QUANTIZING RECURSIVE REASONING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive reasoning models solve hard puzzles by applying compact, weight-tied blocks over many refinement steps. Because these blocks are reused many times, quantizing them creates a severe error accumulation challenge: the quantization error is incurred at every step. While 8-bit quantization (integer or float) preserves accuracy, moving to a per-tensor 4-bit format causes a systematic bias to accumulate. The ensuing drift catastrophically degrades exact-solution accuracy on Sudoku from 84.1% to 0.0% (only $\sim 25\%$ of cells correct). In this work, we show that this collapse is caused by activation-scaling *granularity* rather than bit-width or number format. Crucially, moving to per-block scaling restores stable contraction and most of the lost accuracy. To implement this, we apply MXINT4, a blockwise integer format for weights and activations, to recursive reasoning models. It is competitive with blockwise float formats on our tasks, while keeping integer elements and power-of-two block scales. Finally, recursion depth and reuse modulate quantization sensitivity, with the deepest trained model we test (EqR) the most sensitive. Yet blockwise scaling overcomes this vulnerability, staying robust across the evaluated recursive models and transferring to the open-ended ARC-AGI benchmark.

1 INTRODUCTION

A recent line of research has shown that hard reasoning problems can be solved by recursively running models orders of magnitude smaller than those required to reach similar accuracy with non-recurrent inference. The Hierarchical Reasoning Model (HRM) (Wang et al., 2025) and the Tiny Recursive Model (TRM) (Jolicoeur-Martineau, 2025), with only 27M and 7M parameters, respectively, reach competitive accuracy on Sudoku, maze solving, and ARC-AGI by applying compact, weight-tied transition functions over many refinement steps. For example, TRM reaches 44.6% on ARC-AGI-1, whereas the 671B-parameter DeepSeek R1 reaches 15.8% (Jolicoeur-Martineau, 2025). These models are attractive for high-quality inference on heavily memory constrained platforms because their parameter count is very small. Yet their behavior under quantization remains essentially unstudied: inference quality results are reported almost entirely in full precision, without an accompanying low-precision accuracy or trajectory analysis.

To understand quantization behavior, we view a recursive reasoner as a learned state-space computation. Given an input x, a shared transition operator repeatedly refines a latent reasoning state and an answer state. Quantization fundamentally replaces this operator with a perturbed or noisy transition.

In a standard non-recurrent model, a quantizer perturbs a finite sequence of *different* layers. In a weight-tied recursion, however, the *same* quantized transition is reused many times. This makes the induced trajectory just as important as the one-step error. While the recursion might safely absorb an approximately zero-mean perturbation, a systematic bias behaves differently. It can push the trajectory, or the fixed point it converges to, in a coherent direction at every step, so drift accumulates.

This paper studies quantization at the level of these reused transition operators. We examine when a low-bit transition stays faithful under repeated application, and how to build one that does. We make three primary contributions:

1. Quantization as transition perturbation. We compare a full-precision transition with its quantized counterpart applied to the same reasoning state (Section 3). Empirically, 8-bit integer precision acts as a small perturbation. The quantized trajectory contracts back toward the full-

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.

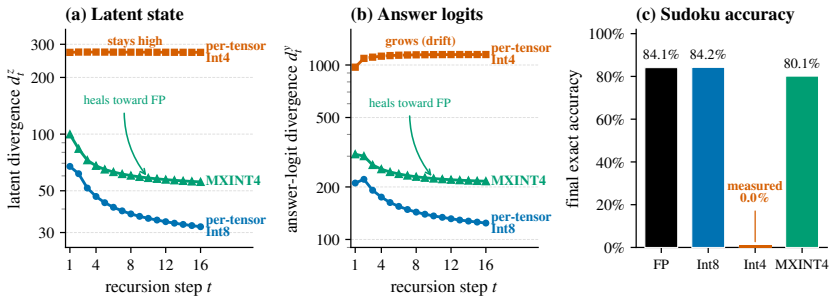


Figure 1: Transition contraction versus drift (Sudoku). (a,b) Across 16 recursion steps, per-tensor Int4 maintains high latent divergence and growing logit divergence; Int8 and per-block MXINT4 contract toward the full-precision path. **(c)** Int4 reduces exact accuracy to 0.0%, whereas Int8 and MXINT4 (80.1%) retain most FP accuracy. The shallower Maze recursion is less sensitive (Int4 77.7% vs. FP 84.7%).

precision path and accuracy is preserved. Conversely, per-tensor 4-bit quantization induces severe trajectory drift, dropping Sudoku exact accuracy from 84.1% to 0.0%. The displacement concentrates in the most-reused recursive layers ($336\times$ per forward pass) and scales directly with reuse depth.

2. A blockwise integer transition rule. We show that activation scaling *granularity* is the dominant control variable (Section 4). Weight-only 4-bit quantization costs only a few points. However, when quantizing activations at 4-bit, both per-tensor integer (Int4, 0.0%) and float (FP4, 0.6%) formats lose exact accuracy. In contrast, per-block scaling restores stable transition behavior and most of the lost accuracy. We apply this principle using MXINT4, a blockwise weight-and-activation format with integer mantissas, which reaches 80.1% (Sudoku) and 84.7% (Maze), competitive with blockwise float formats on these tasks while keeping integer elements and power-of-two block scales. Tested outlier controls (SmoothQuant, Hadamard rotation) fail to provide this same post-training stability.

3. Recursion depth and reuse modulate quantizability. We demonstrate this across TRM and the Equilibrium Reasoner (EqR), with additional stochasticity controls (Section 5). The most sensitive is EqR, which iterates toward a fixed-point attractor over $\sim$288 layer applications. Per-tensor Int4 prevents attractor convergence, and stochastic breadth cannot average the systematic error away. Yet, per-group activation scaling restores stability and preserves depth scaling ($0.01\% \rightarrow 82.3\%$). Finally, this same per-block rule successfully transfers to the open-ended ARC-AGI benchmark. We also show that quantization-aware training (QAT) offers a complementary weight-side route towards even better accuracy (Sections 4.5 and 4.6).

2 Background and Related Work

Recursive and looped reasoning models. Recent models solve hard reasoning tasks with small, repeatedly applied networks. HRM couples two recurrent modules (Wang et al., 2025), while TRM reduces this to a $\sim$7M-parameter block (Jolicoeur-Martineau, 2025). They build on weight-tied and looped transformers (Dehghani et al., 2019; Giannou et al., 2023; Xu & Sato, 2025), adaptive computation (Graves, 2016), and deep-equilibrium models (Bai et al., 2019). The family now includes stochastic, autoregressive, hybrid, long-horizon, and language-model variants (Sghaier et al., 2026; Baek et al., 2026; Rauba et al., 2026; Wang & Reid, 2026; Yang et al., 2026; Wang et al., 2026), but these studies report full-precision behavior. We study the missing interaction between quantization and recursion structure.

Quantizing weight-tied and recurrent networks. Recurrent quantization is more brittle than feedforward quantization (Ott et al., 2016; Hubara et al., 2018), motivating RNN-specific quantizers and 4-bit LSTM pipelines (He et al., 2016; Alom et al., 2018; Fasoli et al., 2021). Prior methods manage repeated error or accumulator growth (Park et al., 2018; Colbert et al., 2023b;a), and work on weight sharing and equilibria provides theoretical context (Lan et al., 2020; Bai et al., 2019; Li et al., 2026a). We instead measure contraction versus drift in modern recursive reasoners and its dose–response to reuse depth.

Iterated models: diffusion and looped-model quantization as the closest related work. Quantized diffusion is the closest analogue: post-training quantization (PTQ) perturbs an iterated shared-weight transition and its error compounds across steps (Shang et al., 2023; Li et al., 2023; He et al., 2023). Diffusion, however, conditions on timestep t, enabling t-dependent quantizers (So et al., 2023; Huang et al., 2024); our reasoners reuse one step-independent operator. Moreover, small perceptual deviations can be tolerated in images, whereas one displaced symbol can invalidate an exact logical solution. We expand this comparison and its connection to exposure bias in Section A.8. The most direct concurrent work, LoopQ (Fang et al., 2026), quantizes looped language models with loop-aware adapters, trajectory-aware calibration, and group-wise integer scaling, and similarly bounds cross-loop error amplification. We study recursive reasoners on exact-solution tasks and identify activation-scaling granularity as the variable that controls stability.

Quantizing reasoning models, and low-bitwidth number formats. At lower bit-widths, reasoning accuracy becomes increasingly fragile (Liu et al., 2025a; Zhao et al., 2025); an early quantized error can cascade through a chain of thought (Li et al., 2025; 2026b). Existing mitigations use QAT, lookahead losses, or selective weight protection (Lv et al., 2026; Zhang et al., 2026b; Choi et al., 2026). Our setting repeatedly applies the same faulty block.

Microscaling (MX) instead gives small blocks a shared exponent (Rouhani et al., 2023; Open Compute Project, 2023). Four-bit MXFP remains challenging in LLMs (Zhang et al., 2026a; Fasoli et al., 2026), often requiring rotations or specialized training (Ashkboos et al., 2024; Liu et al., 2025b; Tseng et al., 2025; Chmiel et al., 2025). The integer–float crossover is regime-dependent (Chen et al., 2025; Zhang et al., 2023); here, per-block integer is competitive with per-block float on small weight-tied reasoners. Unlike BitSkip's layerwise early exit (Bhuvaneswaran & Liu, 2025), we study error accumulated across reuses of one block.

3 Quantization Bias in Reused Transition Operators

3.1 Recursive Transitions and Quantized Trajectories

A weight-tied recursive reasoner repeatedly applies a single learned operator to refine a latent state. Quantization replaces this operator with a perturbed version at every reuse. This raises a key question: does the quantized trajectory stay close to the full-precision path, or does a small one-step perturbation compound into a coherent trajectory-level drift? We study the TRM (Jolicoeur-Martineau, 2025), which refines a latent state z and answer embedding y by applying a single two-layer, $\sim$7M-parameter block f_θ. This block is applied over T deep-supervision steps consisting of several inner cycles. In our Sudoku configuration the inner "L-level" layers are applied 336 times per forward pass. In contrast, the readout heads are applied only 16 times. We evaluate on *Sudoku-Extreme* (hard 9×9 puzzles as length-81 token sequences) and *Maze-Hard* (30×30 start-to-goal path-finding) (Wang et al., 2025; Jolicoeur-Martineau, 2025), scored by *exact accuracy*: an instance counts only if every cell or path-step is correct (Section A.13).

Writing the recursive state $s_t = (z_t, y_t)$ and the full-precision recursion $s^{\mathrm{fp}}_{t+1} = F_\theta(s^{\mathrm{fp}}_t, x)$, the quantized recursion is

$$s^q_{t+1} = F_\theta(s^q_t, x) + \epsilon_Q(s^q_t, x). \tag{1}$$

Here, ϵ_Q is the induced input–output error of the *entire* quantized transition, including all internal weight and activation quantizers. We do not assume this error is simply additive output-level noise. What matters is whether its mean component aligns across steps. At each step we therefore track task accuracy alongside two trajectory divergences:

$$d^z_t = \|h^q_t - h^{\mathrm{fp}}_t\|, \qquad d^y_t = \|\ell^q_t - \ell^{\mathrm{fp}}_t\|. \tag{2}$$

Here h_t is the answer-region hidden state (z_H in the TRM code, i.e. the answer state y) and ℓ_t its logits. The shape of these divergence curves distinguishes a stable perturbation (where the divergence contracts) from a biased one (where displacement is reintroduced at every step). Unless noted otherwise, reported numbers are final-step point estimates. The per-tensor PTQ rows (Table 1) and the QAT/weight-learned step-size quantization (LSQ) comparison (Table 2) report mean$\pm$std (population std) over three seeds. Per-block (MX) rows are deterministic given the checkpoint. Per-step trajectory diagnostics quantize only the linear layers (including the output and halting heads), so their endpoints differ slightly from the broader-coverage ladder rows of Table 1 (Section A.13).

3.2 LOW-BIAS PERTURBATIONS CONTRACT; BIASED 4-BIT PERTURBATIONS DRIFT

Figure 1 presents our central trajectory observation. At 8-bit precision (Int8), the quantized transition behaves as a low-bias perturbation. The divergence is largest at the first step but *shrinks* monotonically as the recursion proceeds. Consequently, final accuracy matches full precision (Sudoku Int8 $= 84.2\%$ vs. FP $= 84.1\%$ in Figure 1; Maze Int8 $= 83.6\%$ vs. FP $= 84.7\%$ in Table 1). Each application of the transition contracts the perturbed state back toward the full-precision trajectory faster than the quantizer displaces it.

