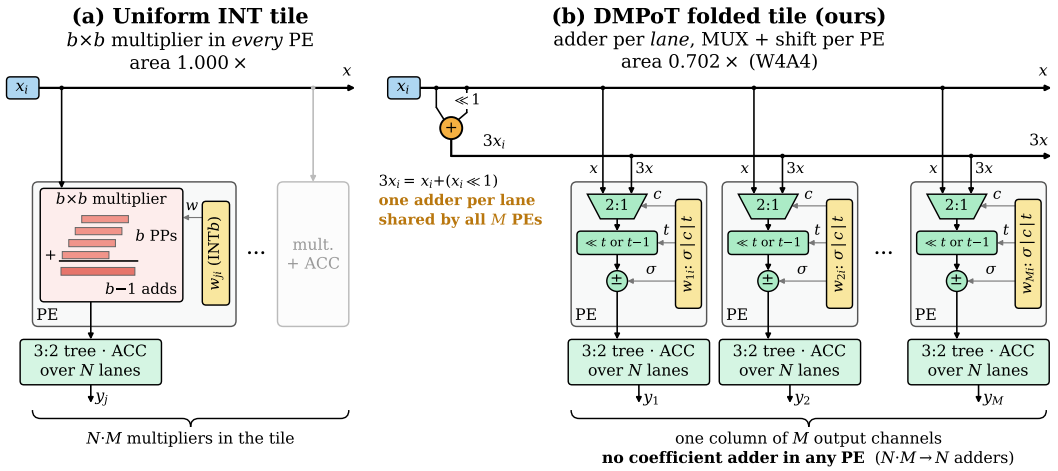

Figure 8: Uniform INT and folded DMPoT tile datapaths, one activation lane in cross-section. **(a)** uniform INT tile, a multiplier per PE. **(b)** folded tile, one $3x_i$ adder per activation lane at the array edge (§5). Areas are in Tables 10 and 12.

Reading the anchor table across widths. This paragraph expands the anchor-table reading of §5. The INT row of Table 10 carries absolute gate-equivalents, exposing the $1.24\times$ INT5/INT4 growth. Read against it, the eight-bit-activation deployment costs $1.71\times$ the W4A4 INT tile for INT but $1.05\times$ for DMPoT, a harness-scoped comparison (§E.4). Folding itself is a property of the lattice, not of the schedule, which is why §5 prices the FPoT grid on a folded tile as well.

Order and folding. The DMPoT saving over INT decomposes into two steps. Multiplying the narrow operand first (*order*) and hoisting the per-PE adder to one per lane (*fold*) contribute -16.0% and -6.0% at W4A4 and -16.6% and -10.8% at W4A8, the reordering dominating at both widths. The split between the two steps is itself the most toolchain-sensitive quantity we measure, and only at the wide activation. The W4A4 split is stable across builds (-16.8%/-4.7% under Yosys 0.9 and -9.0%/-6.8% under 0.33, the reordering dominant in all three). The W4A8 split ranges from -17.5%/-9.9% under 0.9 to -6.9%/-12.0% under 0.33, the one build in which folding, not the reordering, dominates the wide-activation tile (Table 11). We therefore read W4A4 as the reliable statement of the decomposition and treat the W4A8 split as bracketed by the builds. The *total* is robust in all three (the folded tile is the cheapest listed near-lossless datapath under every build), and so is the sharing evidence below, which does not depend on the split. APoT folds too. Its folded tiles reach $0.759\times$ (W4A4) and $0.679\times$ (W4A8) but never fall below folded DMPoT (0.702/0.618), and at $b{=}4$ folding buys APoT -2.7% against -6.0% for DMPoT (a 0.06% tie under Yosys 0.33, a 1.7% cost under 0.9). The larger in-octave alphabet needs more shared lane products, which consume what the fold saves. The fold's advantage is a sharing effect, measured against the same-order unfolded tile (the *order* row above, not the per-PE-adder row that Table 13a labels "unfolded"). It is absent at $M{=}1$, where folding costs 1.1% (W4A4) and 7.0% (W4A8), profitable from $M_{\text{cross}}{=}4$ (W4A4) and 2 (W4A8), and saturated by $M{=}32$ (Table 13a). Along the other dimension the folded tile and the INT tile both scale near-linearly in N, so the ratio of Table 13b is flat, and the small-N points carry more synthesis noise than the rest of the sweep (the $N{=}1$ point). Figure 8 draws one activation lane of the folded tile in cross-section, against the per-PE multiplier it replaces.

Per-module accounting. Synthesized standalone at W4A4, a folded element measures about 12% *larger* than an unfolded one. The module boundary presents $x_i{\ll}1$ and $3x_i$ as free primary inputs, so the element is charged for the wider operands and credited with none of the sharing that pays for them. Summing per-module areas therefore predicts folding costs $1.12\times$ *more*. The flattened tile, in which $3x_i$ is derived once from the shared eight-bit x_i, measures a 6.0% saving at W4A4 (10.8% at W4A8), which is why we report only flattened-tile area.

Insensitivity to the reduction depth. Sweeping $N \in \{8, 16, 32, 64\}$ at $M{=}32$ keeps the folded-to-INT ratio in a narrow band (0.702–0.737 at W4A4, 0.614–0.642 at W4A8) and below APoT at every N. Folding saturates by $M{=}32$ and the N-axis is flat, so the $(32, 32)$ anchor is a genuine asymptote.

Table 12: Area of a 32×32 MAC tile (32 activation lanes $\times$ 32 output channels) in gate equivalents and as a ratio to the uniform INT tile (flow in §E.1). The three DMPoT rows are one grid at three stages of the datapath (§E.2).

	W4A4		W4A8	
Engine	GE	/ INT	GE	/ INT
Uniform INT	225 087	1.000	383 966	1.000
PoT (k=1, shift only)	151 088	0.671	227 992	0.594
APoT n=2, unfolded	175 571	0.780	275 681	0.718
APoT n=2, folded	170 818	0.759	260 575	0.679
DMPoT unfolded	199 996	0.889	318 701	0.830
DMPoT order-only	168 012	0.746	265 782	0.692
DMPoT folded	158 007	**0.702**	237 187	**0.618**

Table 13: Area relative to the same-bit uniform INT tile across both array dimensions. **(a)** output-channel count M at N=32 lanes. **(b)** lane count N at M=32. In both panels the three DMPoT rows are the stages of Table 12. M=1 (nothing shared across output channels) and N=1 (a single lane) are boundary cases. All rows have four-bit weights, and five-bit members are priced at $(32, 32)$ only (Table 10).

(a) output channels M, at N=32

Config	Engine	M=1	M=2	M=4	M=8	M=16	M=32	M=64
W4A4	PoT (k=1)	0.694	0.674	0.680	0.673	0.672	0.671	0.669
	APoT n=2 (unfolded)	0.801	0.768	0.791	0.779	0.779	0.780	0.780
	DMPoT unfolded	0.918	0.900	0.896	0.899	0.894	0.889	0.894
	DMPoT order-only	0.755	0.738	0.761	0.748	0.747	0.746	0.746
	DMPoT folded	0.764	0.749	0.726	0.715	0.704	0.702	0.699
W4A8	PoT (k=1)	0.612	0.588	0.601	0.593	0.598	0.594	0.597
	APoT n=2 (unfolded)	0.736	0.722	0.726	0.715	0.723	0.718	0.727
	DMPoT unfolded	0.834	0.841	0.831	0.834	0.835	0.830	0.838
	DMPoT order-only	0.699	0.694	0.700	0.697	0.691	0.692	0.700
	DMPoT folded	0.748	0.685	0.656	0.640	0.628	0.618	0.621

(b) reduction lanes N, at M=32

Config	Engine	N=1	N=2	N=4	N=8	N=16	N=32	N=64
W4A4	PoT (k=1)	0.640	0.672	0.678	0.671	0.696	0.671	0.707
	APoT n=2 (unfolded)	0.790	0.801	0.840	0.791	0.814	0.780	0.816
	DMPoT unfolded	0.966	1.004	0.960	0.924	0.925	0.889	0.918
	DMPoT order-only	0.753	0.768	0.786	0.750	0.764	0.746	0.771
	DMPoT folded	0.660	0.704	0.737	0.703	0.723	0.702	0.737
W4A8	PoT (k=1)	0.629	0.590	0.603	0.586	0.613	0.594	0.628
	APoT n=2 (unfolded)	0.738	0.769	0.769	0.722	0.740	0.718	0.739
	DMPoT unfolded	0.912	0.904	0.861	0.857	0.847	0.830	0.849
	DMPoT order-only	0.788	0.726	0.730	0.687	0.712	0.692	0.720
	DMPoT folded	0.627	0.616	0.634	0.614	0.631	0.618	0.642

The scatter across N is structural, not run-to-run. The flow is deterministic and N inputs do not group evenly into 3:2 compressors.

Per-output cost decomposition. A least-squares fit $\mathrm{area}(M) = c_0 + c_1 M$ at N=32 separates the shared cost from the replicated one. At W4A4 the folded tile pairs the lowest marginal cost after PoT ($c_1 = 4899$ GE per output channel against INT's 7020) with the largest fixed term of the engines ($c_0 = 781$ GE, the size of N=32 shared adders plus the operand derivation at the array edge). At W4A8 the same fit gives $c_0 = 2122$ GE and $c_1 = 7349$ GE against INT's 11891. Folding raises c_0 and lowers c_1, so it loses at M=1 and wins once M amortizes the shared adders. The intercepts also absorb a few percent of nonlinearity, so the claim rests on the slopes and the M-curve of Table 13a.

