LOOPY: LOW-BIT QUANTIZATION FRAMEWORK FOR LOOPED LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped language models provide a parameter-efficient way to scale iterative test-time computation by repeatedly executing a shared recurrent core. Post-training quantization (PTQ) can reduce the memory footprint and inference cost of looped language models, but errors introduced by a quantized shared core affect subsequent cores. Among PTQ methods, channel scaling and orthogonal rotations preserve the floating-point computation while producing representations with different quantization quality. We find that quantization configuration candidate rankings can change with recurrent depth, motivating configuration selection at the target deployment depth. However, evaluating every candidate over the full calibration set at this depth is costly. We therefore propose LOOPY, a PTQ framework that formulates shared-core quantization through a recurrent-depth-aware objective, selecting shared low-bit representations by their final prediction loss at the target deployment depth. Channel scaling and orthogonal rotations parameterize the candidate representations. To approximately solve this selection problem efficiently, LOOPY progressively allocates calibration windows to promising candidates while preserving complete target-depth execution, using only forward evaluations. Across eight settings, LOOPY achieves the state-of-the-art results among different baselines. On Ouro-1.4B under W4A4, LOOPY reduces LAMBADA perplexity by 36.5% relative to SpinQuant. Our code is available at `https://anonymous.4open.science/r/Loopy-ICLR2027/`.

1 INTRODUCTION

Looped language models increase effective depth by repeatedly executing a shared recurrent core composed of transformer blocks, enabling iterative computation without storing a separate set of weights for every recurrent step as shown in Figure 1 (Dehghani et al., 2019; Zhu et al., 2025; Saunshi et al., 2025; Geiping et al., 2025). Post-training quantization (PTQ) provides a complementary approach to efficient deployment by reducing the precision of weights and activations, thereby lowering memory requirements and enabling faster inference with low-bit kernels (Xiao et al., 2023; Ashkboos et al., 2024; Liu et al., 2025). Applying PTQ to looped language models therefore offers an effective solution to low-bit computation while preserving prediction quality.

Channel scaling and orthogonal rotations mitigate quantization outliers while preserving the floating-point computation (Xiao et al., 2023; Lin et al., 2024; Ashkboos et al., 2024; Liu et al., 2025). However, these transformations do not uniquely determine a good quantization configuration (Liu et al., 2025).

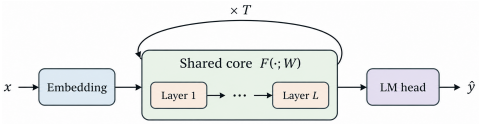


Figure 1: Looped language model.

To preserve parameter sharing, we use the same quantized weights and transformations across recurrent steps (Section 3.1). As hidden states evolve across recurrent steps (Figure 14), quantization errors propagate through subsequent computation. Consequently, smaller local reconstruction error need not imply lower final prediction loss (Appendix C.2). LoopQ also considers recurrent dynamics through loop-aware quantization (Fang et al., 2026). These observations motivate target-depth evaluation, supported by our calibration-depth ab-

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.

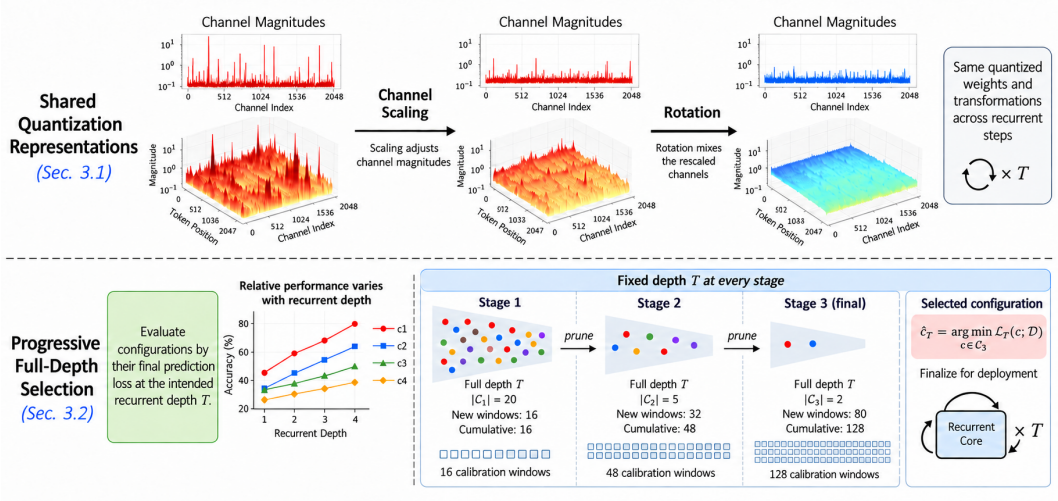


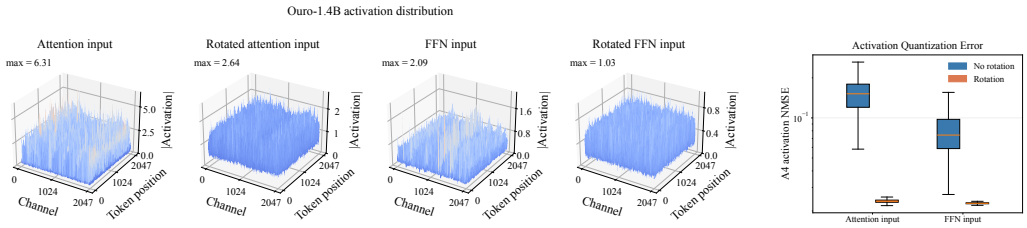
Figure 2: **Loopy method overview.** Top: channel scaling and rotation define quantized representations shared across recurrent steps. Bottom: progressive selection allocates more calibration windows to surviving candidates while preserving the target depth T.

lations (Figure 7). However, evaluating every candidate on all calibration windows at this depth is costly.