At per-tensor 4-bit the sign of this effect completely flips. The trajectory divergence now *grows* with depth. On Sudoku, the answer-logit divergence climbs from $\sim$970 at the first step to $\sim$1150 by step 16. The latent divergence saturates at a large value $\sim$270 and fails to decay. As a result, per-tensor Int4 PTQ drives exact accuracy from 84.1% down to 0.0%. The same operator that damped 8-bit error now compounds 4-bit error: the quantized transition ceases to be a stable approximation and instead induces a coherent drift. On the easier Maze task (which converges in roughly three steps), the effect is milder. Int4 reaches 77.7% versus 84.7% at full precision, a $\sim$7-point drop. The fundamental asymmetry remains the same: 8-bit divergence decays while per-tensor 4-bit does not. Previewing Section 4, the per-block 4-bit fix (MXINT4 in Figure 1) contracts like 8-bit, so scaling *granularity* rather than bit-width sets the regime.

This sharp, controllable contraction-to-drift transition extends prior evidence of recurrent brittleness and accumulated per-step error (Ott et al., 2016; Hubara et al., 2018; Park et al., 2018; Arora et al., 2022).

3.3 TRANSITION BIAS VERSUS ZERO-MEAN NOISE

Viewing quantization as a transition perturbation suggests a simple failure mechanism. A quantizer whose grid is poorly matched to the activation distribution produces an expected rounding error $\mathbb{E}[\mathcal{Q}(a) - a] \neq 0$. In a feedforward network these biases occur across different, independent layers and need not align. In a weight-tied recursion the same transition is reused, so the mean component of ϵ_Q is injected in a similar direction at every application and the resulting displacement scales with the number of reuses. This behaves like a biased perturbation, unlike zero-mean noise.

We confirm the localization directly by calibrating an Int4 model and measuring each layer's relative output error (Figure 12). The error *magnitude* is dominated by the recursive down-projection layers (relative error $\sim$1.1–1.65), exactly the layers re-applied hundreds of times per forward pass: the largest transition perturbations sit precisely where the recursion reuses them most. While Figure 12 shows magnitude, the perturbation's *systematic* nature is established behaviorally in Section 5.3 and measured directly in Section 4.3: averaging independent trajectories and adding stochastic breadth fail to lift Int4 accuracy above zero, the signature of a coherent bias (consistent with the nonzero signed per-layer means found during calibration).

This distinguishes our setting from diffusion-model quantization, its closest related work among iterated generative models. A diffusion denoising step *conditions on the timestep* t. This means accumulation can be damped with t-aware quantizers and calibration (He et al., 2023; So et al., 2023; Huang et al., 2024). A weight-tied TRM has no analogous t: it re-applies one operator with a single set of quantization parameters, which makes the accumulation more direct to analyze but larger in magnitude (extended comparison in Section A.9).

3.4 A DYNAMICAL-SYSTEMS PERSPECTIVE

The contraction-versus-drift dichotomy admits a lightweight, conditional model (Proposition 1, a qualitative displacement scaffold rather than a tight bound, with the full statement in Section A.11). We separate the per-step perturbation ϵ_Q of Equation (1) into its magnitude (bounded by ε) and its trajectory mean $\boldsymbol{\mu}_Q(x)$, without assuming the piecewise-constant quantizer is continuous. Motivated by the contracting Int8 divergence (Figure 1), we model the full-precision transition as locally contractive with modulus $L < 1$. Two conclusions follow:

- **Displacement:** if the ball of radius $\varepsilon/(1 - L)$ around the FP fixed point lies within the contractive region, quantized iterates enter and stay in it. The contracting 8-bit curves are consistent with this condition, and the drifting per-tensor 4-bit curves indicate that it fails.

- **Coherent accumulation:** magnitude alone does not distinguish bias from a same-size zero-mean fluctuation. Under a fixed-Jacobian linearization ($\|J\| \leq L$), a near-constant per-step error $\boldsymbol{\mu}_Q(x)$ (consistent with Section 4.3) drives the error to $(I - J)^{-1}\boldsymbol{\mu}_Q(x)$, whereas an independent zero-mean perturbation yields zero displacement in expectation.

This bias-versus-noise separation explains why injecting zero-mean per-step noise fails to rescue Int4 (Section 5.3) and why per-block scaling succeeds: it reduces both ε and $\|\boldsymbol{\mu}_Q(x)\|$, and MXINT4 empirically returns to contracting trajectories. The account is conditional. We do not measure the contractive region, so the regime split is inferred from observed drift rather than verified (Section A.11). Two controls support it (Section A.5): a dose–response sweep where the FP–Int4 gap widens with reuse, and a dimension-matched untied control ($7\times$ less reuse) that roughly doubles Int4 cell-accuracy (54% vs. 25%) yet leaves exact accuracy near-zero (1.6%). Weight reuse amplifies the drift, but the root error is the activation-quantization problem of Section 4.

The next section constructs a transition rule that explicitly reduces this bias.

4 Blockwise integer quantization of recursive transitions

We seek a low-bit transition that remains stable under reuse by controlling activation-quantization bias at the local scale of the activation distribution. We therefore apply a blockwise integer quantizer to the reused transition's weights and activations.

4.1 Activation quantization dominates transition bias

We first decompose the 4-bit error into a weight contribution and an activation contribution by quantizing each in isolation (Figure 6 in the appendix). Quantizing weights alone is comparatively benign: on Sudoku it moves the tied model from 84.1% to 78.5% and the untied model from 69.8% to 68.9%. The transition changes qualitatively only when activations are quantized as well. Full Int4 (4-bit w+a) collapses the tied model to 0.0% and the untied model to 1.6%, while 8-bit remains stable throughout (tied Int8 $= 84.1\%$). Reuse also compounds the weight-side error: with weight-only Int4 the untied model keeps 99% of FP accuracy versus 93% for the tied model (Section A.5). The dominant error term is the activation quantizer inside the reused transition, so a stable low-bit transition must explicitly control activation scales.

4.2 Granularity controls the transition kernel

The format ladder separates the number system from scaling granularity. We hold the embedding in FP except in the per-tensor Int4/Int8 rows (Figure 2, Section A.13). At 4-bit (w+a), both per-tensor formats lose accuracy completely (Sudoku Int4 0.0%, FP4 0.6%), while the matched-coverage per-block MXFP4 format recovers substantially to 72.7%. A separate, broader-coverage 6-bit result, MXFP6, reaches $84.9\% \approx$ FP 84.1%. Weight-only quantization stays close to full precision across every format, and all 8-bit formats operate at parity. The same split holds on the attention-based Maze task: per-tensor FP4 reaches 0.4%, matched-coverage MXFP4 reaches 85.1%, and the broader-coverage MXFP6 reference reaches 85.4%, so the effect extends well beyond Sudoku's MLP recursion. Maze adds one caveat: at *per-tensor* 4-bit the integer format keeps 73.5% while float falls to 0.4%, opposite to the usual LLM ordering, where float typically leads at 4-bit. We therefore scope our claim narrowly. The per-tensor failures span integer and (on Sudoku) float formats, while per-block scaling substantially restores both number systems, and per-tensor float never exceeds per-tensor integer in our data. Granularity is the mechanism that moves accuracy from near-zero to substantial recovery, and the residual per-tensor int-vs-float gap on Maze is orthogonal to this primary effect.

Mechanistically, per-block scaling succeeds exactly as Section 3.3 predicts: giving each block its own scale removes much of the systematic, nonzero-mean transition perturbation, and switching from per-tensor Int4 to per-block MXFP4 flips the Sudoku latent divergence from growing ($270 \to 274$) to shrinking ($159 \to 129$). Granularity alone controls the contraction-versus-drift boundary of Section 3.2, which also reconciles our results with the large-LLM MXFP4 consensus (Section A.2).

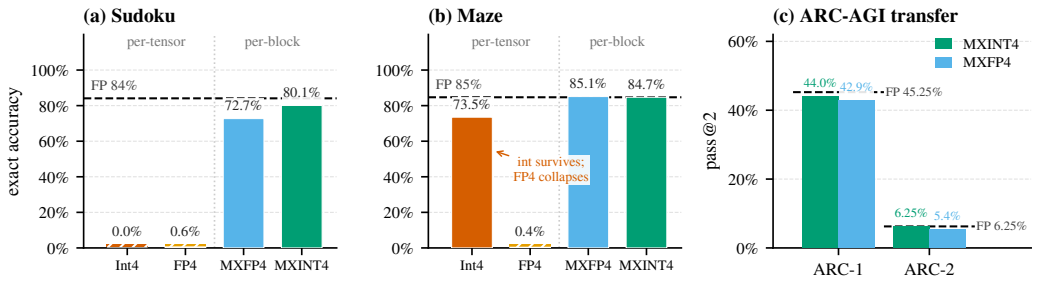


Figure 2: Scaling granularity controls stability. Four-bit formats (per-tensor Int4 also quantizes the embedding). **(a,b)** Per-tensor Int4/FP4 fail on Sudoku and FP4 fails on Maze, while per-block MXINT4/MXFP4 recover most or a substantial part of the lost accuracy, respectively. **(c)** MXINT4 also matches the FP point estimate on ARC-AGI (pass@2). Full results and 3-seed errors are in Table 1.

4.3 Ruling out non-granularity explanations

If granularity is the operative axis, the alternatives should fail to recover per-tensor Int4, and a sweep over granularity alone should isolate it. We confirm both. **Calibration freshness.** Replacing the frozen, calibrated per-tensor activation scale with one recomputed online at every recursion step (per-tensor *dynamic* Int4, no calibration) does not recover the collapse: static and dynamic per-tensor Int4 both reach 0.0% on Sudoku, and dynamic also collapses on Maze (to 0.0%, below the gracefully degrading static scale of Section 3.2), whereas per-block MXINT4 recovers full-precision accuracy on both (80.1%/84.7%) with a divergence that *decays* across recursion. This result shows that scale freshness alone is insufficient: the resulting step-varying error is not absorbed by the recursion. **Calibration size.** Increasing the PTQ calibration set $4\times$ ($512 \to 2048$ samples) leaves per-tensor Int4 at 0.0% on Sudoku. **Outlier corrections.** Sweeping SmoothQuant's migration strength and adding a Hadamard rotation leaves it at $\leq 0.7\%$ (full sweep in Table 3). **Granularity ladder.** Holding 4-bit, dynamic-absmax coverage and per-tensor weights fixed and sweeping only the activation group size, exact accuracy climbs monotonically: 0.0% (per-tensor) $\to 4.6\%$ (per-token) $\to 14.5\%$ (per-group-32) $\to 15.0\%$ (per-group-16), with nothing else moving, and matched-coverage MXINT4 lifts this $\sim$15% ceiling to 80.1%.[1] Clipping, freshness, calibration size, and static outlier corrections are all ruled out, and spatial granularity is the operative axis.

Section 3.3 predicted *why*: a coherent bias re-applied at every reuse. We measure this directly by projecting each layer's mean quantization error onto the dominant latent-divergence direction and tracking that direction across recursion steps. The direction is near-constant across steps (cross-step cosine 0.99 on Sudoku, 0.97 on Maze) and consistent across examples ($\|\overline{e^h}\|/\overline{\|e^h\|} \approx 0.9$), so the per-step errors accumulate coherently rather than averaging out. The diagnostic is defined in Section A.10.

4.4 The algorithm: blockwise integer transition quantization

Per-block float stabilizes the transition, but its per-element floating-point arithmetic is the most expensive part of a microscaling datapath, a real cost in the resource-constrained edge/TinyML regime we target (Dhahri & Urban, 2025; Zhuo et al., 2022). We therefore instantiate the same transition-level principle with integer mantissas. We quantize weights and activations of each recursive linear layer in blocks B of 32 along the input dimension. For each block B we choose a power-of-two shared scale s_B, set $q_{\max} = 2^{k-1} - 1$, use $s_B = 2^{\lceil \log_2(\max_{a \in B} |a|/q_{\max}) \rceil}$, and quantize symmetrically to the integer range $[-q_{\max}, q_{\max}]$, dequantizing before the next operation:

$$q_B = \mathrm{clip}\left(\left\lfloor \frac{a_B}{s_B} \right\rceil, -q_{\max}, q_{\max} \right), \qquad \widehat{a}_B = s_B q_B. \tag{3}$$

The scale is local enough to match the activation distribution encountered by the recursive operator, while the mantissa remains integer.