E.3 EQUIVALENCE CHECKING

Area is only meaningful if the engines compute what they claim to. The committed scripts `sat_pe.py` and `sat_apotw.py` prove, by exhaustive SAT inside Yosys (`sat -verify` on a flattened miter, all input assignments, $A \in \{4, 8\}$):

- every grid engine's processing element equals its golden (integerized level) $\times\, x$ reference: the $\{1, 3\}$-lattice forms of DMPoT-4 and FPoT (unfolded and folded), all three APoT realizations at b=4 and b=5 and the two-adder b=6, 7 elements, DMPoT-5, the k=4 control, LogART-5, and the chain (`stair`) engines at b=6–8. The factor of 2 (or 4) from integerizing the in-octave coefficient of FPoT and LogART (or the k=4 control), which DMPoT's integer levels do not need, is absorbed into the static per-channel scale, so the folded array reproduces the unfolded array's outputs bit-exactly and every accuracy number in this paper transfers to it unchanged. For the folded elements the miter also rebuilds the shared lane operands ($3x$, $5x$, $7x$, $9x$, $33x$), so the proof covers the lane-precompute composition;
- complete tiles (lane precompute, processing elements, carry-save tree, and final adder) equal a behavioral golden dot product at small sizes ($N \leq 4$, `sat_pe.py --tile`). A full SAT proof at the measured $(32, 32)$ point is out of reach. The composition argument there is structural, since the same generator emits every size;
- the one exception: the shift-only PoT element is exact only up to its $W+A$-bit datapath. Its widest shifts (2^6 at b=4, up to 2^{14} at b=5) clip, and SAT proves it correct modulo 2^{W+A}. A full-range PoT datapath would be wider and costlier than the one priced here, so the PoT ratios are a *lower* bound on a bit-exact implementation, a conservative direction for every comparison in this paper, since PoT serves only as the cheapest-and-unusable reference point.

E.4 WHAT THE AREA NUMBERS DO NOT INCLUDE

This subsection supports the limitations of §7. We restate the limits of §5 in full because they bound every hardware claim in the paper.

Absolute GE is harness-dependent. The accumulator width $W+A+8$, the in-element sign–magnitude handling and the explicit 3:2 reduction are choices of this harness. A different set of choices changes the absolute counts substantially (an independent implementation we compared against produced INT tiles roughly $1.3\times$ larger). The INT-normalized ratios are the portable quantity, and absolute figures should be quoted only together with the harness specification.

Synthesis is untimed and the tiles are combinational. No clock or delay target is given to ABC, so every area is a mapped-cell area at unconstrained delay, with no buffering or gate sizing for timing, and no measurement of how the ratios move under timing closure. The tiles also contain no sequential cells. The accumulator is a combinational output, and output registers, pipeline stages, and the clock tree are all outside the measurement. Both omissions are uniform across engines, and the excluded sequential overhead is a cost every engine shares, so pricing it would dilute every ratio toward one rather than reorder the comparison, but the dilution itself is real, and the ratios here are datapath-only.

Interconnect is not priced. Gate-area synthesis does not model routing. The folded array broadcasts both x_i and $3x_i$ down each column, which is $2.5\times$ the activation-broadcast wiring of the unfolded array at W4A4 (4 and 6 bit) and $2.25\times$ at W4A8 (8 and 10 bit). This is the one cost the folded design plausibly pays and that our measurement cannot see. Settling it requires place-and-route.

Gate area, not power. We report standard-cell area for the processing elements and the MAC array, and nothing else. No power or energy figure is claimed. Switching power depends on operand statistics, the clock tree, wire load, and memory traffic, none of which a combinational tile synthesis observes. A credible power comparison needs end-to-end RTL and, ultimately, silicon. We therefore treat the area ratio as the hardware claim and leave power characterization to future work rather than substituting a proxy for it.

Weight MACs only. The tiles model the weight-MAC datapath of the linear projections. Attention-score computation, KV-cache storage and traffic, activation memory, and control are all outside the measurement. End-to-end benefit therefore scales with the share of inference cost attributable to

Table 14: Weight bit-width sweep on Llama-3-8B at fixed **A8KV8** (mean$_{\pm\text{std}}$ over three calibration seeds). At every width the DMPoT row is the chain-b member, DMPoT-4 at b=4. FPoT holds its defining density k=2 at every bit-width. APoT is undefined at b=8 (–). Rows and preprocessing protocol as in Table 1. LogART (†) is a post-training transplant on the uniform INT preprocessing (Appendix A.5). PPL: WikiText-2 (↓). 0-shot^9: nine-task mean (↑, %).

	W4A8KV8		W5A8KV8		W6A8KV8		W7A8KV8		W8A8KV8	
Weight Quant	PPL↓	0-shot^9↑	PPL↓	0-shot^9↑	PPL↓	0-shot^9↑	PPL↓	0-shot^9↑	PPL↓	0-shot^9↑
Uniform	$6.60_{\pm.01}$	$67.5_{\pm.1}$	$6.29_{\pm.00}$	$68.0_{\pm.0}$	$6.20_{\pm.00}$	$68.0_{\pm.2}$	$6.17_{\pm.00}$	$68.1_{\pm.1}$	$6.16_{\pm.00}$	$68.2_{\pm.1}$
PoT	$7.64_{\pm.02}$	$64.0_{\pm.7}$	$7.65_{\pm.03}$	$63.4_{\pm1.4}$	$7.67_{\pm.05}$	$63.6_{\pm.4}$	$7.68_{\pm.02}$	$63.9_{\pm.6}$	$7.52_{\pm.12}$	$63.8_{\pm.7}$
APoT	$6.67_{\pm.01}$	$67.8_{\pm.5}$	$6.40_{\pm.01}$	$68.1_{\pm.4}$	$6.25_{\pm.02}$	$68.2_{\pm.1}$	$6.21_{\pm.00}$	$68.2_{\pm.1}$	–	–
LogART†	$11.34_{\pm.66}$	$57.7_{\pm1.2}$	$11.73_{\pm.87}$	$55.1_{\pm2.9}$	$14.15_{\pm2.74}$	$55.0_{\pm1.8}$	$15.24_{\pm1.93}$	$52.6_{\pm2.3}$	$21.64_{\pm10.66}$	$48.4_{\pm8.3}$
FPoT	$6.85_{\pm.01}$	$67.3_{\pm.5}$	$6.72_{\pm.01}$	$66.4_{\pm.2}$	$6.73_{\pm.01}$	$65.6_{\pm.7}$	$6.74_{\pm.01}$	$66.1_{\pm.4}$	$6.58_{\pm.00}$	$67.7_{\pm.2}$
DMPoT (Ours)	$\mathbf{6.67}_{\pm.01}$	$\mathbf{67.3}_{\pm.3}$	$\mathbf{6.30}_{\pm.00}$	$\mathbf{67.9}_{\pm.3}$	$\mathbf{6.19}_{\pm.00}$	$\mathbf{68.2}_{\pm.1}$	$\mathbf{6.17}_{\pm.00}$	$\mathbf{68.1}_{\pm.1}$	$\mathbf{6.16}_{\pm.00}$	$\mathbf{68.1}_{\pm.1}$

Table 15: MAC-tile area of the sweep widths b=5–8 at the $(32, 32)$ point (Yosys 0.68, sky130), as a ratio to the same-width INT tile, whose GE is the first row. At b=5 the grid rows repeat the W5A8 column of Table 10, where DMPoT is the folded one-adder tile. At $b\geq6$ DMPoT is the chain, a w=$b-2$ odd core with three shift classes (§B.5). At these widths full-range k=2 (the FPoT grid and the LogART transplant) and PoT are SAT-exact shift engines with accumulators widened to their span, and APoT is a three-alphabet PE, undefined at b=8.

b	5	6	7	8
uniform INT$_b$ tile (GE)	475 852	594 766	714 383	787 659
PoT (k=1) / INT$_b$	0.552	1.025	1.463	2.399
APoT / INT$_b$	0.726	1.210	1.317	–
full-range k=2 (LogART, FPoT) / INT$_b$	0.682	0.775	0.975	1.445
DMPoT tile (GE)	323 640	502 119	583 102	721 511
ratio to INT$_b$	**0.680**	**0.844**	**0.816**	**0.916**
odd-core width $w = b-2$	3	4	5	6
w/b	0.600	0.667	0.714	0.750

weight MACs, which is largest in the compute-bound prefill phase and smaller under memory-bound decode.

F Weight Bit-Width Sweep

This appendix collects the weight bit-width sweep: the accuracy of each chain member (Table 14), the area of the array that deploys it (Table 15), and the per-task scores behind those means (Table 16). All of it is Llama-3-8B at fixed A8KV8, three calibration seeds. The four-bit configurations of the main table are in Appendix I.

F.1 Datapaths of the wider members

The widths of the Table 14 sweep are also synthesized, so its accuracy trend can be read against what the corresponding arrays cost. Table 15 reports the deployed $(32, 32)$ DMPoT tile at each width, the folded one-adder tile at b=5 and the chain datapath beyond it, against a same-width INT multiplier tile measured in the same sweep. It remains below the INT multiplier at every width (0.680 at b=5, then 0.844, 0.816 and 0.916 at b=6–8), so the two-bit-narrower odd core of §B.5 is worth real area, but progressively less of it. The ratio w/b rises from 0.600 to 0.750 over the same range. Built as a multiplier for continuity with the wider members, the five-bit grid reads $0.796\times$ instead (chain-5, Table 11). The two figures are the same grid built two ways. Area and accuracy therefore agree on where the family is worth deploying, and both say the same thing the adder budget does.