Motivated by these observations, we introduce LOOPY, a recurrent-depth-aware PTQ framework for looped language models, as shown in Figure 2. The key idea is to formulate shared-core quantization as representation selection under a target-depth prediction objective: a shared quantized representation is evaluated by its final prediction loss after complete execution at the intended recurrent depth. Channel scaling and orthogonal rotations define the candidate representation space. LOOPY approximately solves the resulting selection problem through progressive full-depth evaluation, keeping the recurrent depth fixed while allocating additional calibration windows to surviving candidates. This procedure uses only forward evaluations, and the selected quantized weights and transformations remain shared across recurrent steps.

We evaluate LOOPY on three representative looped language models: the Ouro series (Zhu et al., 2025), Huginn (Geiping et al., 2025), and Recurrent-Llama (McLeish et al., 2025), under both W4A8 and W4A4 quantization. Our main contributions are summarized as follows.

- We systematically study low-bit quantization in looped language models. We uncover a loop-specific quantization phenomenon: a quantization configuration shared across recurrent steps can exhibit depth-dependent quality because of the recurrent dependency of looped models. This serves as the motivation for our method LOOPY.
- We introduce LOOPY, a post-training quantization framework centered on a recurrent-depth-aware objective for selecting shared low-bit representations. LOOPY parameterizes these representations through channel scaling and orthogonal rotations, and approximately solves the selection problem through progressive full-depth evaluation. This procedure preserves the target recurrent depth while reducing calibration work by allocating additional windows only to surviving candidates.
- We evaluate LOOPY across multiple looped language model families under W4A4 and W4A8 quantization. On Ouro-1.4B under W4A4, LOOPY reduces LAMBADA perplexity by 36.5% relative to the adapted SpinQuant baseline. We further evaluate LOOPY with low-bit KV quantization and show that Q/K Hadamard transforms substantially mitigate the degradation from 4-bit KV quantization.



(a) Activation magnitudes. (b) A4 NMSE.

Figure 3: **Rotation improves activation-side quantization on Ouro-1.4B.** (a) Attention and feed-forward network (FFN) inputs of physical block 22 at recurrent step 4, before and after rotation. Appendix F provides all 24 blocks. (b) Activation normalized mean-squared error (NMSE) distributions without and with rotation, on a logarithmic scale.

2 Motivation

In a looped language model, a shared recurrent core repeatedly updates the hidden state according to

$$h_{t+1} = F(h_t; W), \qquad t = 0, \ldots, T-1,$$

where W denotes the shared weights and T is the recurrent depth. After quantization, the corresponding computation becomes $\hat{h}_{t+1} = \hat{F}_c(\hat{h}_t)$, where c specifies the quantized weights and transformation settings of the core's linear operators. The configuration c is shared across recurrent steps, while hidden states and dynamic activation scales may vary. Appendix G illustrates activation changes across recurrent calls.

2.1 Rotation Mitigates Activation Outliers

We examine how rotation affects activation quantization on Ouro-1.4B. Figure 3(a) shows reduced activation peaks in the displayed block's attention and FFN inputs. We measure A4 reconstruction error using normalized mean-squared error (NMSE), $\|Q(a)-a\|_2^2/\|a\|_2^2$, where Q denotes activation quantization. Figure 3(b) shows lower median NMSE after rotation for both inputs, motivating the shared rotation transforms in Section 3.1.

2.2 Configuration Quality Depends on Recurrent Depth

To examine whether configuration quality depends on recurrent depth, we evaluate 20 rotation–scaling configurations on Ouro-1.4B W4A4 at depths $T = 1, \ldots, 4$. Figure 4 shows changes in their relative WinoGrande performance across depths, with the corresponding rankings provided in Appendix H.

Together with the recurrent error analysis in Appendix C, these observations motivate selecting shared representations by their target-depth prediction loss, formalized in Section 3.

3 Method

Loopy formulates shared-core quantization as recurrent-depth-aware representation selection. Each candidate specifies quantized weights and transformations that remain shared across recurrent steps, while activation distributions evolve with the hidden state. We evaluate each candidate through complete execution at the intended deployment depth T. Given a candidate space $\mathcal{C}$ and calibration windows $\mathcal{D}$, we define the recurrent-depth-aware quantization objective as

$$c_T^\star \in \arg\min_{c \in \mathcal{C}} \mathcal{L}_T(c; \mathcal{D}), \qquad \mathcal{L}_T(c; \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} L_T(c, x), \tag{1}$$

Here, $L_T(c, x)$ denotes the mean next-token negative log-likelihood (NLL) on window x after T recurrent steps. This objective evaluates the quality of a shared quantized representation through its