[1] The ladder holds weights per-tensor and uses a float absmax scale, isolating the *activation*-granularity axis, so its $\sim$15% ceiling is by design (full MXINT4 scales both weights and activations).

Table 1: PTQ format ladder (final-step exact accuracy). Per-tensor Int4/Int8 also quantize the embedding. Daggered MXFP6 is included only as a broader-coverage 6-bit reference. Per-tensor rows report mean±std over three calibration seeds; MXINT4 is deterministic given the checkpoint, and Sudoku MXFP4 is $72.7\% \pm 0.4$. QAT/LSQ details are in the appendix.

Quantizer (PTQ)	Activation scaling	Integer-MAC compatible	Sudoku (MLP)	Maze (attention)
Int8	per-tensor	✓	$84.0\% \pm .1$	$83.6\% \pm .3$
FP8 (e4m3)	per-tensor	✗	$84.1\% \pm .1$	$84.6\% \pm .2$
FP8 (e5m2)	per-tensor	✗	$84.0\% \pm .1$	$84.7\% \pm .2$
Int4	per-tensor	✓	$0.0\% \pm .01$	$73.5\% \pm .4$
FP4	per-tensor	✗	$0.6\% \pm .01$	$0.4\% \pm .3$
MXINT4	**per-block, integer**	✓	**80.1%**	**84.7%**
MXFP4	per-block, float	✗	$72.7\% \pm .4$	85.1%
MXFP6† (6-bit ref.)	per-block, float	✗	∼84.9%	∼85.4%
FP (reference)	n/a	✗	84.1%	84.7%

At matched coverage this MXINT4 quantizer, our OCP-style 4-bit integer analogue of microscaling (Rouhani et al., 2023; Open Compute Project, 2023), is competitive with per-block float on both task families (Figure 2). MXINT4 reaches 80.1% (Sudoku, MLP) and 84.7% (Maze, attention), versus MXFP4's 72.7% and 85.1%, placing MXINT4 closer to FP on Sudoku and the two formats essentially level on Maze. Repeating the Sudoku per-block rows across three calibration seeds, MXINT4 is bit-exact at 80.1% (its block scale is the per-block absmax computed online, so run-to-run variance is zero by construction), while MXFP4 is $72.7\% \pm 0.4\%$. The 7.4-point Sudoku gap is $\sim 17\times$ the MXFP4 spread, so it is not a single-seed artifact. We still read the two as competitive on these tasks (MXFP4 is marginally ahead on Maze) rather than asserting a universal integer-over-float ordering (Section 3.3 discusses the grid-bias interpretation).

The two 4-bit per-block formats have identical memory footprint (block-32, one shared E8M0 scale, 4.25 bits/element). In our accuracy experiments, both are emulated by fake quantization/dequantization before ordinary linear operations; we do not measure integer execution. MXINT4's integer elements and power-of-two scales are compatible with an integer-MAC datapath, whereas MXFP4 requires per-element float decoding, and both would require per-block scaled accumulation (Rouhani et al., 2023; Open Compute Project, 2023). We therefore treat the integer datapath as motivation rather than a measured benefit (Section 6).

4.5 Quantization-aware training: a secondary route

QAT alone compensates for much of the per-tensor activation loss: naive Int4 QAT improves Sudoku from 0.0% to $71.8\% \pm 0.3\%$ and Maze from 73.2% to $82.2\% \pm 0.7\%$, so the residual learned-quantizer gain lies almost entirely in the *weights*. We therefore evaluate weight-side LSQ (Esser et al., 2020), which adds $+9.3$ points where naive QAT struggles (Sudoku, $81.1\% \pm 0.6$) and is tied where it already succeeds (Maze, $81.8\% \pm 1.0\%$). Controls confirm the gain comes from the learned weight step size rather than bias precision, and that the earlier attention instability was specific to activation-LSQ (Section A.6, Table 2). Despite these QAT gains, per-block MXINT4 remains the primary post-training mechanism. Per-iteration normalization and cross-iteration distillation do not help at convergence (Section A.6).

4.6 Transfer to ARC-AGI

The rule also transfers to ARC-AGI, the open-ended abstraction-and-reasoning benchmark this line of work is known for (Jolicoeur-Martineau, 2025): few-shot grid-transformation tasks with no fixed output size, scored by *pass@2* (solved if either of two attempts matches the target, Section A.13). We evaluate ARC-AGI-1 and its harder 2025 successor ARC-AGI-2, the strongest test that our account extends beyond structured Sudoku/Maze environments (full table in Table 4, Section A.12, with the cross-architecture summary in panel (c) of Figure 2). Per-tensor Int4-PTQ gives 0.0% on both datasets, while two distinct routes return to full-precision-level pass@2: per-block scaling at post-training (MXINT4, no retraining, $44.0\%/6.25\% \approx$ FP) and naive Int4-QAT ($45.9\%/6.25\% \approx$ FP). The same granularity pattern holds. Weight-LSQ is neutral here (it helps the MLP recursion but not

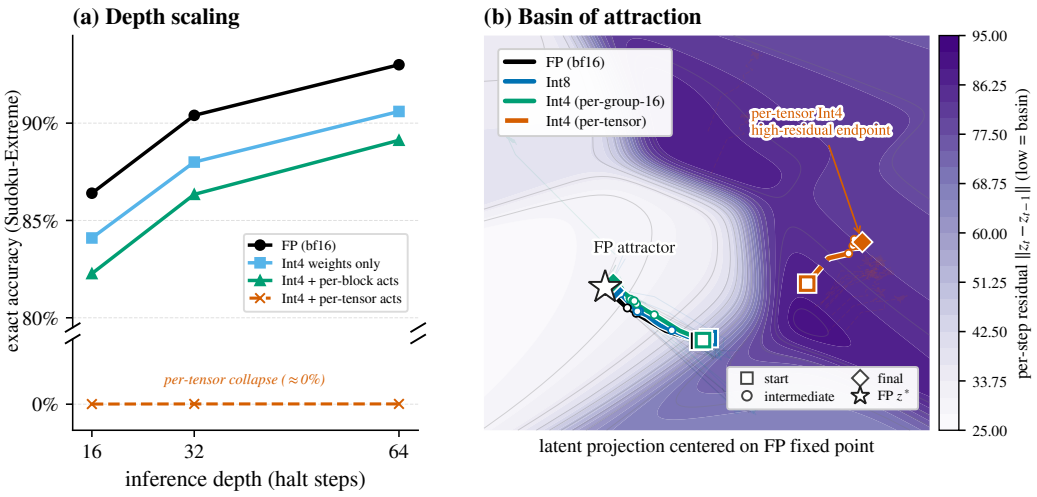


Figure 3: Per-block scaling preserves EqR depth scaling and FP-basin convergence. (a) Per-tensor Int4 activations remain near zero accuracy, while per-block activations restore depth scaling (89.1% at $D64$). All Int4 variants use per-channel weights. **(b)** FP, Int8, and per-group-16 Int4 approach the FP-centered basin; per-tensor Int4 does not. The 2-D PCA view is illustrative (≈30% variance); residuals and accuracy quantify the effect.

attention, Section 4.5). ARC-AGI-2 is the most quantization-sensitive, with Int8 losing about half its pass@2 score. Because all ARC numbers are single-seed and ARC-AGI-2 scores amount to only a few of its 120 tasks, we read them strictly as point estimates rather than a calibrated ordering.

5 Recursion Structure as Transition Robustness

Beyond TRM, a recursive reasoner's quantizability is shaped by its recursion structure: depth and reuse modulate sensitivity. Across the evaluated models and controls, the deepest model (EqR) is most sensitive, test-time stochasticity does not remove systematic bias, and per-block activation scaling remains effective.

5.1 Evaluated Recursive Models and Stochasticity Control

We compare two trained recursive models and one controlled intervention. EqRs iterate toward a fixed-point attractor and scale compute through depth and breadth (Bai et al., 2019; Huang et al., 2026); TRM follows a finite refinement path; and injecting Probabilistic Tiny Recursive Model (PTRM)-style test-time noise into TRM probes whether stochastic breadth can average away the error. Trained stochastic transitions remain outside our experimental scope (Sghaier et al., 2026; Baek et al., 2026).

5.2 Equilibrium Recursion is Especially Sensitive

EqR pushes recursion depth to the extreme: its released Sudoku-Extreme checkpoint unrolls roughly 288 layer applications at our base depth (far more at higher depths) and reads its answer from a converged attractor (Huang et al., 2026), making it the sharpest test of our accumulation hypothesis: a systematic per-step bias here can displace the entire attractor.[2]

In full precision EqR scales with depth: exact accuracy rises from 86.4% at $D16$ to 93.0% at $D64$ as the final-step residual contracts ($27.5 \to 16.2$, Figure 3). Per-tensor Int4 *activation* quantization destroys this: accuracy drops to effectively zero at every depth (0.01% at $D16$, $0.02\% \pm 0.002\%$ across three seeds at $D32$ against FP 90.4%) and the equilibrium fails to converge, with the residual

[2]We use a minimal `bf16`-consistent fake-quantizer that is FP-neutral (verified FP 86.4%). An off-the-shelf harness was not, over ∼288 applications. EqR numbers are for the one released Sudoku-Extreme checkpoint, multi-seed at $D32$ and single-seed at other depths (Section A.13).

stuck near 110 versus $\sim$16 even with a favorable dynamic scale, a massive displacement of the full-precision attractor (Figure 3b).

As with TRM, activation-side sensitivity dominates: per-channel weight-only Int4 loses little and keeps depth scaling ($84.1\% \rightarrow 90.6\%$ across $D16$–$D64$), and the per-tensor weight-only drop to 5.1% is a coarseness artifact. EqR's stochastic *breadth* also fails to average the activation error away. Aggregation improves full precision (majority vote 91.4%, any-correct 93.2%) but Int4 stays at $\sim 0\%$: every trajectory settles into the same shifted basin and agrees on wrong answers, confirming a coherent, systematic bias (Section A.4).

Finally, our blockwise rule generalizes to this extreme (Figure 9): per-group-16 scaling lifts full-Int4 exact accuracy from 0.01% to 82.3% at $D16$ (per-token 60.6%, per-group-32 82.2%, within ~ 4 points of FP 86.4%) and restores depth scaling (86.3% at $D32$, 89.1% at $D64$). Per-block scaling preserves the most sensitive trained model we evaluate.

5.3 Test-time stochasticity does not remove the systematic bias

As a stochasticity control, test-time stochasticity fails to rescue per-tensor Int4 TRM: we inject PTRM-style Gaussian noise into the deterministic TRM checkpoint, rather than a trained stochastic model, and sweep $K \in \{1, 4, 16\}$ trajectories (FP also at $K{=}64$) and all noise scales. Both averaging and halt-head selection keep exact accuracy at 0.0%, while the same selector lifts full precision to 95.0% (Section A.4, Figure 8). The same biased transition drives every trajectory, so aggregation cannot recover an unbiased answer. Whether *trained-in* stochasticity (e.g. GRAM's learned per-step transitions (Baek et al., 2026)) confers robustness remains open (Section A.4).

Across the tested settings, severity tracks effective depth: EqR ($\sim$288 applications) is most sensitive, test-time stochastic defenses do not mitigate drift, and per-block activation scaling works throughout.

6 Discussion and limitations

A transition-level rule, and what governs it. We quantize recursive reasoners at the level of their reused transition operator, using local blockwise activation scales with integer mantissas and power-of-two block scales. Per-tensor 4-bit fails because its activation grid induces a biased perturbation re-applied throughout the trajectory, which blockwise scaling reduces locally. More broadly, quantizability depends on more than bit-width and number format. It also depends on the recursive structure that re-applies the transition: how often the operator is reused, whether the answer is read from a finite trajectory or a fixed point, and whether stochastic transitions merely carry variance or absorb systematic bias.

Hardware. Our experiments emulate MXINT4 and MXFP4 with fake quantization/dequantization and do not benchmark kernels. Both have a 4.25-bit/element footprint and require per-block scaled accumulation. MXINT4's integer elements and power-of-two scales are compatible with integer MACs, whereas MXFP4 requires float decoding. NVIDIA Blackwell and AMD MI355X execute MXFP4 natively (Alvarez et al., 2025; AMD, 2026), and Tenstorrent devices support block floating-point formats in which 16 values share one exponent (Tenstorrent, 2026), structurally close to MXINT4. Support on edge accelerators remains limited, and measured kernels are future work.