The other grids at the same widths. APoT's levels at $b=6, 7$ are sums of at most three powers of two, so a two-adder PE realizes the grid. A per-code decode selects three shifted copies of the activation and two additions combine them, and the datapath is proved equivalent to the golden multiplier by exhaustive SAT, as for every PE in this paper. Synthesized under the same protocol it costs $1.210\times$ (b=6) and $1.317\times$ (b=7) the same-width INT tile (Table 10). The construction exists, but the three-term decode outgrows the multiplier it replaces, and APoT defines no b=8 grid. PoT,

Table 16: Per-task zero-shot accuracy for the weight bit-width sweep on **Llama-3-8B** at fixed A8KV8, b=5–8 (four-bit results in Table 22, conventions in Appendix I). APoT is undefined at b=8 (–). LogART (†) is a post-training transplant on the uniform INT preprocessing (Appendix A.5).

Config	Method	ARC-C	ARC-E	Hella	Wino	PIQA	BoolQ	OBQA	Lam	SIQA	AVG
FP (bf16) reference		53.1	77.9	79.2	73.4	81.0	81.3	45.0	75.7	47.1	68.2
W5A8KV8	Uniform	52.5	78.6	78.7	73.1	80.5	80.8	44.3	76.0	47.1	68.0
	PoT	46.0	70.9	75.3	72.1	77.7	72.5	42.7	68.6	44.9	63.4
	APoT	53.0	78.8	78.6	73.5	80.2	81.1	45.4	75.6	47.2	68.1
	LogART†	34.2	56.2	66.4	66.9	72.2	63.5	37.7	57.1	41.9	55.1
	FPoT	50.7	76.2	78.5	72.5	79.7	78.6	44.2	71.1	45.8	66.4
	DMPoT (Ours)	**53.2**	**77.8**	**78.8**	**73.5**	**80.4**	**80.0**	**44.9**	**76.1**	**46.9**	**67.9**
W6A8KV8	Uniform	53.3	78.1	78.9	73.3	80.6	80.7	44.9	75.5	46.9	68.0
	PoT	46.8	71.2	75.8	71.1	77.5	74.2	41.5	69.2	44.9	63.6
	APoT	53.9	78.5	78.9	73.7	80.5	81.7	44.5	75.4	46.8	68.2
	LogART†	35.8	57.7	65.5	66.7	70.8	68.9	36.7	52.5	40.8	55.0
	FPoT	49.5	73.8	78.1	72.6	79.5	75.3	44.1	70.9	46.3	65.6
	DMPoT (Ours)	**54.1**	**78.3**	**79.1**	**73.2**	**80.6**	**81.5**	**44.7**	**75.6**	**47.0**	**68.2**
W7A8KV8	Uniform	53.5	77.8	79.2	73.6	80.7	80.9	44.7	75.3	47.1	68.1
	PoT	46.1	71.1	75.9	72.3	77.6	77.0	42.5	67.6	45.0	63.9
	APoT	53.8	78.1	79.0	74.0	80.8	81.4	44.7	75.4	47.1	68.2
	LogART†	33.0	54.3	63.0	64.6	70.9	67.1	35.3	44.9	40.8	52.6
	FPoT	50.9	75.4	78.2	72.2	79.4	77.2	44.5	70.9	46.5	66.1
	DMPoT (Ours)	**53.6**	**78.3**	**79.1**	**73.4**	**80.6**	**81.1**	**44.6**	**75.6**	**47.0**	**68.1**
W8A8KV8	Uniform	53.4	78.2	79.1	73.6	80.7	81.3	44.9	75.4	47.0	68.2
	PoT	46.8	72.9	75.6	72.6	77.9	70.8	41.7	70.5	45.4	63.8
	APoT	–	–	–	–	–	–	–	–	–	–
	LogART†	32.0	50.6	56.4	65.2	67.9	57.7	35.3	30.5	40.0	48.4
	FPoT	52.3	77.5	77.8	74.3	80.1	80.9	44.8	75.3	46.5	67.7
	DMPoT (Ours)	**53.0**	**77.9**	**79.1**	**73.4**	**80.7**	**81.1**	**45.1**	**75.2**	**47.1**	**68.1**

FPoT and LogART reach synthesis only with an accumulator widened to their span. With in-octave density pinned, added bits buy range, and their level spans at b=6–8 (15–126 octaves) overflow the shared $W{+}A{+}8$ accumulator. The `potw` and `fpotw` engines therefore carry the full product ($A{+}2^{b-1}{-}1$ and $A{+}2^{b-2}{+}1$ bits) with a matching accumulator, and are SAT-proved exact like every other PE (the b=4, 5 `pot` engine clips instead). Priced that way (Table 15), PoT costs $1.025\times$, $1.463\times$ and $2.399\times$ the same-width INT tile at b=6, 7, 8, and the full-range k=2 lattice, which is both the FPoT grid and the LogART transplant's deployed grid at these widths, $0.775\times$, $0.975\times$ and $1.445\times$. The log-uniform grid stays below INT through b=7 and crosses above it at b=8, while its accuracy has already stalled (Table 14). On the absolute accuracy–area plane every one of these points is dominated by a lower-width chain member (the folded five-bit tile, 324 kGE at 6.30 PPL, against 461 kGE at 6.73 for the k=2 grid at b=6 in Fig. 3a). The chain therefore remains the only grid in Table 14 that is both measured and on the Pareto front beyond five bits. Against uniform INT at b=8 (6.16 PPL, 788 kGE), the six-bit member matches its nine-task mean (68.2) within 0.04 PPL on 502 kGE, and the seven-bit member comes within 0.01 PPL on 583 kGE.

G Fused Triton Kernel for DMPoT Training

This appendix details the training kernel behind the setup of §6.

One kernel serves the whole family. Every deployed DMPoT-4 level lies on the $\{2^t,\ 1.5{\cdot}2^t\}$ lattice shared with the log-uniform (FPoT) member, so the schedule is a compile-time branch (`skip_e1`) over one code path, run in linear-nearest mode (`round_mode=linear`) so its decision boundaries are the arithmetic midpoints of §3. In standard PyTorch the fake-quantization pipeline (quantize, dequantize, STE) materializes about eight weight-sized intermediates per weight matrix (sign, absolute value, $\log_2$, exponent, zero and kill masks, dequantized output), and gradient checkpointing regenerates them at every recomputed layer. The fused kernel keeps them in registers and emits one weight-sized output. It selects the nearest level, dequantizes with the factor $2^{\lfloor e/2 \rfloor}$ at even e and

$\mu \cdot 2^{\lfloor e/2 \rfloor}$ at odd e, where the in-octave mantissa μ is a kernel argument ($\mu{=}\sqrt{2}$ gives the exact grid through which the DMPoT transform is trained and $\mu{=}1.5$ gives the deployed levels, Appendix H.3), and passes the gradient straight through a custom `torch.autograd.Function`. The 400-step transform training of Qwen2.5-14B runs on a single A100-80GB at batch size 4 with gradient checkpointing.

Throughput, and which grids fit. Any grid on the lattice compiles into the same launch and runs 1.4–21.6× faster than the generic `searchsorted` path that an off-lattice level such as APoT's $10 = 2^3{+}2^1$ (Li et al., 2020) requires. The difference is training-loop cost, not feasibility. APoT is a straight-through target in the main results (§6.2). Arithmetic is float32, a pure-PyTorch fallback serves environments without Triton, and Algorithm 1 (Appendix A.6) states the quantizer whose straight-through branch the kernel fuses.

H Additional Ablations

This appendix supports the ablation summary of §6.4. Every sweep in it was re-run on the deployed DMPoT-4 grid rather than carried over from measurements on the log-uniform ($J{=}0$, FPoT) member of the family. No number is transferred from another grid.

H.1 Calibration sensitivity

A common concern for post-training quantization methods is sensitivity to the calibration set, in both size and domain. We evaluate DMPoT on Llama-3-8B (W4A4KV4) while varying: (1) the number of calibration samples $n \in \{128, 256, 512\}$, and (2) the calibration dataset, WikiText-2 (Merity et al., 2017) (the default used throughout the paper) versus C4 (Raffel et al., 2020) (a large-scale web-crawled corpus). Every row resumes the deployed checkpoint, whose transform was trained on WikiText-2, so only the GPTQ calibration set changes. The GPTQ settings are fixed.

Table 17: Calibration sensitivity on Llama-3-8B (W4A4KV4, DMPoT-4, 400 steps) across calibration dataset and sample count n. Seed 42 only. Every row resumes the deployed checkpoint, so only the GPTQ calibration set changes. ΔPPL is versus WikiText-2 $n{=}128$ at the same seed (7.441). The three-seed deployed mean is 7.417 (Table 1).

Cal. Dataset	n	PPL↓	0-shot[9]↑ (%)	ΔPPL
WikiText-2	128	7.441	65.16	—
	256	7.441	64.89	0.000
	512	7.446	65.09	+0.005
C4 (Raffel et al., 2020)	128	7.508	65.22	+0.066
	256	7.522	65.14	+0.080
	512	7.525	64.58	+0.084

What this sweep tests. Two robustness questions are separable here. The first, sample count, is whether $n{=}128$ suffices, or whether the per-channel scale search and GPTQ's error compensation keep improving with more calibration data. The second, domain, is whether GPTQ calibrated on a different corpus reaches the same accuracy. Table 17 answers both for the GPTQ stage under a fixed transform. Sample count does not matter. WikiText-2 moves by 0.005 PPL between $n{=}128$ and $n{=}512$. Domain matters slightly. A C4 calibration set costs 0.066–0.084 PPL at every n, an order of magnitude below the grid effects this paper measures. The default setting used everywhere else in this paper is $n{=}128$ on WikiText-2.