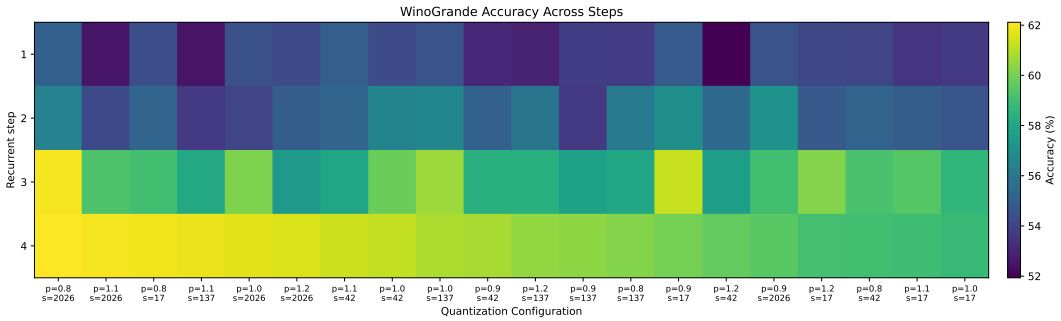


Figure 4: **Accuracy across recurrent depths.** WinoGrande accuracy (%) for 20 configurations of Ouro-1.4B W4A4. Columns identify scaling power p and rotation seed s and are ordered by decreasing accuracy at $T = 4$. Rows show recurrent depth. A shared color scale preserves absolute differences across depths. Appendix H shows the corresponding within-depth rankings.

complete target-depth recurrent computation. Channel scaling and orthogonal rotations parameterize the candidate space, while progressive full-depth selection provides an efficient solution.

3.1 Shared Quantization Representations

Each candidate $c = (s, p, b)$ specifies a rotation seed, a channel-scaling power, and a Hadamard block-size setting. The same configuration determines operator-specific transforms $D_{\ell,p}$ and $R_{\ell,s,b}$, which remain fixed across recurrent calls. For each linear operator ℓ, the paired transforms satisfy

$$xW_\ell^\top = \left(xD_{\ell,p}^{-1}R_{\ell,s,b}\right)\left(W_\ell D_{\ell,p}R_{\ell,s,b}\right)^\top. \tag{2}$$

In exact arithmetic, these transforms preserve the recurrent hidden-state trajectory (Appendix C.1). Quantization generally breaks this equivalence, motivating selection by final prediction quality.

For each recurrent linear operator ℓ, we obtain a positive channel-scaling vector d_ℓ from calibration and construct

$$D_{\ell,p} = \mathrm{diag}\left(\frac{d_\ell^p}{\sqrt{\max(d_\ell^p)\min(d_\ell^p)}}\right), \tag{3}$$

$$R_{\ell,s,b} = S_{\ell,s}^{\mathrm{pre}}\,\mathrm{blkdiag}(H_b,\ldots,H_b)S_{\ell,s}^{\mathrm{post}}, \tag{4}$$

where powers are applied elementwise, H_b is a normalized Hadamard matrix of size b, and $S_{\ell,s}^{\mathrm{pre}}$ and $S_{\ell,s}^{\mathrm{post}}$ are diagonal sign matrices generated deterministically from the seed and operator identity. We use randomized Hadamard rotations (Ashkboos et al., 2024), restricting b to supported powers of two that divide the operator's input dimension. Search varies (s, p, b) while keeping the calibrated scaling vectors fixed.

For each candidate, we quantize the transformed weights and dynamically quantize the transformed inputs, yielding

$$\widehat{f}_{\ell,c}(x) = Q_a\left(xD_{\ell,p}^{-1}R_{\ell,s,b}\right)Q_w\left(W_\ell D_{\ell,p}R_{\ell,s,b}\right)^\top, \tag{5}$$

Here, Q_w and Q_a denote weight and activation quantization. Weight quantization is fixed for each candidate, whereas activation scales are computed dynamically per token. The transforms act locally at linear operators, leaving residual-stream coordinates unchanged and requiring no step-specific weight copies. Figure 5 illustrates these paired transforms in attention and FFN operators.

3.2 Progressive Full-Depth Selection

Using fewer calibration windows estimates the same target-depth objective, whereas reducing recurrent depth changes the computation being evaluated. LOOPY therefore progressively allocates windows to surviving candidates while preserving complete target-depth execution (Figure 2).

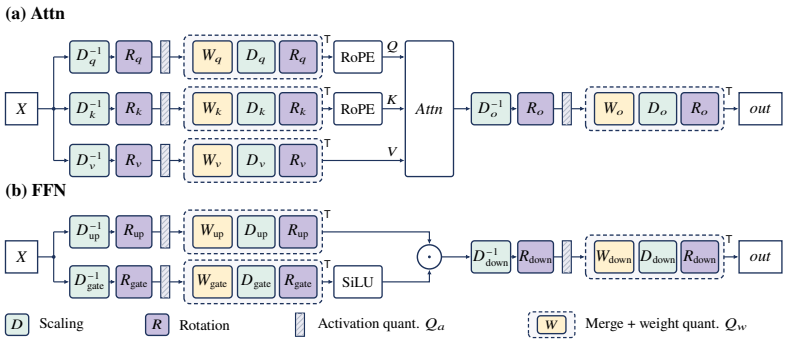


Figure 5: **Operator-wise quantization transforms**. Channel scaling and rotation in attention and FFN, with weight-side transforms merged before quantization and input transforms applied online.

Following successive halving (Jamieson & Talwalkar, 2016), we construct nested calibration subsets $\mathcal{D}_1 \subset \cdots \subset \mathcal{D}_R = \mathcal{D}$ and initialize $\mathcal{C}_1 = \mathcal{C}$. At stage r, all active candidates are evaluated on the same windows through complete T-step execution:

$$\widehat{\mathcal{L}}_r(c) = \frac{1}{|\mathcal{D}_r|} \sum_{x \in \mathcal{D}_r} L_T(c, x), \qquad c \in \mathcal{C}_r. \tag{6}$$

We reuse previously computed losses and evaluate only newly added windows at each stage.