Limitations. Our evaluations cover TRM on Sudoku-Extreme and Maze-Hard, extended to EqR, PTRM, QAT/LSQ, and ARC-AGI (Sections 4.5, 4.6, and 5). Per-tensor PTQ and QAT/weight-LSQ report mean$\pm$std over three seeds, and other measurements are single-seed or deterministic. HRM-scale generality and trained-in stochasticity (GRAM) remain future work. Our microscaling result is regime-specific, consistent with the large-LLM MXFP4 consensus (Zhang et al., 2026a; Fasoli et al., 2026). Finally, Proposition 1 is a qualitative scaffold rather than a tight bound or full fixed-point perturbation theory.

AI use statement

In this work, we used generative AI tools, including Anthropic Claude and OpenAI Codex, to assist with performance engineering of the experimental code, including refactoring and improving

execution efficiency, and to improve the grammar, clarity, and concision of author-written manuscript text. We also used these tools to search for and summarize related work, which helped us identify papers to read. We further used these tools to help formalize the proof of Proposition 1 (Appendix A.11); the authors checked the argument, and the result was verified with a Lean 4 formalization. The tools were not used to formulate research questions or hypotheses, design experiments, generate data or experimental results, or draw scientific conclusions. The authors reviewed all retained AI-assisted code changes and validated the final implementation by executing the relevant experiments and inspecting their recorded outputs. All AI-assisted language edits were reviewed and approved by the authors. The authors take responsibility for the final text, code, claims, and artifacts.

Reproducibility statement

The completed experiments are evaluated on the public Sudoku-Extreme and Maze-Hard benchmarks using models trained with the released TRM codebase and the quantization code described in Section 4, and on ARC-AGI-1/2. Exact configurations, seeds, and per-step logging scripts are provided in the supplementary material.

References

Md Zahangir Alom, Adam T Moody, Naoya Maruyama, Brian C Van Essen, and Tarek M Taha. Effective quantization approaches for recurrent neural networks, 2018. URL https://arxiv.org/abs/1802.02615.

Eduardo Alvarez, Omri Almog, Eric Chung, Simon Layton, Dusan Stosic, Ronny Krashinsky, and Kyle Aubrey. Introducing NVFP4 for efficient and accurate low-precision inference. NVIDIA Technical Blog, June 2025, 2025.

AMD. AMD Instinct MI350 series microarchitecture. ROCm documentation, https://rocm.docs.amd.com/en/latest/reference/gpu-arch/mi350.html, 2026. Accessed: 2026-09-24.

Kushal Arora, Layla El Asri, Hareesh Bahuleyan, and Jackie Chi Kit Cheung. Why exposure bias matters: An imitation learning perspective of error accumulation in language generation. In *Findings of the Association for Computational Linguistics (ACL)*, 2022.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. QuaRot: Outlier-free 4-bit inference in rotated LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2404.00456.

Junyeob Baek, Mingyu Jo, Minsu Kim, Mengye Ren, Yoshua Bengio, and Sungjin Ahn. Generative recursive reasoning, 2026. URL https://arxiv.org/abs/2605.19376.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Ramshankar Bhuvaneswaran and Handan Liu. BitSkip: An empirical analysis of quantization and early exit composition in transformers, 2025. URL https://arxiv.org/abs/2510.23766.

Mengzhao Chen, Meng Wu, Hui Jin, Zhihang Yuan, Jing Liu, Chaoyi Zhang, Yunshui Li, Jie Huang, Jin Ma, Zeyue Xue, Zhiheng Liu, Xingyan Bin, and Ping Luo. INT v.s. FP: A comprehensive study of fine-grained low-bit quantization formats, 2025. URL https://arxiv.org/abs/2510.25602.

Yi-Chung Chen, Zhi-Kai Huang, and Jing-Ren Chen. StepbaQ: Stepping backward as correction for quantized diffusion models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan,

U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 54054–54078. Curran Associates, Inc., 2024. doi: 10.52202/079017-1712. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/615675cc6e94ddb1a783904fb178b5f6-Paper-Conference.pdf.

Brian Chmiel, Maxim Fishman, Ron Banner, and Daniel Soudry. FP4 all the way: Fully quantized training of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2505.19115.

Euntae Choi, Sumin Song, and Sungjoo Yoo. LAQuant: A simple overhead-free large reasoning model quantization by layer-wise lookahead loss, 2026. URL https://arxiv.org/abs/2605.08755.

Huanpeng Chu, Wei Wu, Chengjie Zang, and Kun Yuan. QNCD: Quantization noise correction for diffusion models. In *ACM International Conference on Multimedia (ACM MM)*, 2024.

Ian Colbert, Alessandro Pappalardo, and Jakoba Petri-Koenig. A2Q: Accumulator-aware quantization with guaranteed overflow avoidance. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023a.

Ian Colbert, Alessandro Pappalardo, and Jakoba Petri-Koenig. Quantized neural networks for low-precision accumulation with guaranteed overflow avoidance, 2023b. URL https://arxiv.org/abs/2301.13376.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations (ICLR)*, 2019.

Rayen Dhahri and Steffen Urban. Quant-trim in practice: Improved cross-platform low-bit deployment on edge NPUs, 2025. URL https://arxiv.org/abs/2511.15300.

Vage Egiazarian, Roberto L. Castro, Denis Kuznedelev, Andrei Panferov, Eldar Kurtic, Shubhra Pandit, Alexandre Noll Marques, Mark Kurtz, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Bridging the gap between promise and performance for microscaling FP4 quantization. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=zCBGe9AqJZ. arXiv:2509.23202.

Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2020.

Rui Fang, Hsi-Wen Chen, and Ming-Syan Chen. LoopQ: Quantization for recursive transformers, 2026. URL https://arxiv.org/abs/2605.16343.

Andrea Fasoli, Chia-Yu Chen, Mauricio Serrano, Xiao Sun, Naigang Wang, Swagath Venkataramani, George Saon, Xiaodong Cui, Brian Kingsbury, Wei Zhang, Zoltán Tüske, and Kailash Gopalakrishnan. 4-bit quantization of LSTM-based speech recognition models. In *Interspeech*, 2021. URL https://arxiv.org/abs/2108.12074. arXiv:2108.12074.

Andrea Fasoli, Monodeep Kar, Chi-Chun Liu, Swagath Venkataramani, Viji Srinivasan, Leland Chang, and Naigang Wang. Is finer better? The limits of microscaling formats in large language models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2601.19026.

Natalia Frumkin and Diana Marculescu. Q-Sched: Pushing the boundaries of few-step diffusion models with quantization-aware scheduling, 2025. URL https://arxiv.org/abs/2509.01624.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Alex Graves. Adaptive computation time for recurrent neural networks, 2016. URL https://arxiv.org/abs/1603.08983.

Qinyao He, He Wen, Shuchang Zhou, Yuxin Wu, Cong Yao, Xinyu Zhou, and Yuheng Zou. Effective quantization methods for recurrent neural networks, 2016. URL https://arxiv.org/abs/1611.10176.

Tianxing He, Jingzhao Zhang, Zhiming Zhou, and James Glass. Exposure bias versus self-recovery: Are distortions really incremental for autoregressive text generation? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5087–5102. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.415. URL https://aclanthology.org/2021.emnlp-main.415/.

Yefei He, Luping Liu, Jing Liu, Weijia Wu, Hong Zhou, and Bohan Zhuang. PTQD: Accurate post-training quantization for diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning, 2026. URL https://arxiv.org/abs/2605.21488. ICML 2026.

Yushi Huang, Ruihao Gong, Jing Liu, Tianlong Chen, and Xianglong Liu. TFMQ-DM: Temporal feature maintenance quantization for diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18, 2018.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks, 2025. URL https://arxiv.org/abs/2510.04871. Samsung SAIL Montréal.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations (ICLR)*, 2020.

James Li, Philip H. W. Leong, and Thomas Chaffey. Quantization robustness of monotone operator equilibrium networks, 2026a. URL https://arxiv.org/abs/2603.10562.

Mingxiao Li, Tingyu Qu, Ruicong Yao, Wei Sun, and Marie-Francine Moens. Alleviating exposure bias in diffusion models through sampling with shifted time steps. In *International Conference on Learning Representations (ICLR)*, 2024.

Xiuyu Li, Yijiang Liu, Long Lian, Huanrui Yang, Zhen Dong, Daniel Kang, Shanghang Zhang, and Kurt Keutzer. Q-Diffusion: Quantizing diffusion models. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Zhen Li, Yupeng Su, Runming Yang, Congkai Xie, Zheng Wang, Zhongwei Xie, Ngai Wong, and Hongxia Yang. Quantization meets reasoning: Exploring LLM low-bit quantization degradation for mathematical reasoning, 2025. URL https://arxiv.org/abs/2501.03035.

Zhen Li, Yupeng Su, Songmiao Wang, Runming Yang, Congkai Xie, Aofan Liu, Ming Li, Jiannong Cao, Yuan Xie, Ngai Wong, and Hongxia Yang. Quantization meets reasoning: Exploring and mitigating degradation of low-bit LLMs in mathematical reasoning, 2026b. URL https://arxiv.org/abs/2505.11574.

Ruikang Liu, Yuxuan Sun, Manyi Zhang, Haoli Bai, Xianzhi Yu, Tiezheng Yu, Chun Yuan, and Lu Hou. Quantization hurts reasoning? An empirical study on quantized reasoning models. In *Conference on Language Modeling (COLM)*, 2025a. arXiv:2504.04823.

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)*, 2025b. arXiv:2405.16406.

Keyu Lv, Manyi Zhang, Xiaobo Xia, Jingchen Ni, Shannan Yan, Xianzhi Yu, Lu Hou, Chun Yuan, and Haoli Bai. What makes low-bit quantization-aware training work for reasoning LLMs? A systematic study, 2026. URL https://arxiv.org/abs/2601.14888.

Mang Ning, Enver Sangineto, Angelo Porrello, Simone Calderara, and Rita Cucchiara. Input perturbation reduces exposure bias in diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 26245–26265. PMLR, 2023. URL https://proceedings.mlr.press/v202/ning23a.html.

Mang Ning, Mingxiao Li, Jianlin Su, Albert Ali Salah, and Itir Onal Ertugrul. Elucidating the exposure bias in diffusion models. In *International Conference on Learning Representations (ICLR)*, 2024.

Open Compute Project. OCP microscaling formats (MX) specification version 1.0, 2023. URL https://www.opencompute.org/documents/ocp-microscaling-formats-mx-v1-0-spec-final-pdf. Accessed May 30, 2026.

Joachim Ott, Zhouhan Lin, Ying Zhang, Shih-Chii Liu, and Yoshua Bengio. Recurrent neural networks with limited numerical precision, 2016. URL https://arxiv.org/abs/1608.06902.

Eunhyeok Park, Dongyoung Kim, Sungjoo Yoo, and Peter Vajda. Precision highway for ultra low-precision quantization, 2018. URL https://arxiv.org/abs/1812.09818.

Paulius Rauba, Claudio Fanconi, and Mihaela van der Schaar. Tiny autoregressive recursive models, 2026. URL https://arxiv.org/abs/2603.08082.

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.

Sooyoung Ryu, Mathieu Salzmann, and Saqib Javed. Q-Drift: Quantization-aware drift correction for diffusion model sampling, 2026. URL https://arxiv.org/abs/2603.18095.

Amin Sghaier, Ali Parviz, and Alexia Jolicoeur-Martineau. Probabilistic tiny recursive model, 2026. URL https://arxiv.org/abs/2605.19943.

Yuzhang Shang, Zhihang Yuan, Bin Xie, Bingzhe Wu, and Yan Yan. Post-training quantization on diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Junhyuk So, Jungwon Lee, Daehyun Ahn, Hyungjun Kim, and Eunhyeok Park. Temporal dynamic quantization for diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Tenstorrent. TT-Metalium tech report: Data formats. https://github.com/tenstorrent/tt-metal/blob/main/tech_reports/data_formats/data_formats.md, 2026. Accessed: 2026-09-24.

Albert Tseng, Tao Yu, and Youngsuk Park. Training LLMs with MXFP4. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 1630–1638. PMLR, 2025. URL https://proceedings.mlr.press/v258/tseng25a.html.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model, 2025. URL https://arxiv.org/abs/2506.21734. Sapient Intelligence.