H.2 Preprocessing grid versus deployed grid

Table 18 crosses the grid the preprocessing transform was optimized for (P, for DMPoT its exact grid in Appendix H.3) with the grid deployed at quantization (Q). The diagonal is the co-designed protocol of Table 1. The $P{=}$INT row is the mismatch protocol, which the main table keeps only for LogART, a Q-only grid whose learned rounding is too costly to repeat inside the straight-through loop (Appendix A.5). The axes are far from symmetric. Across the co-designed columns, changing Q

Table 18: **Preprocessing grid P (rows) $\times$ deployed grid Q (columns)**. Llama-3-8B W4A4KV4, WikiText-2 PPL↓, mean±sample std over three calibration seeds (42/0/123). The bold diagonal is the co-designed protocol of Table 1. LogART, a Q-only grid, has no P row.

	Deployed grid Q					
Preprocessing grid P	INT	PoT	APoT	LogART	FPoT	DMPoT
INT	**7.358** ±.074	9.160 ±.126	7.461 ±.022	16.05 ±.50	7.851 ±.039	7.503 ±.022
PoT	7.493 ±.022	**8.786** ±.037	7.500 ±.006	12.10 ±1.39	7.833 ±.042	7.504 ±.014
APoT	7.378 ±.010	8.708 ±.087	**7.410** ±.027	11.92 ±.82	7.765 ±.036	7.429 ±.029
FPoT	7.437 ±.023	8.886 ±.061	7.448 ±.008	15.28 ±2.02	**7.596** ±.020	7.458 ±.029
DMPoT	7.383 ±.043	8.764 ±.072	7.392 ±.012	12.70 ±2.58	7.726 ±.024	**7.417** ±.041

at fixed P moves perplexity by up to 1.80 PPL and changing P at fixed Q by at most 0.45 (8.69 and 4.13 with the LogART column). The value of co-design, the gap from the P=INT row down to the diagonal, tracks how coarse the grid is: $+0.05$ for APoT, $+0.09$ for DMPoT, $+0.26$ for FPoT and $+0.37$ for PoT. Seed dispersion in the LogART column is large (±0.5 to ±2.6, against ±0.006 to ±0.13 elsewhere), so only its gap down from P=INT is read, not the ordering among the co-designed rows.

H.3 $\sqrt{2}$ APPROXIMATION QUALITY

This appendix validates, end to end, the approximate-grid deployment protocol of §3. The exact DMPoT-4 magnitudes are $\{1,\ 2,\ 2\sqrt{2},\ 4,\ 4\sqrt{2},\ 8,\ 8\sqrt{2}\}$. Every second level carries an in-octave ratio of $\sqrt{2}$, which no shift can produce. The substitution $\sqrt{2} \to 1.5$ is forced by the one-adder budget (§4.2), at a 6.1% ratio error. Applying it to the exact grid gives the deployed magnitudes $\{1, 2, 3, 4, 6, 8, 12\}$. The same projection applied to the J=0 member yields $\{1, 1.5, 2, 3, 4, 6, 8\}$, a different lattice. The derivation is what selects which member lands where, and the projection cost is comparable for both members ($1.007 \to 1.055$ proxy MSE for J=0 and $0.929 \to 0.984$ for $[1, 2^3]$, Table 3).

Table 19 evaluates how that 6.1% should be handled. The pipeline has two stages, transform training with straight-through gradients and calibration (per-channel scale search, grid alignment and GPTQ), and the protocols differ only in where the deployed levels enter it: in the straight-through forward pass, in the level set on which calibration takes its rounding decision, or only in the levels to which calibration and inference reconstruct each weight.

Protocol	straight-through forward	rounding decision	reconstruction
Exact-$\sqrt{2}$ reference	exact	exact	exact
Exact-grid rounding	exact	exact	deployed
Deployed-grid rounding (ours)	exact	deployed	deployed
Deployed-grid training	deployed	deployed	deployed

The reference never substitutes and has no shift–add realization. Exact-grid rounding picks each level at the midpoints of the exact grid but reconstructs it to the deployed value, so GPTQ compensates the substituted values but not the displaced decision boundaries. Deployed-grid rounding is the approximate-grid quantization of §3 and the protocol of every DMPoT row in the main tables. Deployed-grid training also trains the transform through the deployed levels. FPoT (Anonymous, 2026) names the last three Exact Grid, Approx Grid and Aware. The table also lists two-adder coefficients, counted as in §4.2, that approximate $\sqrt{2}$ more closely.

DMPoT-4 deploys the one-adder ratio with deployed-grid rounding, so that weight calibration absorbs the substitution as part of the quantization grid. In grid MSE the substitution moves the grid from 0.929 to $0.984\times$ INT4 (§4), so it is not free. End to end, the deployed grid nevertheless reaches PPL 7.417 on Llama-3-8B W4A4KV4 over three seeds, i.e. uniform INT4 accuracy. Exact-grid rounding costs $+0.090$ PPL at the deployed one-adder, a penalty monotone in the ratio error, whereas deployed-grid rounding stays within 0.024 of the exact-$\sqrt{2}$ reference at every budget. Moving the deployed levels into transform training changes perplexity by -0.009 to $+0.026$ against deployed-grid rounding, inside the seed spread of both columns. The two-adder coefficients are listed for

Table 19: How the in-octave $\sqrt{2} \to 1.5$ substitution should be handled for DMPoT-4 (Llama-3-8B W4A4KV4), with two two-adder ratios closer to $\sqrt{2}$ for comparison (adders counted as in §4.2). The protocols (where the deployed levels enter) are in Appendix H.3. Mean$_{\pm\text{std}}$ over three calibration seeds (42/0/123), with 0-shot[9] in %.

			Exact-grid rounding		**Deployed-grid rounding (ours)**		Deployed-grid training	
Realization	ratio (err.)	add.	PPL↓	0-shot[9] ↑	PPL↓	0-shot[9] ↑	PPL↓	0-shot[9] ↑
Exact-$\sqrt{2}$ reference[†]	1.414 (—)	—			PPL↓ $7.393_{\pm.029}$, 0-shot[9] ↑ $65.21_{\pm.27}$ (no substitution)			
two-adder, $1{+}2^{-1}{-}2^{-4}$	1.4375 (1.6%)	2	$7.393_{\pm.038}$	$65.28_{\pm.12}$	$7.386_{\pm.030}$	$65.18_{\pm.19}$	$7.377_{\pm.025}$	$65.19_{\pm.52}$
two-adder, $1{+}2^{-2}{+}2^{-3}$	1.375 (2.8%)	2	$7.424_{\pm.038}$	$65.35_{\pm.23}$	$7.396_{\pm.034}$	$65.28_{\pm.33}$	$7.397_{\pm.007}$	$65.35_{\pm.66}$
one-adder (deployed)	**1.5 (6.1%)**	**1**	$7.507_{\pm.036}$	$64.42_{\pm.29}$	**7.417**$_{\pm.041}$	**65.09**$_{\pm.17}$	$7.443_{\pm.028}$	$65.45_{\pm.07}$

[†] $\sqrt{2}$ is irrational and has no finite shift–add realization. It would require a multiplier-grade unit.

completeness rather than as deployment candidates, because they lie outside $\mathcal{R}_1$ (§4.2, Appendix B.7) and exceed the one-adder budget under which the deployed grid is selected.

H.4 Decision rule: zero threshold and rounding mode

We evaluate the two halves of the DMPoT decision rule, the zero threshold and the rounding mode. All experiments use Llama-3-8B (W4A4KV4, 400 OSTQuant training steps). Both sweeps are three-seed resume-evals of the same deployed DMPoT-4 checkpoint, so their blocks are directly comparable. Neither carries a measurement transferred from another grid.

The log-domain reading of the zero boundary is

$$\tau \;=\; 2^{-1/(2k_1)}, \tag{15}$$

with k_1 the density of the schedule's bottom octave. The deployed set is irregular in $\log_2$ (steps of 1, 0.585, 0.415 octaves), which is why the shipped rule is the arithmetic midpoint instead. What makes DMPoT-4 zero fewer weights than the $J{=}0$ grid (9.7% against 13.2% under the Gaussian proxy, Table 3) is not the threshold but the range. Reaching 12 rather than 8 at the same level budget lets the MSE-optimal scale s fall, and the zero bin shrinks with it (measured in Appendix C).

Values with $|w/s| < \tau$ map to zero, so $\tau \in (0, 1]$. Rows are labelled by τ rather than by the exponent α of $\tau = 2^{\alpha/k_1}$, because one default α lands on a different threshold for each grid. Setting $\alpha{=}{-}0.5$ gives $\tau{=}0.707$ for DMPoT-4 ($k_1{=}1$) and 0.841 for the $J{=}0$ grid ($k_1{=}2$).

Table 20: The two halves of the decision rule, Llama-3-8B (W4A4KV4, DMPoT-4, 400 steps, three-seed means±std). **(a)** the zero threshold τ, below which $|w/s|$ maps to zero, with $\tau = 2^{\alpha/k_1}$ and k_1 the level density of the *bottom* octave ($k_1{=}1$ for DMPoT-4). **(b)** the rounding rule, at $\tau{=}0.5$. The deployed configuration is bolded in both blocks.