After each intermediate stage, we retain the K_{r+1} candidates with the lowest estimated loss,

$$\mathcal{C}_{r+1} = \mathrm{Lowest}_{K_{r+1}}\left(\mathcal{C}_r; \widehat{\mathcal{L}}_r\right), \qquad r = 1, \ldots, R-1, \tag{7}$$

where the sample budgets and retained candidate counts are specified before search. All candidates within a stage are compared on the same calibration windows, and ties are resolved using a fixed candidate order. As the active set shrinks, additional windows refine the comparison among the remaining configurations. At the final stage, every surviving candidate is evaluated on the full calibration set, and we select

$$\hat{c}_T = \arg\min_{c \in \mathcal{C}_R} \mathcal{L}_T(c; \mathcal{D}). \tag{8}$$

Early elimination may discard the full-search optimum. A sufficient condition for preserving it is that, at each screening stage, the full-set loss gap between the optimum and candidates beyond the retention cutoff exceeds twice the maximum estimation error (Appendix C.5). This condition motivates allocating more windows to refine candidate comparisons while retaining enough candidates to accommodate uncertainty. We assess the resulting approximation empirically through selection regret and prediction quality on separate evaluation data.

Let $n_r = |\mathcal{D}_r|$, with $n_0 = 0$. By reusing previously computed losses, LOOPY requires

$$N_{\mathrm{eval}} = \sum_{r=1}^{R} |\mathcal{C}_r|(n_r - n_{r-1}) \tag{9}$$

candidate–window evaluations, compared with $|\mathcal{C}|\,|\mathcal{D}|$ for exhaustive search. This count measures evaluation work, and wall-clock cost also includes candidate construction and scheduling. At deployment, the selected transformed weights are precomputed and quantized, while input transforms and dynamic activation quantization are applied during execution. Algorithm 1 summarizes the selection procedure. In Algorithm 1, `BuildOrRestore` constructs or restores candidates from fixed calibration bases, and `EvalFullDepth` sums window-level mean NLLs after T recurrent steps.

3.3 TARGET-DEPTH SELECTION ANALYSIS

Recurrent error propagation can reverse candidate rankings across depths, even under shared dynamics (Appendix C.3). We therefore distinguish depth mismatch from finite-window estimation error.

Algorithm 1 Loopy Progressive Full-Depth Selection

Require: $\mathcal{C}$, T, nested subsets $\{\mathcal{D}_r\}_{r=1}^{R}$, retained counts $\{K_r\}_{r=2}^{R}$
$\mathcal{C}_1 \leftarrow \mathcal{C}$, $\mathcal{D}_0 \leftarrow \varnothing$, $S(c) \leftarrow 0$ for all $c \in \mathcal{C}$
for $r = 1, \ldots, R$ **do**
  for $c \in \mathcal{C}_r$ **do**
    $\widehat{F}_c \leftarrow \textsc{BuildOrRestore}(c)$
    $S(c) \leftarrow S(c) + \textsc{EvalFullDepth}(\widehat{F}_c, T, \mathcal{D}_r \setminus \mathcal{D}_{r-1})$
    $\widehat{\mathcal{L}}_r(c) \leftarrow S(c)/|\mathcal{D}_r|$
  end for
  if $r < R$ **then**
    $\mathcal{C}_{r+1} \leftarrow \textsc{KeepLowest}(\mathcal{C}_r, \widehat{\mathcal{L}}_r, K_{r+1})$
  end if
end for
$\hat{c}_T \leftarrow \textsc{Best}(\mathcal{C}_R, \widehat{\mathcal{L}}_R)$
return $\textsc{BuildOrRestore}(\hat{c}_T)$

Fix the candidate pool $\mathcal{C}$ and calibration set $\mathcal{D}$, and abbreviate $\mathcal{L}_t(c; \mathcal{D})$ as $\mathcal{L}_t(c)$. Define the pairwise depth mismatch

$$\beta_{\tau,T} = \max_{c,c' \in \mathcal{C}} \left| [\mathcal{L}_\tau(c) - \mathcal{L}_\tau(c')] - [\mathcal{L}_T(c) - \mathcal{L}_T(c')] \right|. \tag{10}$$

This measures changes in candidate comparisons rather than candidate-independent loss shifts.

Selection-regret bound. Consider nested top-K screening of fixed candidates, with stage $r < R$ evaluated at depth τ_r. Suppose $\max_{c \in \mathcal{C}_r} |\widehat{\mathcal{L}}_r(c) - \mathcal{L}_{\tau_r}(c)| \le \varepsilon_r$, and final selection minimizes $\mathcal{L}_T$ over the survivors using all of $\mathcal{D}$. Then

$$\begin{aligned} \mathrm{Reg}_T(\hat{c}_T) &:= \mathcal{L}_T(\hat{c}_T) - \min_{c \in \mathcal{C}} \mathcal{L}_T(c) \\ &\le \max_{r<R} \big(\beta_{\tau_r,T} + 2\varepsilon_r \big). \end{aligned} \tag{11}$$

The bound does not require a unique optimum or its survival. If the last global optimum is discarded, every candidate retained at that stage satisfies the bound, as do all subsequent survivors.