Guan Wang, Changling Liu, Chenyu Wang, Cai Zhou, Yuhao Sun, Yifei Wu, Shuai Zhen, Luca Scimeca, and Yasin Abbasi Yadkori. HRM-Text: Efficient pretraining beyond scaling, 2026. URL https://arxiv.org/abs/2605.20613.

Wenlong Wang and Fergal Reid. Tiny recursive reasoning with Mamba-2 attention hybrid, 2026. URL https://arxiv.org/abs/2602.12078.

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

Kevin Xu and Issei Sato. On expressive power of looped transformers: Theoretical analysis and enhancement via timestep encoding. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69613–69646. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/xu25x.html.

Chenxiao Yang, Nathan Srebro, and Zhiyuan Li. Recursive models for long-horizon reasoning, 2026. URL https://arxiv.org/abs/2603.02112.

Yuzhe Yao, Feng Tian, Jun Chen, Haonan Lin, Guang Dai, Yong Liu, and Jingdong Wang. Timestep-aware correction for quantized diffusion models. In *European Conference on Computer Vision (ECCV)*, 2024.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. ZeroQuant: Efficient and affordable post-training quantization for large-scale transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Manyi Zhang, Ji-Fu Li, Zhongao Sun, Haoli Bai, Hui-Ling Zhen, Zhenhua Dong, and Xianzhi Yu. Benchmarking post-training quantization of large language models under microscaling floating point formats, 2026a. URL https://arxiv.org/abs/2601.09555.

Nan Zhang, Eugene Kwek, Yusen Zhang, Muyu Pan, Suhang Wang, Prasenjit Mitra, and Rui Zhang. QuantLRM: Quantization of large reasoning models via fine-tuning signals, 2026b. URL https://arxiv.org/abs/2602.02581.

Yijia Zhang, Lingran Zhao, Shijie Cao, Wenqiang Wang, Ting Cao, Fan Yang, Mao Yang, Shanghang Zhang, and Ningyi Xu. Integer or floating point? New outlooks for low-bit quantization on large language models, 2023. URL https://arxiv.org/abs/2305.12356.

Jiaqi Zhao, Ming Wang, Miao Zhang, Yuzhang Shang, Xuebo Liu, Yaowei Wang, Min Zhang, and Liqiang Nie. Benchmarking post-training quantization in LLMs: Comprehensive taxonomy, unified evaluation, and comparative analysis, 2025. URL https://arxiv.org/abs/2502.13178.

Ke Zhu, Yin-Yin He, and Jianxin Wu. Quantized feature distillation for network quantization. In *AAAI Conference on Artificial Intelligence*, 2023.

Shaojie Zhuo, Hongyu Chen, Ramchalam Kinattinkara Ramakrishnan, Tommy Chen, Chen Feng, Yicheng Lin, Parker Zhang, and Liang Shen. An empirical study of low precision quantization for TinyML, 2022. URL https://arxiv.org/abs/2203.05492.

A SUPPORTING FIGURES AND DETAILS

This appendix collects supporting evidence referenced in the main text. The main QAT/weight-LSQ rows (Table 2) are reported over three seeds. All remaining tables report single-seed point estimates.

A.1 PER-STEP ACCURACY CURVES

Figure 4 collects the per-step trajectory diagnostic (accuracy and latent divergence) for all four formats, complementing the divergence curves of Figure 1. At 8-bit, accuracy steadily rises toward full precision as the recursion proceeds while the divergence contracts. Conversely, at per-tensor 4-bit, accuracy either fails to improve entirely (Sudoku) or saturates below full precision (Maze).

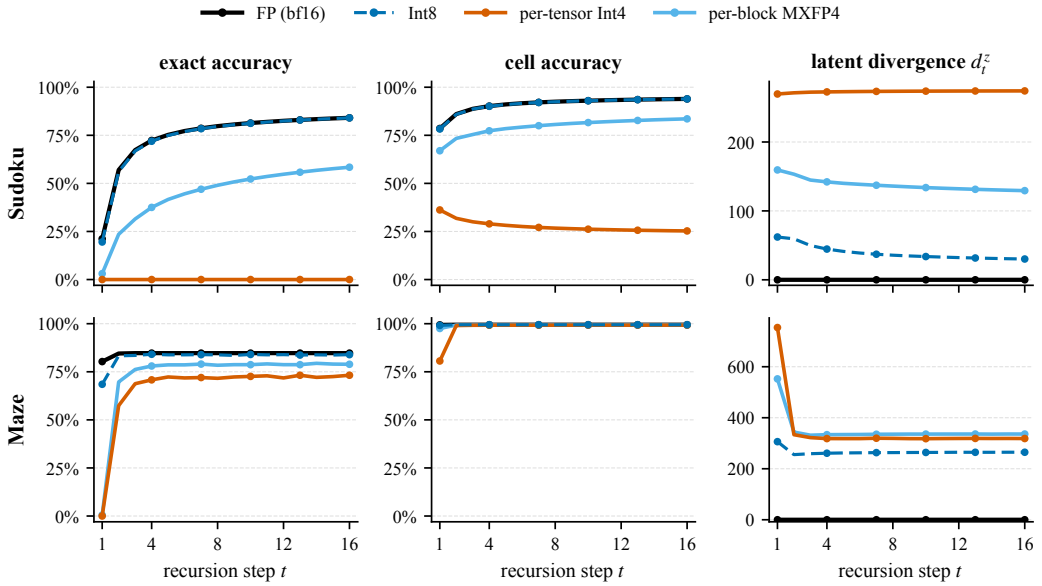


Figure 4: Per-step trajectory atlas (Int8 and per-tensor Int4 use the broader integer coverage of Table 1, including the embedding; MXFP4 keeps the embedding in FP). Exact accuracy, cell accuracy, and latent divergence d_t^z vs. recursion step for FP, Int8, per-tensor Int4, and per-block MXFP4, on Sudoku (top) and Maze (bottom). Int8 tracks full precision and its divergence contracts; per-tensor Int4 collapses on the deep Sudoku recursion (flat-high divergence, 0% exact) yet survives the shallow Maze; per-block MXFP4 converts the growing per-tensor divergence into a shrinking one, recovering most accuracy. Int8 is dashed so it does not occlude the FP curve.

A.2 MICROSCALING STABILIZES THE ACTIVATION TRANSITION

Figure 4 shows the trajectory-level view behind the granularity result of Section 4.2: step-by-step, moving from per-tensor Int4 to per-block MXFP4 converts a growing latent divergence into a shrinking one on both tasks.

This connects our result to a current tension in the LLM literature. While 4-bit MXFP is widely reported as lossy and an open challenge (Zhang et al., 2026a; Egiazarian et al., 2026), our results show that per-block 4-bit matches or approaches full precision on structured reasoning tasks. A plausible reason is the operating regime: tiny weight-tied models may have narrow, well-behaved weight distributions that are well matched to per-block scaling, avoiding the microscaling block-size limits typically reported for large LLMs (Fasoli et al., 2026).

A.3 PER-BLOCK INTEGER ON SUDOKU

Figure 5 reports the per-block integer (MXINT) sweep on Sudoku referenced in Section 4.4. This includes the MXINT8 gate result (84.1%, matching FP). The MXINT4 weight-only configuration, not plotted, reaches 82.5%.

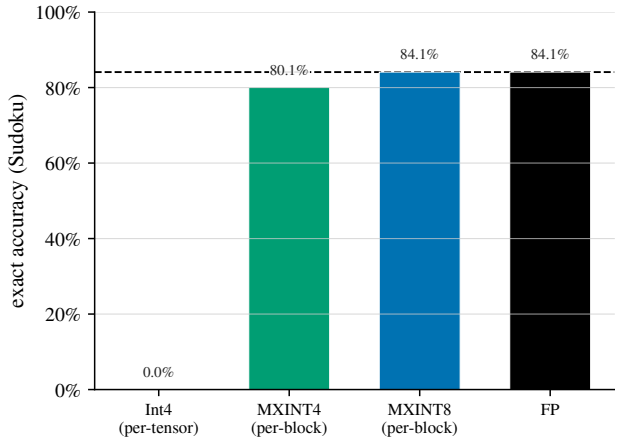


Figure 5: Per-block integer (MXINT) results on Sudoku. MXINT8 matches FP; MXINT4 preserves near-FP accuracy where per-tensor Int4 does not.

A.4 RECURSION-STRUCTURE SUPPORTING FIGURES

Figure 7a provides the EqR breadth result behind Section 5.2. Aggregating stochastic trajectories successfully lifts full-precision performance (majority vote 91.4%, any-correct 93.2%). However, under Int4, accuracy remains pinned at $\sim$0%. Because every trajectory follows the exact same biased transition, averaging simply cannot expose an unbiased path.

Figure 7b provides the per-step view behind Section 5.3. Under Int4, deterministic and PTRM trajectories (whether using averaging or selection) stay at zero accuracy at every single recursion step. Simultaneously, the latent divergence remains trapped near a shifted attractor.

Test-time stochasticity (PTRM), in full. We begin with the near-zero Int4 (4-bit w+a) TRM from Section 3. We then inject PTRM-style test-time stochasticity (Sghaier et al., 2026). This adds zero-mean Gaussian noise at each recursion step, with K trajectories aggregated either by averaging or by selecting the best trajectory using TRM's learned halt head. We sweep K and the noise scale without retraining.

Int4 exact-accuracy stays firmly at 0.0% for every $K \in \{1, 4, 16\}$ and every noise standard deviation in $\{0.02, \ldots, 1.0\}$, applied to either latent state. Larger noise only degrades cell-accuracy. Averaging modestly shrinks the latent divergence ($274 \to 241$) but fails to improve exact accuracy, while selection simply commits to a single biased trajectory. The control is the exact same machinery on full precision, where halt-head selection lifts accuracy monotonically ($84.1\% \to 89.5\% \to 93.3\% \to 95.0\%$ for $K = 1, 4, 16, 64$). The selection mechanism is real and implemented correctly. However, Int4 offers no high-accuracy trajectory to select because every path is corrupted by the same biased transition.

Trained-in stochasticity (GRAM). The PTRM result specifically addresses stochasticity added *at inference*. Whether *trained-in* stochasticity confers inherent robustness remains an open question for future work. Models like GRAM utilize learned variational per-step transitions (Baek et al., 2026), but their code and checkpoints are currently unreleased. A controlled proxy, a TRM trained with additive per-step Gaussian noise (std 0.05–0.2) and then quantized, still reaches 0.0% Int4 exact accuracy at every noise level, although cell accuracy improves at some levels (single seed). This proxy is not a GRAM reproduction, so whether amortized-variational stochasticity confers tolerance to a *systematic* quantization bias remains open.

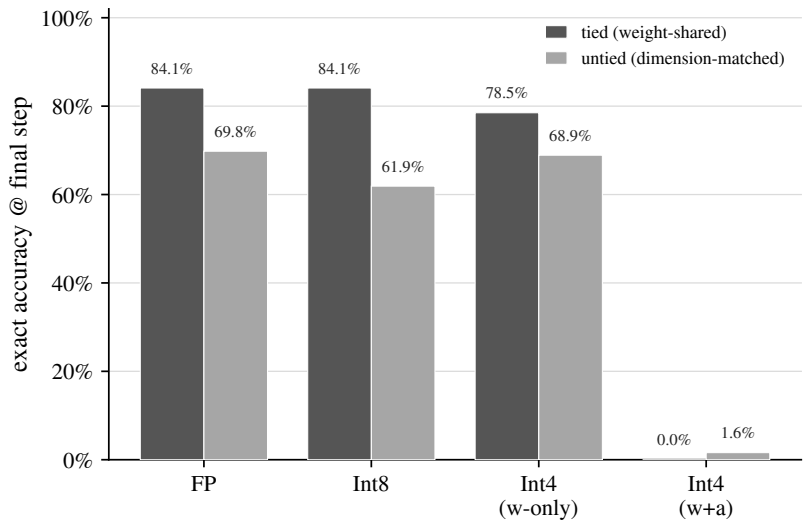
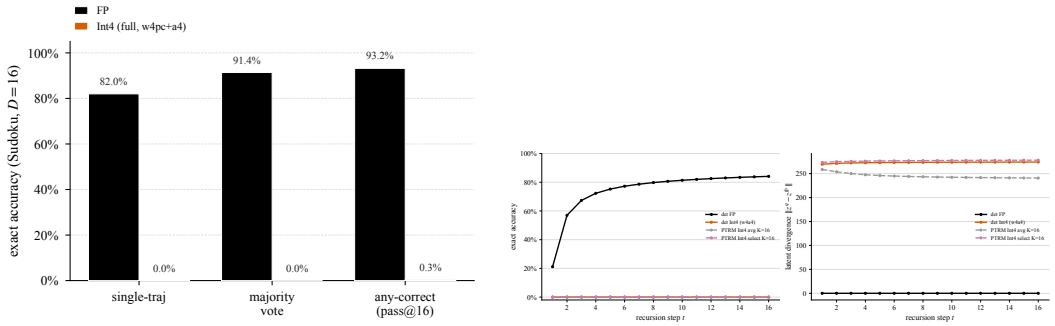


Figure 6: Activation quantization dominates transition bias. Weight-vs.-activation decomposition on Sudoku for tied and untied TRM. Weight-only Int4 shifts accuracy only modestly (tied $84.1\% \rightarrow 78.5\%$, a $\sim 7\%$ relative drop); adding activation quantization (4-bit w+a) changes the transition enough to remove exact solutions (tied $\rightarrow 0.0\%$, untied $\rightarrow 1.6\%$). Int8 remains stable.