(a) zero threshold				
α	τ		PPL↓	0-shot[9]↑ (%)
0	1.000	all sub-unit magnitudes zeroed	7.490 $\pm.045$	64.99 $\pm.40$
-0.25	0.841	log midpoint of the $J{=}0$ grid	7.447 $\pm.038$	65.12 $\pm.22$
-0.5	0.707	log-space zero boundary	7.434 $\pm.053$	65.13 $\pm.40$
-1	**0.500**	**arithmetic midpoint (deployed)**	**7.417** $\pm.041$	**65.09** $\pm.17$
-1.5	0.354		7.434 $\pm.051$	64.99 $\pm.57$
-2	0.250		7.445 $\pm.050$	64.93 $\pm.54$
-3	0.125		7.478 $\pm.048$	64.62 $\pm.33$
-100	≈ 0	no zeroing	7.495 $\pm.032$	64.57 $\pm.83$
(b) rounding rule, at $\tau{=}0.5$				
Log-space NN (relative distance)			7.438 $\pm.039$	64.68 $\pm.30$
Linear-space NN (deployed)			**7.417** $\pm.041$	**65.09** $\pm.17$

What this sweep tests. The deployed row is already measured. It is the configuration used everywhere else in this paper, so its entry is the main-table DMPoT result. Two other anchors matter beyond the general question of sensitivity. The log-space reading of Eq. (15) would put DMPoT-4's boundary at $\tau = 0.707$, the log-space midpoint of its bottom octave, whereas its level-to-level decision rule is

linear-nearest (§3), whose self-consistent zero boundary is the arithmetic midpoint $\tau = 0.5$ that the deployed grid uses. Whether the choice between the two costs accuracy is the sharpest question the table asks. At the other end, $\tau \to 0$ disables zeroing entirely, which isolates how much of the grid's accuracy depends on having an explicit zero at all. The answer to both (Table 20a) is that the grid is insensitive. The deployed arithmetic midpoint τ=0.5 is the best row (7.417), but the log-space τ=0.707 is only 0.017 behind and the whole mid-range $\tau \in [0.354, 0.707]$ lies within 0.02 PPL. The mismatch between the log-domain reading of Eq. (15) and the deployed linear-nearest rule therefore costs nothing measurable. The ends do cost. Zeroing every sub-unit magnitude (τ=1) and disabling zeroing altogether ($\tau \to 0$) both give back about 0.08 PPL, so an explicit zero is worth having and its exact boundary is not.

Rounding rule. Log-space nearest-neighbor rounding places decision boundaries at the *geometric mean* of adjacent levels. On a fixed-ratio grid it has the closed form $e = \text{round}(\log_2(|w/s|) \cdot k)$, but on the DMPoT-4 grid the boundaries must be taken against the deployed level set directly. The alternative is linear-space nearest-neighbor, which places boundaries at the *arithmetic mean*, potentially better aligned with MSE-based objectives such as GPTQ, since it minimizes absolute rather than relative quantization error per element, and it is what the deployed kernel implements.

Table 20b compares the two modes on the deployed DMPoT-4 grid. The question is sharper here than for a fixed-ratio geometric grid. Because DMPoT-4's in-octave ratio is 1.5 rather than $\sqrt{2}$, the geometric and arithmetic boundaries are further apart, so insensitivity cannot be assumed. The deployed kernel already rounds linear-nearest (§3, Appendix G). The measurement confirms that choice without making it critical. Linear-nearest is ahead by 0.021 PPL and 0.41%p, a gap of about half a seed standard deviation, so the two boundaries are not interchangeable but neither is decisive.

H.5 Preprocessing ablation

Table 21 ablates the type and quality of the preprocessing transform under DMPoT-4 weight quantization on Llama-3-8B W4A4KV4. Configurations vary along two orthogonal axes: rotation quality (none → fixed Hadamard → learned) and the presence of per-channel scale.

Table 21: Preprocessing component ablation within OSTQuant under DMPoT-4 weight quantization, Llama-3-8B W4A4KV4 (400 steps, three-seed means±std over seeds 42/0/123, activations and KV cache uniform INT4). The bold row is the deployed configuration, the DMPoT cell of Table 1.

Preprocessing	PPL↓	0-shot[9]↑ (%)
No transform	$59.23_{\pm.93}$	$41.15_{\pm.11}$
Scale only	$8.593_{\pm.077}$	$61.53_{\pm.08}$
Fixed Hadamard	$7.923_{\pm.105}$	$62.83_{\pm.33}$
Learned rotation	$7.752_{\pm.028}$	$63.76_{\pm.30}$
Learned rotation + scale	$\mathbf{7.417_{\pm.041}}$	$\mathbf{65.09_{\pm.17}}$

What this sweep tests. The two axes are expected to act on different failure modes. Rotation suppresses position-level outliers, which a logarithmic grid resolves poorly, while per-channel scale normalizes amplitude so that the grid's dynamic range is used. Table 21 measures both. Neither is sufficient alone, and the transform-free configuration is not merely worse but unusable (PPL 59.2, 41.2%p 0-shot, near chance). Rotation is the larger lever. It alone reaches 7.752 against 8.593 for scale alone, and learning it beats a fixed Hadamard (7.752 vs 7.923) while collapsing the across-seed spread from ±0.105 to ±0.028. Scale then supplies the remaining 0.335 PPL. The ordering matches the FPoT-grid result but the magnitudes are milder throughout, which is the density-matched grid being less dependent on the transform than the log-uniform one.

I Per-Task Downstream Results

This appendix reports individual zero-shot task scores for all models and bit-width configurations in Tables 1 and 14. For each task we report the metric the harness reports as primary: `acc_norm` for ARC-Challenge, ARC-Easy, HellaSwag, OpenBookQA and PIQA, `acc` for the remaining four. The "AVG" column, the 0-shot[9] metric of the main paper, is therefore the unweighted mean of the nine cells beside it. All scores are in %. Six weight-grid methods are compared: **Uniform** (uniform INT

baseline), **PoT** (base 2, weight-only), **APoT** (additive PoT), **LogART** (log-based), **FPoT** (base $\sqrt{2}$), and **DMPoT (Ours)** (the density-matched foldable grid of §3). The unquantized FP (bf16) reference is shown above each model's quantized results. Tables 22, 23, 24, 25, 26, 27 and 28 report per-task results for the two main-table configurations, W4A4KV4 and W4A8KV8. The per-task scores of the bit-width sweep are in Table 16 (Appendix F). **All cells are three-seed means** (seeds 42/0/123). The per-task means therefore average to the same 0-shot[9] as the source tables.

Table 22: Per-task zero-shot accuracy for **Llama-3-8B** (conventions in Appendix I). LogART (†) is a post-training transplant on the uniform INT preprocessing (Appendix A.5).

Config	Method	ARC-C	ARC-E	Hella	Wino	PIQA	BoolQ	OBQA	Lam	SIQA	AVG
FP (bf16) reference		53.1	77.9	79.2	73.4	81.0	81.3	45.0	75.7	47.1	68.2
W4A4KV4	Uniform	49.2	75.1	76.0	69.2	78.2	78.2	41.9	73.4	45.6	65.2
	PoT	41.1	64.9	71.9	66.6	74.6	68.8	39.3	65.5	43.2	59.5
	APoT	49.3	75.2	76.0	69.5	78.0	79.2	42.5	72.8	45.3	65.3
	LogART†	30.2	49.8	59.1	60.6	65.8	63.6	34.5	45.7	41.0	50.0
	FPoT	47.1	75.2	74.9	69.1	77.8	79.2	42.1	73.1	44.5	64.8
	DMPoT (Ours)	**48.5**	**74.8**	**75.9**	**70.7**	**77.4**	**78.9**	**41.9**	**72.2**	**45.4**	**65.1**
W4A8KV8	Uniform	52.2	78.6	78.0	72.6	80.0	80.0	44.1	75.4	46.5	67.5
	PoT	45.7	71.2	75.4	71.1	77.9	78.0	41.8	69.5	45.0	64.0
	APoT	52.0	78.9	77.8	73.9	79.9	80.2	45.0	75.9	46.2	67.8
	LogART†	37.7	61.2	67.3	68.8	71.8	68.7	39.3	61.3	43.3	57.7
	FPoT	51.9	78.9	76.9	73.4	79.2	81.0	43.5	75.3	45.7	67.3
	DMPoT (Ours)	**50.6**	**76.5**	**77.9**	**73.9**	**80.0**	**81.4**	**44.0**	**75.3**	**46.2**	**67.3**

Table 23: Per-task zero-shot accuracy for **Llama-2-7B** (conventions in Appendix I). LogART (†) is a post-training transplant on the uniform INT preprocessing (Appendix A.5).

Config	Method	ARC-C	ARC-E	Hella	Wino	PIQA	BoolQ	OBQA	Lam	SIQA	AVG
FP (bf16) reference		45.9	74.5	75.9	68.8	78.7	78.0	44.2	73.6	46.0	65.1
W4A4KV4	Uniform	42.6	70.3	73.1	66.6	77.4	74.2	40.2	70.7	43.9	62.1
	PoT	38.5	63.5	68.9	63.5	75.5	68.5	38.1	63.8	41.7	58.0
	APoT	42.0	70.0	72.8	65.8	76.9	74.1	40.9	69.9	43.8	61.8
	LogART†	31.7	55.1	49.8	56.8	65.6	60.7	32.8	26.4	38.8	46.4
	FPoT	41.8	70.0	72.0	66.6	76.9	74.2	40.1	70.6	44.8	61.9
	DMPoT (Ours)	**42.5**	**69.6**	**72.4**	**66.9**	**76.8**	**74.9**	**40.4**	**71.3**	**43.8**	**62.1**
W4A8KV8	Uniform	44.3	73.2	74.5	68.7	78.3	76.2	41.7	72.6	45.1	63.8
	PoT	42.3	69.1	72.1	67.5	76.6	73.3	40.7	67.3	44.0	61.4
	APoT	44.0	72.7	74.3	69.0	78.5	76.5	43.5	72.8	45.3	64.1
	LogART†	34.1	58.7	55.0	63.5	70.9	61.4	37.1	42.5	39.8	51.4
	FPoT	44.0	71.8	74.1	68.7	78.3	76.7	42.0	73.4	44.7	63.7
	DMPoT (Ours)	**44.8**	**73.0**	**74.1**	**69.3**	**78.1**	**76.3**	**42.3**	**73.5**	**44.8**	**64.0**

Cell-level readings. This appendix also expands the aggregates of §6.2 at the level of the fourteen model×configuration cells of Table 1. DMPoT has lower perplexity than the uniform INT baseline in three of the fourteen cells (all three Qwen2.5 models at W4A4KV4). Its zero-shot accuracy is never more than 0.5%p below INT in any cell, and is above it in eight. Against FPoT, a comparison in which datapath, level budget and preprocessing protocol are all held fixed, DMPoT has the higher zero-shot accuracy in eleven of the fourteen cells alongside the lower perplexity in all fourteen. On Qwen2.5-1.5B W4A4KV4 it reads 10.95 against FPoT's 11.23.