For LOOPY, $\tau_r = T$ and $\beta_{T,T} = 0$, giving

$$\mathrm{Reg}_T(\hat{c}_T) \le 2 \max_{r<R} \varepsilon_r. \tag{12}$$

Full-depth screening removes depth mismatch, leaving window-estimation error. This conditional bound applies to the fixed calibration set. Our schedule does not certify ε_r, and the bound guarantees neither held-out generalization nor superiority at equal compute. Appendix C.4 proves the regret bound, and Appendix C.5 gives a sufficient condition for exact optimum recovery.

4 EXPERIMENTS

4.1 SETUP

Models and evaluation. We evaluate Ouro-1.4B and Ouro-2.6B at recurrent depth $T = 4$ (Zhu et al., 2025), Huginn-0125 at $T = 32$ (Geiping et al., 2025), and Recurrent-Llama-3.2-1B at $T = 16$ (McLeish et al., 2025). We report perplexity on WikiText-2 (Merity et al., 2016) and LAMBADA (Paperno et al., 2016), and zero-shot accuracy on HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2020), LAMBADA, ARC-Challenge (Clark et al., 2018), and MMLU (Hendrycks et al., 2021). Avg. denotes the unweighted mean of the five task accuracies.

Quantization and baselines. We evaluate W4A4 and W4A8 against recurrent adaptations of AWQ, SmoothQuant, QuaRot, SpinQuant, and FlatQuant (Lin et al., 2024; Xiao et al., 2023; Ashkboos et al., 2024; Liu et al., 2025; Sun et al., 2024). LOOPY and all evaluated baselines uses symmetric group-32 quantization and symmetric per-token activation quantization. Baseline-specific quantization settings and calibration procedures are detailed in Appendix B. LoopQ comparison uses the same Loopy results as the main table and published LoopQ results (Fang et al., 2026).

Table 1: **W4A8 and W4A4 results on looped language models.** WT2 and LMB denote WikiText-2 and LAMBADA perplexity (↓); Avg. is mean task accuracy (%, ↑). Detailed accuracy results are shown in Tables 5 and 6. Bold indicates the best low-bit result per model and precision.

	Ouro-2.6B			Ouro-1.4B			Huginn-0125			Recurrent-Llama-3.2-1B		
Method	WT2	LMB	Avg.	WT2	LMB	Avg.	WT2	LMB	Avg.	WT2	LMB	Avg.
FP ref.	10.51	4.23	69.65	12.03	5.29	64.66	13.21	7.30	50.42	20.72	17.23	38.63
W4A8												
AWQ	11.46	5.22	66.45	13.64	7.68	60.51	14.14	8.85	47.37	28.16	50.70	35.10
SmoothQuant	19.44	15.20	56.15	53.27	134.60	37.78	—	—	25.82	243.77	—	26.40
QuaRot	11.76	5.30	66.76	14.25	8.29	59.09	—	—	25.61	26.03	31.15	36.60
SpinQuant	12.07	5.56	66.21	14.18	7.52	60.42	—	—	24.75	25.96	28.42	36.75
FlatQuant	14.52	8.14	62.42	26.55	23.50	46.21	—	—	24.43	86.73	—	28.22
Loopy	**10.97**	**4.46**	**68.64**	**12.77**	**6.00**	**63.00**	**13.68**	**7.98**	**49.21**	**24.76**	**26.97**	**37.23**
W4A4												
AWQ	340.35	—	26.0	645.60	—	25.6	168.39	117.12	34.2	—	—	25.0
SmoothQuant	—	—	24.5	—	—	25.2	—	—	25.3	—	—	25.5
QuaRot	15.27	9.73	56.9	19.16	22.18	44.7	—	—	25.3	48.08	139.25	31.5
SpinQuant	16.62	8.18	60.9	17.45	12.86	52.1	—	—	25.0	**41.03**	**102.34**	**32.5**
FlatQuant	—	—	24.6	—	—	25.0	—	—	25.0	—	—	25.1
Loopy	**12.35**	**6.20**	**64.37**	**14.73**	**8.17**	**57.75**	**28.09**	**36.36**	**37.32**	42.08	110.83	32.0

— denotes perplexity greater than 10^3.

Calibration and selection. LOOPY selects shared rotation and scaling configurations using progressive full-depth evaluation (Section 3). Calibration bases remain fixed across candidates; nested window subsets and retained candidate counts are specified before search. All LOOPY results in Table 1 and the full-results appendix are obtained using configurations selected by the progressive full-depth procedure described in Section 3.

Implementation. Main accuracy results use quantization and dequantization in BF16 with KV caching disabled. We implement packed low-bit inference in Triton (Tillet et al., 2019), with hardware benchmarks described in Section 4.5.

4.2 MAIN RESULTS

W4A8. In Table 1, LOOPY achieves the lowest WikiText-2 and LAMBADA perplexity and the highest available average accuracy among the evaluated low-bit methods on all four checkpoints.

W4A4. LOOPY achieves the lowest perplexity on both datasets for the two Ouro checkpoints and Huginn-0125. Average accuracy exceeds the evaluated low-bit baselines on these three checkpoints. Recurrent-Llama exhibits greater degradation under W4A4 than W4A8, compared SpinQuant under W4A4, which is one of our work's limitation. Its inherited Llama pre-normalization structure differs from the normalization schemes used by Ouro and Huginn (McLeish et al., 2025; Zhu et al., 2025; Geiping et al., 2025), potentially affecting the propagation of quantization perturbations through recurrent execution.