(a) EqR: stochastic breadth.

(b) PTRM on TRM: per-step accuracy and divergence.

Figure 7: Supporting evidence for the recursion-structure analysis. (a) On EqR, breadth improves full precision but not Int4. Single-traj is the mean accuracy of one stochastically initialized trajectory out of 16, hence below the deterministic FP baseline (86.4%). (b) On TRM, test-time stochasticity leaves Int4 accuracy at zero across all recursion steps.

The basin panel of Figure 3(b) visualizes the granularity effect as a basin landscape. The surface is a Gaussian-smoothed interpolation of measured per-step residuals over a two-dimensional PCA projection of FP-centered EqR states. (This should be read as a trajectory visualization rather than a mathematically fitted potential function). The full-precision path, Int8, and per-block MXINT4 all smoothly move toward the low-residual region around the FP attractor. By contrast, per-tensor Int4 terminates in a high-residual region of the projected state space, despite using the exact same weights as MXINT4.

A.5 Reuse controls: dose–response and the untied weight control

Dose–response in transition reuse. If the observed failure is genuinely driven by accumulation, its severity must scale with how often the block is reused. We tested this by sweeping the number of inner recursion cycles, directly multiplying the per-step reuse count. We then measured the full-precision-vs.-Int4 gap at the final step (Figure 10).

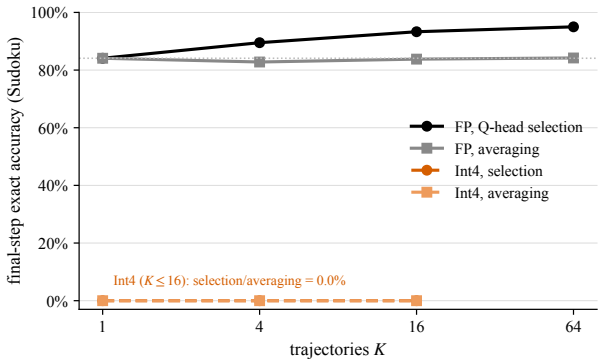


Figure 8: Test-time stochasticity does not improve Int4 accuracy. Final-step exact-accuracy vs. number of trajectories K on Sudoku (PTRM on TRM, noise 0.05). On full precision, halt-head selection climbs $84.1\% \rightarrow 95.0\%$ (averaging flat at $\sim 84\%$); on Int4, both averaging and selection remain at 0.0% for all evaluated $K \leq 16$ (the $K = 64$ Int4 run was memory-limited). The perturbation is systematic, so averaging and selection do not expose an unbiased trajectory.

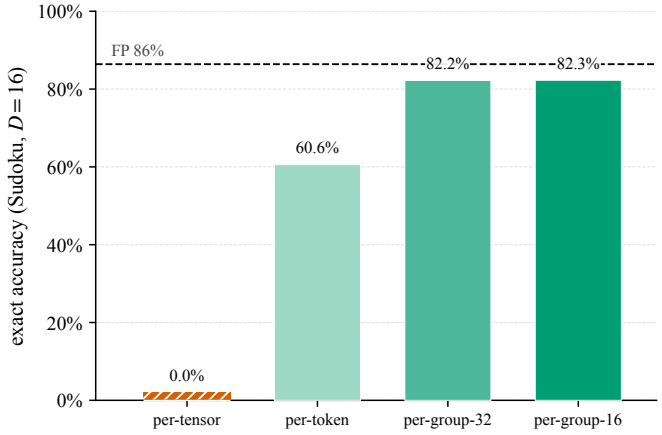


Figure 9: Activation granularity controls EqR quantization. Full-Int4 EqR exact-accuracy at $D16$ (weights per-channel) as activation scaling is refined: per-tensor $0.01\% \rightarrow$ per-token $60.6\% \rightarrow$ per-group-32 $82.2\% \rightarrow$ per-group-16 82.3%, approaching full precision (86.4%). The same per-block principle that preserves TRM and Maze also preserves the extreme-depth equilibrium model.

On Sudoku, the gap strictly *widens* with reuse. Full-precision cell-accuracy rises from 81% to 94% as cycles increase. In stark contrast, Int4 cell-accuracy *falls* from 35% to 25%, and the Int4 latent divergence jumps once enough cycles are present. Adding recursion makes the full-precision model better and the quantized model worse. This is the signature of error accumulation. On Maze, the same sweep remains safely in the contraction regime: Int4 exact-accuracy climbs from 44% to 76% and latent divergence shrinks ($385 \rightarrow 324$). The two tasks simply sit on opposite sides of the contraction-versus-drift boundary. Whether the quantized transition contracts or drifts sets the regime, and reuse depth amplifies it: more reuse compounds the drift on Sudoku but gives the contracting Maze trajectory more steps to converge.

Untied dimension-matched control. To isolate the effect of weight reuse from the inherent severity of 4-bit quantization, we built a dimension-matched control. This control *unties* the L-cycle positions, providing each position with its own distinct weights ($7\times$ less intra-pass reuse, and correspondingly $7\times$ more parameters in those layers). We preserved the rest of the architecture and training pipeline identically.

Figure 11 provides an informative result in both directions. Reuse clearly *is* a primary driver of the error. Under Int4, the untied model retains roughly twice the cell-accuracy of the tied model (54% vs.

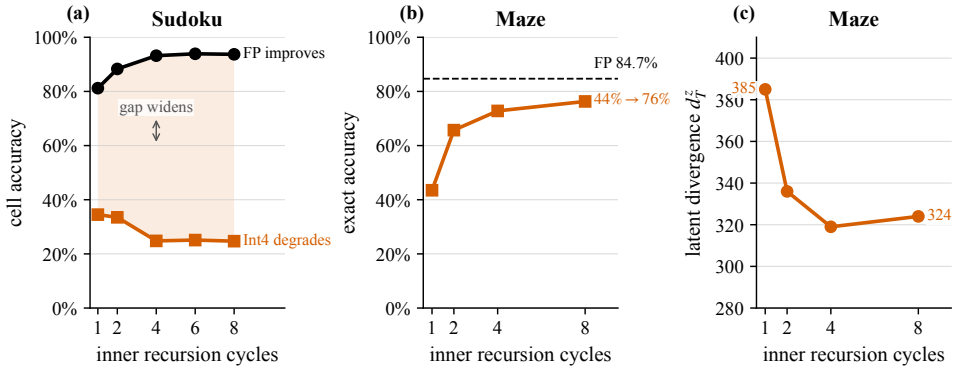


Figure 10: Dose–response to transition reuse. Sweeping inner recursion cycles L changes how many times the quantized block is applied before read-out (Sudoku $L \in \{1, 2, 4, 6, 8\}$, Maze $L \in \{1, 2, 4, 8\}$). **(a)** On Sudoku the FP–Int4 cell-accuracy gap *widens* monotonically with reuse (accumulation). **(b,c)** On the shallower Maze recursion Int4 instead *self-corrects*: exact accuracy climbs toward FP and latent divergence shrinks. Reuse amplifies whichever regime the transition is in, as the accumulation account predicts.

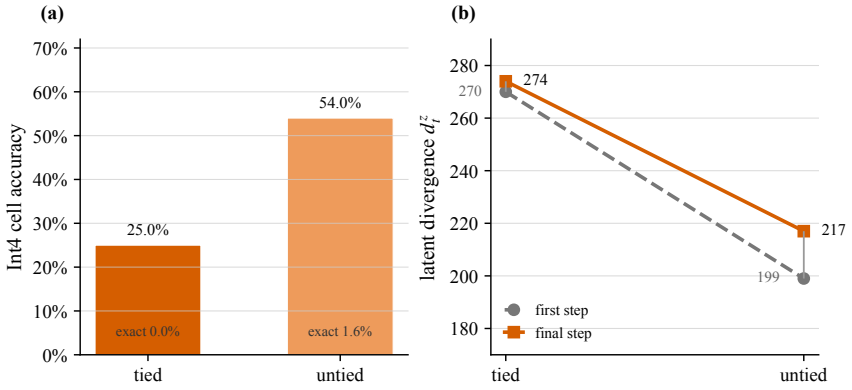


Figure 11: Dimension-matched untied control. Untying the L-cycle positions ($7\times$ less reuse) roughly doubles Int4 cell-accuracy and lowers final latent divergence. Untied Int4 still has near-zero exact accuracy (1.6%), so reuse amplifies the failure of exact solutions without solely causing it.

25%, or 61% vs. 27% of its own full-precision baseline). It also maintains a lower, flatter trajectory divergence (217 vs. 274). However, reuse is incomplete as a sole explanation. Even fully untied, exact-accuracy remains near zero (1.6%), and the trajectory divergence still grows over time instead of decaying. The untied model's full-precision baseline is also lower (69.8% vs. 84.1% exact), which by itself makes exact accuracy more fragile under perturbation.

This happens because the reuse that survives untying is still substantial. Per forward pass, the fully tied model applies each down-projection weight 336 times. Untying the inner L-cycle positions cuts this by $7\times$, resulting in 48 reuses per weight (contributed by the surviving outer H-cycle and deep-supervision loops). Going below 48 reuses would require completely untying those outer loops as well. That residual reuse of 48 applications, combined with the intrinsic harshness of per-tensor 4-bit quantization, is enough to destroy exact solutions. Ultimately, weight reuse heavily amplifies transition drift and is necessary to explain the magnitude of the displacement. However, the near-zero accuracy reflects a combination of this reuse and the underlying activation-quantization granularity problem.

Table 2: Quantization-aware training: a secondary, weight-side route (12k steps; QAT rows report the best periodic evaluation over training). QAT alone already compensates for much of the per-tensor activation loss (naive Int4 QAT, Sudoku $0.0\% \rightarrow 71.8\%$); the residual learned-quantizer gain is in the *weights*: weight-LSQ adds a significant $+9.3$ points on Sudoku (gap $\gg$ std) and is statistically tied on Maze. The naive-QAT and weight-LSQ rows are mean$\pm$std over 3 seeds; the decomposition rows are single-seed. The Int4 PTQ row is the single-seed QAT starting point under the same broader coverage as Table 1, which reports the three-seed ladder mean ($73.5\% \pm 0.4$ on Maze). Per-block MXINT4 remains the primary post-training mechanism.

Method (4-bit)	**Sudoku (MLP)**	**Maze (attention)**
Int4 PTQ (eval-only)	0.0%	73.2%
naive Int4 QAT	$71.8\% \pm 0.3$	$82.2\% \pm 0.7$
+ **weight-LSQ**	$81.1\% \pm 0.6$	$81.8\% \pm 1.0$
activation-LSQ only	70.9%	64.9%
FP (reference)	84.1%	84.7%

A.6 Quantization-aware training: full results

Per-block scaling provides a highly effective post-training transition rule. A second, complementary route is QAT. It is critical to separate this cleanly from the PTQ results, as the two interact informatively.

Crucially, QAT *by itself* compensates for much of the per-tensor activation loss. Naive Int4 QAT immediately raises Sudoku performance from 0.0% (PTQ) to $71.8\% \pm 0.3$. It also improves Maze from 73.2% to $82.2\% \pm 0.7$ (Table 2). Therefore, within QAT, much of the activation-scale mismatch is naturally trained away. The residual gain available to a learned quantizer lies almost entirely in the *weights*. Accordingly, we froze our QAT method to weight-LSQ: learning step sizes solely on the weights (Esser et al., 2020), while utilizing plain Int4 activations.