Observations. Four patterns hold across all seven models and both configurations, each read off the tables above.

- **The grid is invisible task by task.** DMPoT lands within 3%p of uniform INT on *every* one of the 126 task-cells (fourteen model×configuration cells × nine tasks) and within 2%p on 120 of them. The largest single deviation is $+2.9$%p (Qwen2.5-14B ARC-Easy at W4A8KV8), in DMPoT's favor. The near-lossless aggregate of Table 1 is therefore not an average over compensating per-task swings.

Table 24: Per-task zero-shot accuracy for **Llama-2-13B** (conventions in Appendix I). LogART (†) is a post-training transplant on the uniform INT preprocessing (Appendix A.5).

Config	Method	ARC-C	ARC-E	Hella	Wino	PIQA	BoolQ	OBQA	Lam	SIQA	AVG
FP (bf16) reference		49.1	77.5	79.4	71.7	80.6	80.7	45.6	76.7	47.2	67.6
W4A4KV4	Uniform	46.9	74.6	77.0	69.8	78.7	78.5	44.5	74.6	45.8	65.6
	PoT	44.0	70.2	74.2	68.1	77.9	74.7	42.1	69.3	43.9	62.7
	APoT	46.4	73.7	77.0	70.1	79.0	76.9	43.7	75.1	45.6	65.3
	LogART†	31.7	53.7	55.2	59.2	66.9	64.9	35.0	42.3	39.7	49.9
	FPoT	46.8	73.6	76.3	69.5	78.4	78.0	43.9	75.2	45.4	65.2
	DMPoT (Ours)	**46.8**	**74.5**	**76.7**	**70.2**	**78.3**	**77.6**	**42.1**	**74.8**	**45.4**	**65.2**
W4A8KV8	Uniform	49.2	76.5	78.3	72.1	80.0	80.0	44.8	76.2	46.4	67.1
	PoT	47.7	74.3	76.7	70.2	78.9	77.8	43.3	72.3	45.6	65.2
	APoT	48.2	75.4	78.3	71.5	79.6	79.5	44.4	76.2	46.7	66.6
	LogART†	41.6	69.2	67.3	66.3	75.5	72.6	40.1	59.8	42.6	59.4
	FPoT	48.5	76.1	77.3	70.7	79.6	79.7	43.7	77.0	46.6	66.6
	DMPoT (Ours)	**48.8**	**76.2**	**78.2**	**71.8**	**79.6**	**79.9**	**45.2**	**76.4**	**46.3**	**66.9**

Table 25: Per-task zero-shot accuracy for **Qwen2.5-1.5B** (conventions in Appendix I). LogART (†) is a post-training transplant on the uniform INT preprocessing (Appendix A.5).

Config	Method	ARC-C	ARC-E	Hella	Wino	PIQA	BoolQ	OBQA	Lam	SIQA	AVG
FP (bf16) reference		44.9	71.6	67.8	63.5	76.0	72.8	40.4	61.9	49.0	60.9
W4A4KV4	Uniform	31.5	36.6	40.7	59.0	62.4	48.4	34.3	0.0	35.8	38.7
	PoT	31.2	34.9	38.2	55.6	61.4	44.7	34.1	0.0	34.9	37.2
	APoT	32.1	36.7	40.8	59.0	62.3	58.1	34.7	0.0	35.9	39.9
	LogART†	31.7	35.3	40.1	56.0	62.7	51.4	34.2	0.0	35.8	38.6
	FPoT	32.3	37.4	40.4	59.1	62.2	56.9	34.8	0.0	35.9	39.9
	DMPoT (Ours)	**32.1**	**37.2**	**41.3**	**59.0**	**62.4**	**51.0**	**35.1**	**0.0**	**36.1**	**39.3**
W4A8KV8	Uniform	33.4	38.9	43.9	63.0	64.6	67.2	36.5	0.0	37.1	42.7
	PoT	32.8	37.1	40.4	60.8	62.6	49.0	34.7	0.0	36.1	39.3
	APoT	33.6	39.6	43.6	62.7	64.3	68.8	35.7	0.1	37.2	42.8
	LogART†	33.4	38.7	43.4	62.6	64.1	60.3	36.1	0.1	37.0	41.7
	FPoT	33.5	39.3	43.2	62.5	63.9	66.1	36.0	0.0	37.1	42.4
	DMPoT (Ours)	**33.7**	**40.1**	**43.8**	**63.4**	**64.1**	**69.8**	**36.5**	**0.0**	**36.9**	**43.2**

- **PoT degrades broadly, not selectively.** Basic PoT is at or below the uniform baseline on all 63 task-cells at W4A4KV4 and all 63 at W4A8KV8, losing up to 10.2 and 18.3%p respectively. The coarse grid costs capability everywhere rather than on particular task types, which is what a resolution deficit (as opposed to a distributional artifact) predicts.
- **One task collapses for reasons unrelated to the grid.** LAMBADA collapses to at most 0.1% on Qwen2.5-1.5B (FP reference 61.9%) for *every* method including uniform INT, at both configurations. No other model shows it. It is a property of that model under this preprocessing pipeline, not of any weight grid, and it accounts for most of Qwen2.5-1.5B's outlying gap to FP.
- **Activation width, not weight grid, sets the distance to FP.** DMPoT's gap to the bf16 reference narrows at every model when activations and KV move from 4 to 8 bits ($3.1 \to 0.9$ on Llama-3-8B, $2.3 \to 0.6$ on Qwen2.5-7B, and to -0.8 on Qwen2.5-14B, which passes its FP reference). The same narrowing holds for the uniform INT baseline, so it measures the activation path rather than the weight grid.

Table 26: Per-task zero-shot accuracy for **Qwen2.5-7B** (conventions in Appendix I). LogART (†) is a post-training transplant on the uniform INT preprocessing (Appendix A.5).

Config	Method	ARC-C	ARC-E	Hella	Wino	PIQA	BoolQ	OBQA	Lam	SIQA	AVG
FP (bf16) reference		51.4	77.7	79.0	73.2	80.0	84.7	47.0	71.6	54.8	68.8
W4A4KV4	Uniform	50.6	76.4	76.2	69.7	78.5	82.0	44.2	68.0	51.9	66.4
	PoT	47.0	71.0	74.6	65.0	77.6	77.8	41.9	62.3	48.6	62.9
	APoT	49.6	75.0	76.4	68.5	78.2	82.2	44.5	70.0	51.1	66.2
	LogART†	46.9	72.4	75.6	66.4	78.0	80.0	43.5	64.1	50.3	64.1
	FPoT	50.5	77.3	75.4	68.1	78.3	82.6	44.3	68.7	51.8	66.3
	DMPoT (Ours)	**50.3**	**76.6**	**76.2**	**68.9**	**79.0**	**82.0**	**45.0**	**68.7**	**51.8**	**66.5**
W4A8KV8	Uniform	51.9	77.1	78.1	71.9	79.7	84.3	45.4	71.1	53.0	68.1
	PoT	48.4	73.6	76.9	69.3	79.3	80.9	43.9	65.5	51.7	65.5
	APoT	52.2	77.9	78.0	72.5	79.6	84.3	46.1	71.9	54.2	68.5
	LogART†	50.3	76.3	78.3	71.2	79.8	82.5	46.3	67.1	53.5	67.3
	FPoT	53.2	79.1	77.7	71.7	79.8	84.9	45.7	71.6	53.6	68.6
	DMPoT (Ours)	**51.5**	**77.4**	**78.1**	**71.7**	**79.7**	**85.1**	**44.9**	**71.5**	**53.6**	**68.2**

Table 27: Per-task zero-shot accuracy for **Qwen2.5-14B** (conventions in Appendix I). LogART (†) is a post-training transplant on the uniform INT preprocessing (Appendix A.5).

Config	Method	ARC-C	ARC-E	Hella	Wino	PIQA	BoolQ	OBQA	Lam	SIQA	AVG
FP (bf16) reference		59.0	79.3	82.9	74.7	82.1	85.4	44.8	74.1	55.5	70.9
W4A4KV4	Uniform	55.0	80.3	80.0	72.0	79.9	84.1	44.1	71.5	51.0	68.7
	PoT	52.4	76.6	77.3	67.5	78.0	73.9	42.5	66.8	48.1	64.8
	APoT	55.5	80.9	79.9	73.5	79.7	84.2	43.3	73.0	51.2	69.0
	LogART†	51.6	78.4	77.7	68.5	78.3	79.6	43.3	65.6	49.3	65.8
	FPoT	55.3	80.7	79.6	72.5	80.0	84.5	43.7	73.0	51.1	68.9
	DMPoT (Ours)	**57.7**	**82.0**	**80.1**	**73.2**	**80.2**	**83.4**	**44.6**	**72.8**	**50.5**	**69.4**
W4A8KV8	Uniform	58.9	81.5	82.1	75.9	81.8	85.8	45.1	74.9	54.0	71.1
	PoT	55.6	79.7	80.5	73.3	81.0	83.1	44.7	69.4	51.4	68.7
	APoT	59.5	83.9	82.1	76.5	82.1	86.4	44.7	75.1	54.2	71.6
	LogART†	54.8	79.3	81.2	74.2	80.9	82.4	44.0	67.8	50.2	68.3
	FPoT	59.1	82.6	81.9	76.8	82.1	86.9	44.7	73.9	53.4	71.3
	DMPoT (Ours)	**59.9**	**84.4**	**82.0**	**77.5**	**81.8**	**85.5**	**45.1**	**75.5**	**53.4**	**71.7**

Table 28: Per-task zero-shot accuracy for **Mistral-7B** (conventions in Appendix I). LogART (†) is a post-training transplant on the uniform INT preprocessing (Appendix A.5).