Comparison with LoopQ. All LoopQ results are taken from its original paper (Fang et al., 2026), while Loopy results are reused from Table 1. Table 2 shows that Loopy achieves lower W4A4 WikiText-2 perplexity and higher five-task average accuracy on both Ouro checkpoints.

4.3 SELECTION QUALITY–CALIBRATION COST TRADE-OFF

We compare LOOPY with full-depth uniform selection at evaluation depth T, shallow progressive selection at $T/2$, and shallow-to-full selection with evaluation depths $T/4 \to T/2 \to T$. Experiments use Ouro-1.4B W4A4 ($T = 4$) and Huginn-0125 W4A8 ($T = 32$). Full-depth uniform selection allocates a finite calibration budget equally across candidates, while the other strategies allocate additional calibration windows according to their respective selection procedures. We measure the

Table 2: W4A4 comparison with LoopQ. †: published results (Fang et al., 2026); Avg. Acc. averages the five tasks in Table 1.

	WT2 PPL ↓		Avg. Acc. (%) ↑	
Model	LoopQ†	LOOPY	LoopQ†	LOOPY
Ouro-1.4B	15.81	**14.73**	57.08	**57.75**
Ouro-2.6B	15.89	**12.35**	58.13	**64.37**

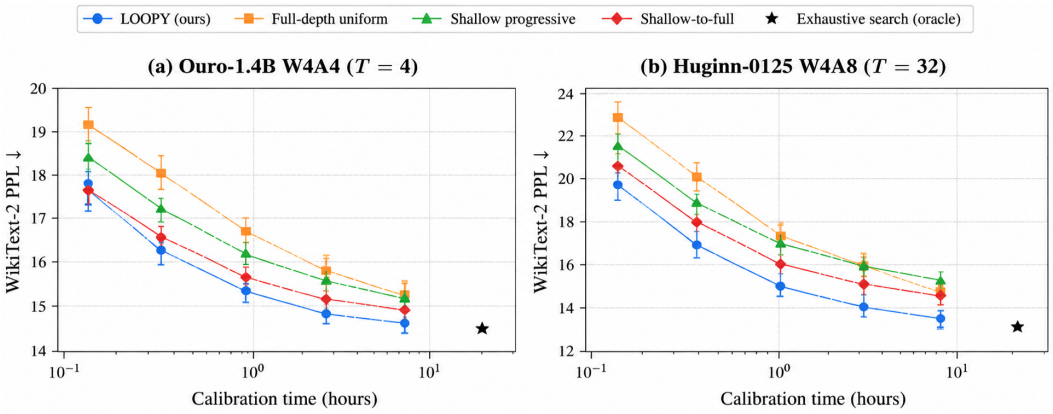


Figure 6: **Selection quality–calibration cost trade-off.** WikiText-2 perplexity versus calibration time for (a) Ouro-1.4B W4A4 ($T = 4$) and (b) Huginn-0125 W4A8 ($T = 32$). Black stars denote exhaustive-search references. Lower perplexity is better.

end-to-end calibration cost of each strategy, including base calibration, candidate construction or restoration, candidate screening, and final model construction.

Within each model, candidate representations and quantization settings are fixed. We repeat selection using $N = 10$ independently sampled calibration sets, with all strategies sharing the same ordered calibration data within each run. Each strategy is evaluated under multiple predefined calibration budgets, and the selected configurations are evaluated at deployment depth T on held-out WikiText-2 using token-normalized perplexity with 2048-token windows for convenience. Figure 6 presents the relationship between calibration cost and selection quality, where lower perplexity indicates better selected configurations. Figure 6 shows that LOOPY achieves a favorable quality–cost trade-off across both models. Compared with full-depth exhaustive search, LOOPY reaches comparable selection quality with substantially lower calibration cost. On Ouro-1.4B W4A4 and Huginn-0125 W4A8, LOOPY reduces end-to-end calibration cost by 4.2× and 3.5×, respectively.

4.4 KV CACHE QUANTIZATION

We evaluate 4-bit and 8-bit KV quantization on Ouro-1.4B at $T = 4$ with fixed Loopy W4A4 and W4A8 configurations, optionally applying post-RoPE Q/K Hadamard transforms following SpinQuant (Liu et al., 2025). We report WikiText-2 perplexity and average accuracy over four downstream tasks, with the full evaluation protocol and per-task results provided in Appendix A.1.

As shown in Table 3, Q/K Hadamard transforms reduce the relative perplexity increase from KV4 quantization from 6.56% to 3.41% under W4A4 and from 4.38% to 2.19% under W4A8. They also improve average accuracy by 1.40 percentage points over KV4 under W4A4.

4.5 DEPLOYMENT EFFICIENCY AND VALIDATION

Decoding throughput. We evaluate Ouro-1.4B at a recurrent depth of $T = 4$ on an NVIDIA H20 GPU under W4A4 and W4A8 quantization. With batch size 4 and an input length of 2048

Table 3: KV quantization results. ΔPPL is relative to BF16 KV within each W/A setting; Avg. Acc. averages four tasks. H denotes the Q/K Hadamard transform.