This weight-LSQ method adds $\sim$9 points on Sudoku, jumping from $71.8\% \pm 0.3$ to $81.1\% \pm 0.6$. This gap far exceeds per-seed spread and closes the majority of the remaining FP gap. A targeted decomposition confirms this gain lives in the weights. Applying LSQ solely to activations yields 70.9% (essentially the naive baseline). Conversely, applying it to the weights yields the full 81.1%. Furthermore, an ablation that varies only the bias precision leaves accuracy completely unchanged, showing the gain stems from the learned step size rather than the bias representation.

This picture is remarkably consistent across architectures. On Maze's attention recursion, weight-LSQ is statistically tied with naive QAT and consistent across seeds ($81.8\% \pm 1.0$ vs. $82.2\% \pm 0.7$, both near the FP 84.7% baseline). Activation-LSQ is unstable on attention (it drops performance to 64.9%). The frozen weight-side method explicitly avoids this trap.

In summary, weight-LSQ delivers a significant win where naive QAT struggles (Sudoku/MLP), while remaining stable where naive QAT already succeeds (Maze/attention). Despite these QAT gains, per-block scaling (Table 1) remains the primary post-training mechanism. Both QAT variants train the weight scale: naive QAT uses Brevitas' decoupled per-tensor weight scale, while weight-LSQ replaces it with a single learnable step size initialized from weight statistics (without LSQ's gradient scaling).

Per-iteration alternatives (E4/E5): negative results. The transition-bias theory suggests two natural per-iteration defenses. The first is iteration-decoupled normalization (E4), which gives the reused block per-iteration normalization statistics so the quantized activation distribution remains stationary. The second is cross-iteration distillation (E5), which distills the quantized recursion toward the full-precision trajectory step-by-step (Yao et al., 2022; Zhu et al., 2023). Both naturally target the per-iteration bias.

However, at convergence, neither method improves upon the baseline. On Sudoku, E4 reaches 70.7%, E5 reaches 69.4%, and their combination reaches 72.7%. None improves on the matching single-seed naive-QAT baseline (72.2%; the three-seed mean is $71.8\% \pm 0.3$, Table 2). We therefore report these strictly as negative results. Once the activation scale is handled properly, the per-iteration

Table 3: Static activation-outlier controls do not rescue per-tensor Int4 (Sudoku, 4-bit w+a, qlinear-only coverage, final step; single-seed). SmoothQuant is swept over its migration strength α; QuaRot applies a fixed Hadamard rotation. The best static correction (α=0.5) reaches 0.7%, versus 80.1% for per-block MXINT4. The Hadamard rotation is applied where the input width is a power of two; other layers (e.g. the SwiGLU down projection) use plain per-tensor Int4.

Method (Sudoku, w4a4)	Exact accuracy
per-tensor Int4 (no correction)	0.0%
SmoothQuant α=0.3	0.0%
SmoothQuant α=0.5 (best)	0.7%
SmoothQuant α=0.7	0.4%
SmoothQuant α=0.8	0.03%
SmoothQuant α=0.85	0.0%
SmoothQuant α=0.95	0.0%
Hadamard rotation (QuaRot)	0.0%
per-block MXINT4 (reference)	80.1%

structure requires no special algorithmic handling. Per-block scaling remains the simplest, most robust transition-level rule.

A.7 Static outlier controls do not rescue per-tensor Int4

Table 3 reports the full outlier-control sweep behind Section 4.3. A natural objection is that per-tensor Int4 fails for the usual reason transformer activations are hard to quantize: a few large outlier channels. We test two canonical activation-outlier methods at matched coverage on Sudoku (the maximum-headroom case): SmoothQuant, which migrates activation scale into the weights (Xiao et al., 2023), swept across its migration strength α, and a fixed Hadamard rotation as in QuaRot (Ashkboos et al., 2024). Neither closes the gap: the best SmoothQuant setting (α=0.5) reaches only 0.7% and Hadamard 0.0%, versus MXINT4's 80.1%. This matches our accumulation account. A *static* per-channel scale or *fixed* rotation corrects the outlier distribution only once; after that the divergence stays high and flat across steps (SmoothQuant $242 \to 234$, Hadamard $258 \to 245$) instead of contracting. To succeed, the activation scale must dynamically track local block ranges at each recursive application. These are static corrections. Learned rotations or transformations such as SpinQuant (Liu et al., 2025b) are not tested here.

A.8 Iterated models and transition-aware quantization

The mechanism we study, per-step error compounding across an iterated computation, is well documented in other iterated domains. In autoregressive and diffusion generation, exposure-bias research frames train–test mismatch as per-step errors accumulating across a generated trajectory (Ning et al., 2024; Arora et al., 2022). Self-recovery analyses similarly caution that autoregressive distortions need not grow monotonically (He et al., 2021). A vast body of mitigation work addresses this specific mismatch through input perturbation, scheduled sampling, or shifted timesteps (Ning et al., 2023; Bengio et al., 2015; Li et al., 2024).

A.9 Diffusion quantization and manifold drift

Quantized diffusion models provide the closest theoretical analogue to per-step error accumulation. A diffusion sampler repeatedly queries a single denoising network with shared weights, and uses the result to sequentially update a latent trajectory. Consequently, PTQ perturbs an iterated transition kernel rather than a collection of independent feedforward layers.

PTQ4DM and Q-Diffusion successfully identify this multi-timestep structure as the central obstacle. Because activation and noise-prediction distributions shift with the timestep, a single-step calibration unavoidably induces step-dependent bias (Shang et al., 2023; Li et al., 2023). Temporal Dynamic Quantization and TFMQ-DM make this exact dependence explicit through timestep-conditioned statistics (So et al., 2023; Huang et al., 2024). PTQD provides a full dynamical account: quantization

noise decomposes into correlated and residual components, induces mean deviation, and accumulates over denoising steps to drastically reduce the signal-to-noise ratio (SNR) at late timesteps (He et al., 2023). QNCD and TAC further distinguish single-step disturbance from inter-step accumulation, framing the latter as exposure bias across the sampling trajectory (Chu et al., 2024; Yao et al., 2024).

This represents the continuous generative counterpart of our 4-bit trajectory drift. In a diffusion sampler, systematic quantization shifts the estimated mean update. As a result, the reverse trajectory progressively abandons the full-precision path and drifts away from the data manifold. Recent methods combat this by treating quantization as a backward displacement (Chen et al., 2024), explicitly altering the sampling trajectory (Frumkin & Marculescu, 2025), or applying a marginal-preserving drift adjustment (Ryu et al., 2026).

In a recursive reasoner, the analogue is a refinement path through latent logic space. Our per-tensor Int4 result is the direct logical analogue of this manifold drift. The quantized transition repeatedly injects a biased activation perturbation, pushing the reasoning trajectory away from the full-precision solution manifold instead of allowing it to contract back.

The SNR comparison highlights exactly why this effect is so catastrophic for reasoning models. Diffusion samplers are most fragile late in the reverse chain, when the residual update is small and quantization noise easily dominates microscopic refinements. Recursive reasoners possess an analogous late-stage regime. Their terminal states are strictly constrained by discrete global validity: a Sudoku assignment must perfectly obey all row, column, and box constraints. In continuous image space, modest manifold drift merely manifests as blur or local artifacts. In discrete logical space, the threshold is sharp. A small, systematic displacement changes a single symbol, violating a global constraint and instantly reducing exact accuracy to zero. Both systems accumulate timestep error under iterated shared weights. However, while the perceptual manifold tolerates small geometric deviations, the logical solution manifold is effectively all-or-nothing.

A.10 Mechanism: per-layer quantization bias

Figure 12 details the per-layer calibration data supporting Section 3.3. The relative output error is largest in the recursive down-projection layers. These are exactly the components the recursion re-applies most frequently per forward pass. This localizes where the largest transition perturbations enter.

Coherence diagnostic. We run the collapsing static per-tensor Int4 model and its FP reference in lockstep for 20 evaluation batches (global batch size 512 on Sudoku and 256 on Maze). At recursion step t, we compute the latent difference $e_t^h = h_t^q - h_t^{\mathrm{fp}}$ per example, averaged over answer tokens. We report the cosine between the example means $\overline{e_t^h}$ and $\overline{e_{t+1}^h}$ at adjacent steps, and the cross-example coherence $\|\overline{e_t^h}\|/\overline{\|e_t^h\|}$ (overlines denote the mean over examples). The dominant latent-divergence direction is the first principal component of the per-step mean vectors $\{\overline{e_t^h}\}$; layerwise mean output errors are projected onto this direction. This yields the cross-step cosines 0.990/0.972 and cross-example coherence of approximately 0.9 reported in Section 4.3.

A.11 Dynamical-systems perspective: full statement

The contraction-versus-drift phenomenon admits a lightweight, conditional mathematical characterization. For a fixed input x and a closed neighbourhood U containing the trajectory region visited by the recursion, we use only two properties of the per-step perturbation ϵ_Q from Equation (1), its magnitude and its trajectory mean:

$$\sup_{s\in U}\|\epsilon_Q(s,x)\| \le \varepsilon, \qquad \boldsymbol{\mu}_Q(x) \coloneqq \frac{1}{T}\sum_{t=0}^{T-1}\epsilon_Q(s_t^q, x). \tag{4}$$

Here, ε bounds the magnitude over U. The transition F_θ is one full supervision step, which composes the $\sim$21 applications of the block f_θ within that step (336 over 16 steps). The contraction modulus L and the error ϵ_Q are therefore defined at this macro-step level, and ϵ_Q is the compound error of the quantized layers inside one step. Crucially, $\boldsymbol{\mu}_Q(x)$ isolates the systematic, nonzero-mean component of the perturbation *along input x's specific trajectory*. (This is an average over steps, which drives

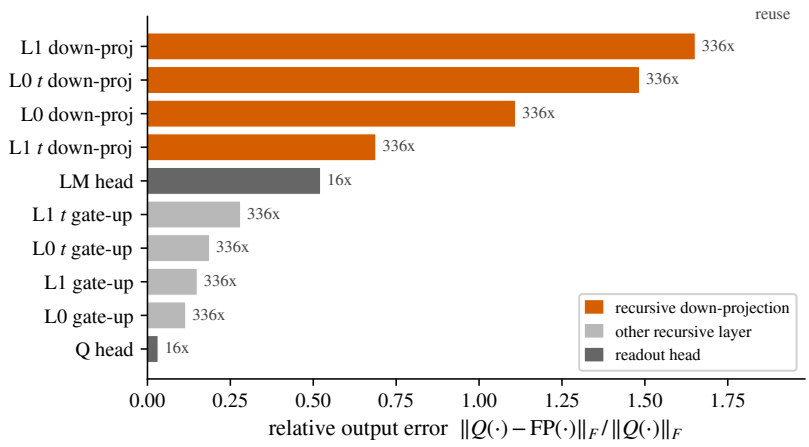


Figure 12: Reused transition layers dominate quantization error. Per-layer relative output error $\|Q(\cdot) - \mathrm{FP}(\cdot)\|_F / \|Q(\cdot)\|_F$ of each linear layer (quantized weights and activations vs. FP, Frobenius norm over the full output, averaged over the 6720 recursive or 320 readout calls) for a calibrated Int4 TRM on Sudoku. Values above 1 mean the error norm exceeds the surviving quantized output. The largest errors occur in recursive down-projection layers that are re-applied $336\times$ per forward pass, compared with $16\times$ for the readout heads. This localizes the transition components where any systematic quantization component is repeatedly injected.

the per-input displacement. It is not an input-pooled average, which could falsely cancel out coherent biases pointing in different directions for different inputs).

A real quantizer is piecewise-constant and therefore not Lipschitz continuous at its rounding boundaries. The statement below therefore assumes only the magnitude bound and requires no continuity of the quantized map. Section 3.3 measures the empirical layerwise analogue of $\boldsymbol{\mu}_Q$.

Empirically, the full-precision recursion is highly contractive near a fixed point s^*. Its residuals decay dramatically over time (e.g., on EqR, the FP residual falls $27.5 \to 16.2$ with depth; at 8-bit, the divergence in Figure 1 rapidly shrinks). Consequently, we *model* F_θ as a contraction with modulus $L < 1$ near s^*. This is supported by the measured decay, though we do not claim it as formally proven.