Config	Method	ARC-C	ARC-E	Hella	Wino	PIQA	BoolQ	OBQA	Lam	SIQA	AVG
FP (bf16) reference		52.2	78.3	80.5	73.6	82.3	82.1	44.2	75.1	45.8	68.2
W4A4KV4	Uniform	49.7	76.0	78.7	70.2	80.3	81.6	43.2	73.9	45.3	66.5
	PoT	47.7	74.1	77.6	68.2	80.0	78.0	41.9	68.8	44.8	64.6
	APoT	49.7	76.8	78.8	70.1	80.9	81.9	42.5	74.3	45.2	66.7
	LogART†	46.1	72.6	77.0	69.2	78.8	76.4	40.3	69.5	43.7	63.7
	FPoT	49.4	76.0	78.2	69.9	80.3	80.0	42.0	73.5	45.2	66.1
	DMPoT (Ours)	**49.7**	**76.9**	**78.7**	**70.9**	**80.5**	**81.7**	**43.8**	**74.5**	**45.1**	**66.9**
W4A8KV8	Uniform	51.0	77.7	79.8	73.5	81.6	81.7	43.7	75.5	45.9	67.8
	PoT	48.9	75.0	78.6	71.2	80.5	77.0	41.9	70.9	45.3	65.5
	APoT	51.1	77.6	79.8	72.7	81.5	82.5	43.5	75.8	45.8	67.8
	LogART†	49.3	75.5	78.7	71.3	80.7	79.5	43.8	72.0	44.3	66.1
	FPoT	50.5	77.9	79.2	73.2	81.0	80.7	42.7	75.0	45.7	67.3
	DMPoT (Ours)	**51.4**	**77.8**	**79.7**	**72.8**	**81.5**	**81.8**	**43.7**	**75.3**	**45.5**	**67.7**

J Cross-Framework Portability

This appendix supports the framework-agnostic claim of §6. DMPoT is framework-agnostic with respect to the preprocessing it rides on. Nothing in the grid, the calibration, or the folded datapath is specific to the OSTQuant pipeline of Table 1. Table 29 tests this directly by integrating the weight grids into DuQuant (Lin et al., 2024a) in place of its round-to-nearest weight quantizer, on three of the seven models (Llama-3-8B, Llama-2-7B, Llama-2-13B).

Table 29: Cross-framework comparison of the weight grid (activations and KV cache uniform INT throughout) in **OSTQuant** (the pipeline of Table 1) and **DuQuant** (settings in Appendix J), three-seed mean±sample std. PPL: WikiText-2 (↓). 0-shot[9]: nine-task mean (↑, %).

			W4A4KV4		W4A8KV8	
Model	Framework	Weight	PPL↓	0-shot[9]↑	PPL↓	0-shot[9]↑
Llama-3-8B	OSTQuant	Uniform	$7.358_{\pm.074}$	$65.2_{\pm.4}$	$6.601_{\pm.011}$	$67.5_{\pm.1}$
		FPoT	$7.596_{\pm.020}$	$64.8_{\pm.6}$	$6.849_{\pm.007}$	$67.3_{\pm.5}$
		DMPoT (Ours)	$\mathbf{7.417}_{\pm.041}$	$\mathbf{65.1}_{\pm.2}$	$\mathbf{6.672}_{\pm.013}$	$\mathbf{67.3}_{\pm.3}$
	DuQuant	Uniform	$7.663_{\pm.005}$	$63.0_{\pm.3}$	$6.557_{\pm.005}$	$67.0_{\pm.3}$
		FPoT	$7.850_{\pm.021}$	$62.8_{\pm.3}$	$6.695_{\pm.013}$	$66.8_{\pm.2}$
		DMPoT (Ours)	$\mathbf{7.648}_{\pm.019}$	$\mathbf{63.6}_{\pm.3}$	$\mathbf{6.560}_{\pm.008}$	$\mathbf{67.5}_{\pm.2}$
Llama-2-7B	OSTQuant	Uniform	$5.971_{\pm.009}$	$62.1_{\pm.3}$	$5.630_{\pm.035}$	$63.8_{\pm.2}$
		FPoT	$6.114_{\pm.042}$	$61.9_{\pm.2}$	$5.751_{\pm.019}$	$63.7_{\pm.2}$
		DMPoT (Ours)	$\mathbf{6.042}_{\pm.013}$	$\mathbf{62.1}_{\pm.2}$	$\mathbf{5.677}_{\pm.012}$	$\mathbf{64.0}_{\pm.4}$
	DuQuant	Uniform	$6.112_{\pm.033}$	$61.6_{\pm.1}$	$5.621_{\pm.001}$	$64.1_{\pm.2}$
		FPoT	$6.185_{\pm.051}$	$61.3_{\pm.3}$	$5.671_{\pm.006}$	$63.9_{\pm.5}$
		DMPoT (Ours)	$\mathbf{6.109}_{\pm.029}$	$\mathbf{62.0}_{\pm.0}$	$\mathbf{5.640}_{\pm.004}$	$\mathbf{64.2}_{\pm.1}$
Llama-2-13B	OSTQuant	Uniform	$5.260_{\pm.007}$	$65.6_{\pm.2}$	$5.032_{\pm.005}$	$67.1_{\pm.3}$
		FPoT	$5.366_{\pm.017}$	$65.2_{\pm.1}$	$5.106_{\pm.017}$	$66.6_{\pm.1}$
		DMPoT (Ours)	$\mathbf{5.305}_{\pm.016}$	$\mathbf{65.2}_{\pm.2}$	$\mathbf{5.065}_{\pm.023}$	$\mathbf{66.9}_{\pm.3}$
	DuQuant	Uniform	$5.302_{\pm.001}$	$65.6_{\pm.2}$	$4.998_{\pm.003}$	$66.8_{\pm.1}$
		FPoT	$5.348_{\pm.004}$	$65.2_{\pm.5}$	$5.044_{\pm.005}$	$67.0_{\pm.2}$
		DMPoT (Ours)	$\mathbf{5.308}_{\pm.005}$	$\mathbf{65.4}_{\pm.3}$	$\mathbf{5.003}_{\pm.003}$	$\mathbf{67.0}_{\pm.3}$

The column-wise weight calibration is GPTQ (Frantar et al., 2023), applied to the DuQuant-transformed layers with inputs taken from the already-quantized preceding layers: 128 WikiText-2 training sequences of 2,048 tokens, block size 128, dampening 0.01, no activation ordering, and per-channel symmetric weight scales from an MSE shrink search, without grid alignment. All three grids share this calibrator, each with its own rounding rule. DuQuant's transform uses block size 128, 256 rotation steps, one permutation, smoothing α=0.6 and an activation clipping ratio of 0.9, without its learnable weight clipping. The calibration set is the same for every seed, so the three seeds (2/0/123, where 2 is DuQuant's own default and the other two match ours) vary DuQuant's random rotation completion only.

Table 29 reads the same way in both frameworks. Under DuQuant, DMPoT stays within 0.02 PPL of uniform INT in all six cells (below it in two) and below FPoT in all six, by 0.03–0.20 PPL. Zero-shot accuracy is within 0.6%p of uniform INT throughout. The offset between the two frameworks moves the uniform INT row more than the DMPoT row (on Llama-3-8B W4A4KV4, 7.358 → 7.663 against 7.417 → 7.648), so it is a property of the preprocessing, not of the grid.

K Generation Tasks

This appendix supports the generation-quality pointer of §6.4.

Perplexity and multiple-choice accuracy both score a model on a single forward pass over text it is given. Generation quality does not reduce to either. Errors compound across autoregressively sampled tokens, so a change to the weight grid that is invisible at the perplexity level can still move exact-match, functional-correctness, or long-context retrieval scores, and can move them asymmetrically

Table 30: Generation tasks on two instruction-tuned models (W4A4KV4, mean±sample std over three calibration seeds 42/0/123). Five axes: math (GSM8K), code (instruct-mode HumanEval pass@1), instruction following (IFEval, prompt-level strict), safety (AdvBench attack-success rate ↓, judged by Llama-Guard-3-8B), and long context (LongBench-E `passage_retrieval_en`, PR). Settings, per-seed LongBench subsets and the LogART caveat are in Appendix K.

Model	Method	GSM8K	HumanEval	IFEval	AdvBench-ASR↓	LongBench-PR
Qwen2.5-7B-Inst.	FP (bf16)	81.05	82.93	69.50	0.00	87.50
	Uniform	74.15±0.7	73.78±3.8	64.14±1.1	0.58±0.6	89.17±2.9
	PoT	53.95±4.4	58.94±2.5	63.65±1.9	0.83±1.0	84.17±2.9
	APoT	76.57±0.5	75.61±3.0	68.15±1.4	0.26±0.3	86.67±1.4
	LogART	70.53±0.8	69.72±1.5	66.30±0.7	0.51±0.3	86.67±1.4
	FPoT	71.65±3.1	70.33±0.7	66.91±1.0	0.32±0.1	84.17±3.8
	DMPoT (Ours)	**76.22**±0.1	**73.58**±3.4	**67.34**±1.6	**0.51**±0.3	**87.50**±2.5
Llama-3.1-8B-Inst.	FP (bf16)	78.85	66.46	72.27	3.85	82.50
	Uniform	60.78±0.7	52.44±1.8	68.21±3.0	3.72±1.2	79.58±4.7
	PoT	44.58±1.6	27.64±1.5	57.48±2.2	9.55±3.4	61.77±16.5
	APoT	61.10±0.5	51.63±1.5	68.52±1.9	2.63±1.1	82.08±3.1
	LogART	23.58±9.1	3.46±1.8	37.21±8.1	27.76±18.9	5.14±3.1
	FPoT	57.67±1.9	47.76±2.0	66.73±2.2	6.03±1.1	79.44±5.9
	DMPoT (Ours)	**62.19**±0.5	**50.20**±2.5	**67.10**±2.0	**3.46**±1.3	**80.00**±5.0

across axes. Table 30 therefore evaluates five generation-side axes (math, code, instruction following, safety, and long context) on two instruction-tuned models at W4A4KV4.