W/A	KV setting	PPL ↓	ΔPPL (%)	Avg. Acc. (%) ↑
W4A4	BF16 KV	11.4174	—	54.5266
	BF16 KV+H	11.4139	−0.03	54.4818
	KV8	11.4109	−0.06	54.1423
	KV4	12.1660	+6.56	51.9853
	KV4+H	11.8062	+3.41	53.3883
W4A8	BF16 KV	9.3425	—	61.7154
	BF16 KV+H	9.3422	−0.004	61.7387
	KV8	9.3457	+0.03	61.9927
	KV4	9.7513	+4.38	61.4155
	KV4+H	9.5470	+2.19	61.4134

Table 4: **Decoding throughput on NVIDIA H20 (tokens/s).** B/input denotes batch size and input length. Protocol details and additional configurations appear in Appendix D.

	W4A4		W4A8	
Method	B1/128	B4/2048	B1/128	B4/2048
QuaRot	9.08*	35.14*	8.80	34.75
SpinQuant	9.16*	35.85*	9.14	35.78
Loopy (online)	10.01	40.93	10.22	41.17
Loopy (offline)†	**11.21**	**45.45**	**11.37**	**45.83**

† Fully absorbable transform configuration.
* Numerical-mismatch diagnostic runs.

tokens, online Loopy achieves about 41 and 41 decoding tokens per second (tokens/s), respectively, compared with 35 and 35 tokens/s for the official SpinQuant implementation. Table 4 summarizes the results, while Appendix D details the timing protocol and implementation differences.

5 Related Work

Looped language models. Looped Transformers increase effective depth through repeated application of shared parameters, as explored by Universal Transformers and subsequent looped language models (Dehghani et al., 2019; Saunshi et al., 2025; Geiping et al., 2025; Zhu et al., 2025; Jeddi et al., 2026). Parameter sharing reduces the number of distinct weights required to represent a deep computation, motivating studies of memory-efficient recurrent architectures (Vendrell et al., 2026).

Post-training quantization. Existing methods reduce quantization error through weight reconstruction or channel scaling, as exemplified by GPTQ, AWQ, and SmoothQuant (Frantar et al., 2022; Lin et al., 2024; Xiao et al., 2023). Complementing these approaches, rotation-based methods redistribute outliers, with QuaRot using structured orthogonal transformations and SpinQuant learning rotations through an end-to-end quantization objective (Ashkboos et al., 2024; Liu et al., 2025).

Recurrence-aware optimization. Repeated execution of the recurrent core produces evolving hidden states, allowing quantization errors to propagate across recurrent steps. LoopQ addresses this problem through loop-aware transformations and trajectory-aware calibration (Fang et al., 2026).

6 Conclusion

We presented Loopy, a PTQ framework that targets looped language models. By selecting shared low-bit representations according to their final target-depth prediction loss, Loopy aligns quantization selection with recurrent deployment and makes this selection practical through progressive full-depth evaluation. Experiments show benefits varying across architectures.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript organization, language editing, literature checking, and the development and execution of code for manuscript preparation, verification, and supplementary experiments. The authors take full responsibility for the accuracy and integrity of the paper, including its claims, experimental results, references, and supplementary materials.

REPRODUCIBILITY STATEMENT

The supplementary source includes the numerical inputs, figure-generation code, source hashes, and a claim-to-evidence ledger. Appendix B describes the evaluation protocol, while Appendix E documents the supporting records for quantization evaluation, diagnostic analysis, and packed-execution validation.

REFERENCES

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. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.

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

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

Rui Fang, Hsi-Wen Chen, and Ming-Syan Chen. Loopq: Quantization for recursive transformers. *arXiv preprint arXiv:2605.16343*, 2026.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. *Advances in Neural Information Processing Systems*, 38: 41340–41391, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.03300.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pp. 240–248. PMLR, 2016. URL https://proceedings.mlr.press/v51/jamieson16.html.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. *arXiv preprint arXiv:2602.11451*, 2026.

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

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*, 2025.

Sean McLeish, Ang Li, John Kirchenbauer, Dayal Singh Kalra, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Jonas Geiping, Tom Goldstein, and Micah Goldblum. Teaching pretrained language models to think deeper with retrofitted recurrence. *arXiv preprint arXiv:2511.07384*, 2025. URL https://arxiv.org/abs/2511.07384.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. URL https://arxiv.org/abs/1609.07843.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, 2016. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020. doi: 10.1609/aaai.v34i05.6399. URL https://ojs.aaai.org/index.php/AAAI/article/view/6399.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, volume 2025, pp. 14855–14881, 2025.

Yuxuan Sun, Ruikang Liu, Haoli Bai, Han Bao, Kang Zhao, Yuening Li, Jiaxin Hu, Xianzhi Yu, Lu Hou, Chun Yuan, et al. Flatquant: Flatness matters for llm quantization. *arXiv preprint arXiv:2410.09426*, 2024.

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

Victor Conchello Vendrell, Arnau Padres Masdemont, Niccolò Grillo, Jordi Ros-Giralt, Arash Behboodi, and Fabio Valerio Massoli. Memory-efficient looped transformer: Decoupling compute from memory in looped language models. *arXiv preprint arXiv:2605.07721*, 2026.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, et al. Scaling latent reasoning via looped language models. *arXiv preprint arXiv:2510.25741*, 2025.

APPENDIX

A FULL RESULTS

Tables 5 and 6 report the full results. The remaining metrics retain their original evaluation. Avg. is the mean of five task accuracies (%). FP ref. provides full-precision context. Collapsed results are retained. W4A8 averages reproduce the supplied results.

Table 5: Complete W4A8 results. HS: HellaSwag; WG: WinoGrande; LMB: LAMBADA. †: adapted baselines.