Proposition 1 (Local ultimate-boundedness model; conditional)**.** *Fix an input x, and let $F_\theta(\cdot, x)$ be contractive with modulus $L < 1$ on the closed ball $U = \bar{B}_r(s^*)$ around its fixed point s^*, with $\|\epsilon_Q(s, x)\| \le \varepsilon$ on U. If $\varepsilon/(1-L) \le r$, then U is invariant under $\widetilde{F} = F_\theta + \epsilon_Q$, and every quantized trajectory started in U satisfies*[3]

$$\limsup_{t\to\infty} \|s_t^q - s^*\| \le \frac{\varepsilon}{1-L}. \tag{5}$$

Remark (informal). Under the further idealization that F_θ is linearized to a fixed Jacobian J along the trajectory ($\|J\| \le L$) and the per-step perturbation carries a constant mean $\boldsymbol{\mu}_Q(x)$, the trajectory error acquires a coherent component of order $\|\boldsymbol{\mu}_Q(x)\|/(1-L)$. Fluctuations of comparable size that are zero-mean and *independent across steps* act purely through their variance and produce no deterministic displacement in expectation.

The feasibility condition $\varepsilon/(1-L) \le r$ separates two regimes. When ε is small relative to the contraction margin, quantized iterates stay in a small ball around s^*. This is the 8-bit regime (Int8 $\approx$ FP). Per-tensor Int4, however, represents the high-error regime, where the condition fails and the proposition no longer applies. The observed divergence growth signifies a perturbation that either violently shifts the local attractor, pushes the trajectory entirely outside the locally contractive region, or outright destroys local contraction. This is consistent with the EqR residual sticking at $\sim$110

[3]For $s \in U$, $\|\widetilde{F}(s,x) - s^*\| \le \|F_\theta(s,x) - F_\theta(s^*,x)\| + \|\epsilon_Q(s,x)\| \le Lr + \varepsilon \le r$, so U is invariant. Unrolling $\|s_{t+1}^q - s^*\| \le L\|s_t^q - s^*\| + \varepsilon$ gives the bound. Continuity of ϵ_Q is not needed, so the quantized map may settle into a small cycle rather than a fixed point. In the equilibrium limit this matches the displacement-versus-margin bound of Li et al. (2026a).

(versus FP $\sim$16) and the exploding 4-bit divergence seen in Figure 1. In the per-tensor Int4 regime, the answer-state divergence d_t^z levels off near 270, consistent with iterates settling in a displaced region, while the answer-logit divergence d_t^y keeps climbing through step 16. Since ℓ_t is a linear readout of h_t, this growth at near-constant d_t^z can arise either from the answer-state error shifting toward directions the readout amplifies or from the readout's own quantized weights acting on a changing state. We do not separate the two.

Why a systematic bias is the culprit. Let $e_t = s_t^q - s_t^{\mathrm{fp}}$ represent the trajectory error. While the iterates remain in U, the local L-contraction of F_θ provides the magnitude bound $\|e_{t+1}\| \leq L\|e_t\| + \|\epsilon_Q(s_t^q, x)\|$. Unrolling this with $\|\epsilon_Q\| \leq \varepsilon$ yields:

$$\|e_t\| \;\leq\; L^t\|e_0\| + \frac{1-L^t}{1-L}\,\varepsilon \;\xrightarrow[t\to\infty]{}\; \frac{\varepsilon}{1-L}. \tag{6}$$

This bound fundamentally conflates a coherent bias with a zero-mean fluctuation. Both simply contribute ε to the right-hand side. The difference lies in their direction over time. Linearizing F_θ along the trajectory to a fixed Jacobian J ($\|J\| \leq L$) gives $e_{t+1} \approx Je_t + \epsilon_t$. Write $\epsilon_t = \boldsymbol{\mu}_Q(x) + \xi_t$. We model the forcing as near-constant, with small ξ_t, which is consistent with the nearly constant direction of the measured trajectory error (cross-step cosine 0.97–0.99, Section A.10). Unrolling gives

$$e_t = J^t e_0 + (I - J^t)(I - J)^{-1}\boldsymbol{\mu}_Q(x) + \textstyle\sum_{k<t} J^{t-1-k}\xi_k.$$

The first term vanishes, so the error converges deterministically to the displacement $(I - J)^{-1}\boldsymbol{\mu}_Q(x)$, of magnitude at most $\|\boldsymbol{\mu}_Q(x)\|/(1 - L)$, plus a small residual. By contrast, a stochastic perturbation that is zero-mean and independent across steps produces zero displacement in expectation, and only its variance accumulates to a bounded steady state.

Because the recursive reasoner repeatedly applies the exact same quantized transition with the exact same parameters, a nonzero $\boldsymbol{\mu}_Q(x)$ is continuously injected in a consistent direction. This generates a deterministic displacement. Zero-mean fluctuations do not. This bias-versus-noise separation explains why injecting zero-mean noise at test time fails to rescue Int4 accuracy (Section 5.3). Per-block scaling reduces both the magnitude ε and the systematic bias $\|\boldsymbol{\mu}_Q(x)\|$, thereby reducing displacement; empirically, MXINT4 returns to contracting behavior. Because the contraction radius r is not estimated, this observation, rather than the reductions in ε and $\boldsymbol{\mu}_Q$ alone, supports the contraction-regime classification.

Scope. Proposition 1 remains a *conditional local model*. We do not assert that TRM or EqR are globally contractive. The contraction assumption is derived directly from empirical residual decay. Furthermore, two specific idealizations are present. First, the literal piecewise-constant map need not admit an exact fixed point, and its iterates may settle into a small cycle. The proposition covers this case because it bounds where the iterates end up rather than asserting convergence to a point, and the empirical plateaus indicate exactly this kind of convergence into a small neighborhood. Second, we do not directly estimate the contraction radius r. We measure the empirical analogues of ε (relative error), L (residual decay), and $\boldsymbol{\mu}_Q$ (layerwise mean), but the regime split $\varepsilon/(1 - L) \leq r$ is inferred through observed behavioral drift rather than checked mathematically. We therefore present this model as strong qualitative corroboration rather than a strict quantitative fit. Formulating a full, non-idealized nonlinear perturbation theory for discrete reasoners remains exciting future work.

A.12 ARC-AGI TRANSFER: FULL TABLE

Full pass@2 results supporting Section 4.6. The cross-architecture summary is also visualized in panel (c) of Figure 2.

A.13 EXPERIMENTAL CONFIGURATION

Datasets and metrics. *Sudoku-Extreme* (Wang et al., 2025; Jolicoeur-Martineau, 2025) comprises hard 9×9 Sudoku puzzles encoded as length-81 token sequences (one token per cell); we evaluate on

[4]Full-coverage MXFP4 is omitted because it is not coverage-matched to MXINT4 and so would not be an apples-to-apples comparison; both MX entries here use identical matched linear-layer weight+activation coverage.

Table 4: Both quantization routes transfer to ARC-AGI (pass@2). Per-tensor Int4-PTQ gives 0% on both datasets, while two independent routes reach the FP-level point estimate: per-block scaling at post-training (MXINT4, no retraining) and quantization-aware training (naive Int4-QAT). On ARC-AGI-2, 4-bit MXINT4 matches the FP point estimate and lands above 8-bit per-tensor Int8 (single-seed, and ARC-AGI-2 scores amount to only a few of its 120 tasks). MX entries use matched linear-layer coverage.[4]

Quantizer	Scaling	ARC-AGI-1	ARC-AGI-2
FP (reference)	n/a	45.25%	6.25%
Int8	per-tensor, 8-bit (PTQ)	41.6%	2.9%
Int4	per-tensor, 4-bit (PTQ)	0.0%	0.0%
MXFP4	per-block float (PTQ)	42.9%	5.4%
MXINT4	**per-block integer (PTQ)**	**44.0%**	**6.25%**
naive Int4-QAT	per-tensor (QAT)	45.9%	6.25%
+ weight-LSQ	learned weights (QAT)	44.6%	4.86%

the full 422,786-puzzle test split (the test-time stochastic-inference sweeps of Section 5.3 use a fixed 8,192-puzzle subset) (models are trained with 1000 augmentations per puzzle). *Maze-Hard* comprises 30×30 grid mazes (length-900 sequences) that require a valid path from start to goal. For both, *exact accuracy* is the fraction of test instances solved *completely* correctly (all 81 cells, or the entire path), whereas *cell accuracy* is the fraction of individual cells/positions correct, a finer-grained signal that can stay high even when no full solution is produced (e.g. a near-miss Sudoku grid). *ARC-AGI-1* and its harder 2025 successor *ARC-AGI-2* are open-ended abstraction-and-reasoning benchmarks of few-shot grid-transformation tasks with no fixed output size; following the recursive-reasoning line of work (Jolicoeur-Martineau, 2025) we report *pass@2* (a test input counts as solved if either of two attempts, produced by test-time-augmentation voting, matches the target). We evaluate the full public evaluation sets (400 ARC-AGI-1 and 120 ARC-AGI-2 tasks). Following the TRM evaluator, a task with several test inputs receives fractional credit equal to the fraction of its test inputs solved, and the score is the mean over tasks. ARC-AGI-2 scores are low (FP: 6.25%, i.e. 7.5 task-equivalents), so a single task (0.83 points) is material, and we read those numbers as point estimates rather than a calibrated ordering. The trajectory divergences d_t^z and d_t^y of Equation (2) are Frobenius norms over the answer-region tokens and channels, averaged over examples. d_t^z is measured on the answer-region recurrent state (z_H in the TRM code) and d_t^y on the answer logits read out from it.

Unless stated otherwise, results utilize the public Sudoku-Extreme and Maze-Hard benchmarks alongside TRM models trained using the publicly released codebase. (The tied FP baseline is 84.1% on Sudoku and 84.7% on Maze; the untied control FP baseline is 69.8% on Sudoku). The untied control is the best of three saved checkpoints by test accuracy.

EqR evaluations use the released Equilibrium Reasoner Sudoku-Extreme checkpoint (Huang et al., 2026) paired with a `bf16`-consistent fake-quantizer. (This quantizer is FP-neutral, verified at an accuracy of 86.4%). The recursion uses `halt_max_steps` $= 16$. The dose–response sweep specifically varies the inner L-cycles across $\{1, 2, 4, 6, 8\}$ on Sudoku and $\{1, 2, 4, 8\}$ on Maze.

Quantizer coverage across protocols. Per-step trajectory diagnostics (Figure 1 and the dynamic/granularity controls of Section 4.3) quantize the linear layers of the recursive block and keep all other modules in full precision (single seed). The format ladder of Table 1 uses a slightly broader integer coverage with three calibration seeds. Endpoint accuracies therefore differ slightly between the two protocols (e.g. Sudoku Int8 84.2% in Figure 1 vs. $84.0\% \pm 0.1$ in the ladder; Maze per-tensor Int4 77.7% vs. $73.5\% \pm 0.4$). MXFP4, which goes through Brevitas calibration and bias correction, differs most (58.4% in Figure 4 vs. $72.7\% \pm 0.4$ in Table 1); MXINT4 uses neither and gives 80.1% in both. Where coverage differs within a figure or table (the per-tensor integer rows of Table 1, Figure 2, and Figure 4), this is stated. It does not drive the collapse (see below).

Quantization is applied via the configuration interface outlined in the released code. The embedding is held in full precision for the microscaling (MX) formats, the weight-only rows, the floating-point per-tensor rows, and the per-step diagnostics of Figure 1. The integer per-tensor rows (Int4, Int8) of Table 1 use a slightly broader coverage that additionally quantizes the embedding (and, on the attention-based Maze task, the scaled dot-product and RoPE). This does not drive the collapse:

with only the linear layers quantized, per-tensor Int4 still reaches 0.0% on Sudoku (Figure 1). The per-tensor integer and floating-point rows also quantize the SwiGLU input, and on Maze the scaled dot-product attention and RoPE. The ARC per-tensor rows (Table 4) keep the embedding in FP. Calibrated rows (per-tensor integer and floating-point, and MXFP4) collect activation statistics on one global evaluation batch drawn with the calibration seed ($4{,}608$ puzzles on Sudoku, 256 on Maze, 512 in the per-step diagnostics), sharded across four GPUs (each GPU calibrates its own scales on its shard), over the first six recursion steps, followed by Brevitas bias correction. MXINT4 and the dynamic, granularity, and outlier controls compute scales online and use neither. Per-block formats utilize a block size of 32 combined with a power-of-two (E8M0) shared scale, with blocks formed along the contraction (input) dimension for both weights and activations. The block maximum is floored at 10^{-12}, so all-zero blocks are well defined. Full hyperparameter configurations, exact random seeds, and comprehensive per-step logging scripts are provided in the supplementary material.