All generation is greedy, and the seed selects the calibrated checkpoint. For LongBench it also draws the examples scored. GSM8K (all 1,319 problems, 5-shot, flexible-extract match) and IFEval (all 541 prompts, 0-shot, chat template) run in lm-eval-harness with generation capped at 256 and 1,280 tokens. HumanEval runs in instruct mode on all 164 problems (chat prompt, then markdown code extraction, then unit tests), with at most 640 new tokens and one sample per problem. AdvBench runs all 520 prompts under the chat template with 64 new tokens, and Llama-Guard-3-8B labels each response. LongBench scores 40 examples of the LongBench-E `passage_retrieval_en` split drawn per seed, so its dispersion includes subset as well as checkpoint variation, with the official prompt templates, middle truncation to `max_len` = 10000 on Qwen and 8000 on Llama, whose RoPE scaling our loader disables, capping evaluation at its original 8,192-token window, and 32 new tokens. The FP (bf16) reference is scored on the seed-42 draw. Settings not listed follow each model's released generation configuration. Every row is measured. LogART alone is measured outside its published operating point (§6.2). It is an untuned transplant and is seed-unstable, most of all in the safety column, so those cells bound the transplant rather than its published setting. The finding is that generation quality follows perplexity, with one exception worth stating first. On Llama-3.1-8B-Instruct, DMPoT loses 2.2%p of HumanEval pass@1 against uniform INT ($50.20_{\pm 2.5}$ vs $52.44_{\pm 1.8}$), a gap of about one standard deviation on that axis. Elsewhere the grid is within one standard deviation of INT or above it. GSM8K is *above* uniform INT on both models (76.22 vs 74.15 and 62.19 vs 60.78), attack-success rate is below it on both (0.51 vs 0.58 and 3.46 vs 3.72), and IFEval and LongBench move by at most 3.2%p and in both directions. Against the log-uniform member the ordering matches the perplexity tables. DMPoT is ahead of FPoT on nine of the ten measured cells, by 4.6 and 4.5 points on GSM8K. Against APoT the ordering is not the perplexity tie of Table 1. APoT is ahead on eight of the ten cells, by at most 2.1 points (Llama-3.1 LongBench retrieval 82.08 vs 80.00, Qwen HumanEval 75.61 vs 73.58), and DMPoT is ahead on Llama-3.1 GSM8K and Qwen retrieval. On this suite the comparison with APoT is an accuracy–area trade rather than a tie. No axis moves by more than 3.2%p between uniform INT and DMPoT, whereas DMPoT's distance from the FP (bf16) reference reaches 16.7 points (Llama-3.1 GSM8K) and 9.4 (Qwen HumanEval), and uniform INT's is of the same size.

L EXPERIMENTAL SETTINGS

This appendix gives the settings behind Table 1 and the bit-width sweep of Appendix F. The activation and KV-cache quantizer is Eq. (5), the transform structure is in Appendix A.2, the grid alignment in Appendix A.4 and the baseline protocols in Appendix A.5. The DuQuant runs state their own settings in Appendix J. The checkpoints of the generation tasks are prepared with the pipeline below, and Appendix K gives their decoding settings.

Table 31: The seven base checkpoints of Table 1 (Hugging Face identifiers and revisions) and their dimensions. L is the number of decoder blocks, d the hidden size, and H/H_{kv} the numbers of query and key-value heads (head dimension 128 throughout, the size of the online query and key Hadamard transform). d_{ff} is the FFN width, the size of the online Hadamard transform before each down-projection, which Qwen2.5-7B zero-pads from 18,944 to 19,968.

Model	Checkpoint	Revision	L	d	H/H_{kv}	d_{ff}
Llama-3-8B	`meta-llama/Meta-Llama-3-8B`	`8cde5ca8`	32	4,096	32/8	14,336
Llama-2-7B	`meta-llama/Llama-2-7b-hf`	`01c7f73d`	32	4,096	32/32	11,008
Llama-2-13B	`meta-llama/Llama-2-13b-hf`	`5c31dfb6`	40	5,120	40/40	13,824
Qwen2.5-1.5B	`Qwen/Qwen2.5-1.5B`	`8faed761`	28	1,536	12/2	8,960
Qwen2.5-7B	`Qwen/Qwen2.5-7B`	`d1497293`	28	3,584	28/4	18,944
Qwen2.5-14B	`Qwen/Qwen2.5-14B`	`97e1e763`	48	5,120	40/8	13,824
Mistral-7B	`mistralai/Mistral-7B-v0.3`	`caa1feb0`	32	4,096	32/8	14,336

Models. The seven models of Table 1 are pretrained base checkpoints (Table 31), loaded in BF16 at a sequence length of 2,048 tokens for training, calibration and perplexity. The generation tasks use Llama-3.1-8B-Instruct (`meta-llama/Llama-3.1-8B-Instruct`, revision `0e9e39f2`) and Qwen2.5-7B-Instruct (`Qwen/Qwen2.5-7B-Instruct`, revision `a09a3545`), whose dimensions are those of Llama-3-8B and Qwen2.5-7B in Table 31. The measured density (Appendix C), the per-projection sweep (Appendix D), the bit-width sweep and the ablations (Appendix H) use Llama-3-8B.

Quantization. The weights of the seven linear projections in every decoder block are quantized on a level set symmetric about zero, with one scale per output channel. The embeddings, the LM head, the normalization layers and the query, key and value biases of Qwen2.5 stay in BF16. Activations and the KV cache follow Eq. (5) at six points per block: the two normalization outputs, the down-projection input, the attention output, the keys after RoPE and the online Hadamard transform, and the values. The two normalization outputs are clipped at $\gamma{=}0.9$ with four-bit activations and at $\gamma{=}1$ with eight-bit activations. Queries, attention scores and attention probabilities are not quantized.

Transform training. Following OSTQuant, a global rotation of the residual stream and one rotation per layer and key-value head on the value–output path are learned on the Stiefel manifold, together with per-channel smoothing scales at the two normalization outputs, the query–key and value–output pairs, and the down-projection input. Each rotation starts from the eigenvectors of its weight covariance multiplied by a random-sign Hadamard matrix. Training runs 400 steps of Riemannian gradient descent with Adam updates at batch size 4 on 2,048-token sequences of the WikiText-2 training split, with learning rates $1.69{\times}10^{-2}$ for the rotations and $1.79{\times}10^{-3}$ for the smoothing scales, linear decay to zero without warmup, and gradient-norm clipping at 1.0. The loss is $\mathrm{KL}(p_{\mathrm{BF16}} \,\|\, p_{\mathrm{quant}})$ over the BF16 model's top-1,000 tokens, with weight, activation and KV fake quantization active.

Weight calibration. The trained transforms are fused into the weights, and GPTQ then quantizes the decoder blocks one after another on 128 windows of 2,048 tokens from the WikiText-2 training split, taking its inputs from the already-quantized preceding blocks, with dampening $1.0{\times}$ the mean Hessian diagonal, column blocks of 128 and no activation ordering. Before GPTQ, the per-channel scale is fixed by a clip search over 80 shrink factors from 1.00 down to 0.21 on the grid's deployed levels (Algorithm 1), followed for DMPoT-4 and FPoT by the grid alignment of Appendix A.4. The chain members of the sweep ($b{\geq}5$) keep the searched scale.

Seeds. Each of the three calibration seeds (42, 0, 123) sets the random signs of the rotation initialization, the order of the training sequences and the 128 calibration windows. The training corpus, the evaluation data and the harness settings do not depend on it.

Evaluation. Perplexity is measured on the WikiText-2 raw test split, joined into one token stream and cut into non-overlapping 2,048-token windows. The nine zero-shot tasks are ARC-Challenge and ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), PIQA (Bisk et al., 2020), BoolQ (Clark et al., 2019), OpenBookQA (Mihaylov et al., 2018), LAMBADA (Paperno et al., 2016) and Social IQA (Sap et al., 2019). They run in lm-eval-harness (Gao et al., 2024), version 0.4.8, with the task defaults (LAMBADA is the OpenAI variant) and automatic batch size. Appendix I gives the metric of each task.

Software and compute. The runs use Python 3.9, PyTorch 2.2.1 with CUDA 12.1, Transformers 4.44.2 and Triton 2.2.0 (Tillet et al., 2019), each on a single NVIDIA A100-80GB GPU. Transform training of Llama-2-13B and Qwen2.5-14B fits on it with gradient checkpointing. For PoT, FPoT and DMPoT-4, a fused Triton kernel merges the quantize–dequantize–STE pipeline into one launch and shortens the training loop (Appendix G). A complete DMPoT run (transform training, GPTQ, perplexity and the nine tasks) takes 1.7 h on Llama-3-8B and between 0.6 h (Qwen2.5-1.5B) and 3.6 h (Llama-2-13B) across the seven models.