Model	Method	WT2 PPL	LMB PPL	HS	WG	LMB Acc.	ARC-C	MMLU	Avg.
Ouro-2.6B	FP ref.	10.51	4.23	76.45	69.53	70.08	58.36	73.83	69.65
	AWQ	11.4603	5.2171	74.71	67.80	64.22	55.89	69.63	66.45
	SmoothQuant†	19.4442	15.2011	68.29	61.96	45.37	50.26	54.85	56.15
	QuaRot†	11.7643	5.3048	74.13	68.43	64.70	56.83	69.69	66.76
	SpinQuant†	12.0654	5.5639	74.28	66.93	63.13	57.76	68.93	66.21
	FlatQuant†	14.5179	8.1443	71.83	66.69	56.12	54.10	63.35	62.42
	Loopy	10.97	4.46	75.94	70.48	68.76	56.14	71.86	68.64
Ouro-1.4B	FP ref.	12.03	5.29	71.69	66.06	65.05	52.30	68.20	64.66
	AWQ	13.6390	7.6843	68.43	63.85	56.72	51.71	61.85	60.51
	SmoothQuant†	53.2679	134.5964	52.44	56.91	19.44	34.73	25.40	37.78
	QuaRot†	14.2539	8.2905	68.71	64.01	55.85	48.46	58.39	59.09
	SpinQuant†	14.1781	7.5221	69.31	63.77	58.35	50.00	60.66	60.42
	FlatQuant†	26.5550	23.5027	62.40	59.04	38.85	45.56	25.22	46.21
	Loopy	12.77	6.00	70.38	64.33	62.12	52.05	66.13	63.00
Huginn-0125	FP ref.	13.21	7.30	66.68	61.64	55.35	37.54	30.90	50.42
	AWQ	14.1421	8.8549	63.78	59.98	51.56	35.67	25.84	47.37
	SmoothQuant†	5.32×10^4	7.76×10^6	26.59	51.30	0.00	26.28	24.95	25.82
	QuaRot†	6.41×10^3	5.81×10^5	26.90	52.33	0.02	25.85	22.97	25.61
	SpinQuant†	6.65×10^3	5.03×10^5	26.56	49.33	0.04	23.98	23.83	24.75
	FlatQuant†	2.70×10^4	3.76×10^6	25.97	48.15	0.00	25.17	22.86	24.43
	Loopy	13.68	7.98	64.71	61.17	53.81	36.69	29.67	49.21
Recurrent-Llama-3.2-1B	FP ref.	20.72	17.23	47.51	52.49	40.81	29.35	23.00	38.63
	AWQ	28.1567	50.7029	44.58	52.41	27.17	28.24	23.11	35.10
	SmoothQuant†	243.7745	1.53×10^4	32.75	49.09	3.30	23.63	23.22	26.40
	QuaRot†	26.0300	31.1539	43.77	53.83	32.06	28.41	24.95	36.60
	SpinQuant†	25.9630	28.4209	45.00	52.88	34.06	28.58	23.20	36.75
	FlatQuant†	86.7286	1.40×10^3	36.31	49.33	6.71	25.68	23.08	28.22
	Loopy	24.76	26.97	45.88	52.80	34.76	28.75	23.94	37.23

Table 6: Complete W4A4 results. Avg. is the arithmetic mean of the five displayed task accuracies (%). Large perplexities are shown in scientific notation.

Model	Method	WT2 PPL	LMB PPL	HS	WG	LMB Acc.	ARC-C	MMLU	Avg.
Ouro-2.6B	FP ref.	10.51	4.23	76.45	69.53	70.08	58.36	73.83	69.65
	AWQ	340.35	1.35×10^{4}	30.5	49.7	1.9	24.5	23.4	26.00
	SmoothQuant	1.01×10^{4}	1.25×10^{6}	26.3	48.4	0.0	24.7	23.2	24.52
	QuaRot	15.27	9.73	69.1	60.9	52.1	49.2	53.4	56.94
	SpinQuant	16.62	8.18	70.4	63.1	56.2	53.5	61.2	60.88
	FlatQuant	1.57×10^{4}	3.77×10^{6}	26.2	47.9	0.0	24.9	23.8	24.56
	Loopy	12.35	6.20	72.87	65.59	62.35	55.38	65.64	64.37
Ouro-1.4B	FP ref.	12.03	5.29	71.69	66.06	65.05	52.30	68.20	64.66
	AWQ	645.60	7.02×10^{4}	29.7	50.6	0.4	23.4	23.7	25.56
	SmoothQuant	2.22×10^{4}	9.83×10^{6}	26.7	49.1	0.0	25.4	24.8	25.20
	QuaRot	19.16	22.18	62.0	55.5	39.0	40.7	26.2	44.68
	SpinQuant	17.45	12.86	64.4	59.9	47.9	42.1	46.2	52.10
	FlatQuant	4.69×10^{4}	8.32×10^{6}	26.2	49.0	0.0	24.1	25.8	25.02
	Loopy	14.73	8.17	67.70	61.48	55.81	47.70	56.07	57.75
Huginn-0125	FP ref.	13.21	7.30	66.68	61.64	55.35	37.54	30.90	50.42
	AWQ	168.39	117.12	37.3	50.4	31.4	27.5	24.4	34.20
	SmoothQuant	8.46×10^{4}	7.60×10^{6}	26.3	48.5	0.0	26.2	25.4	25.28
	QuaRot	1.82×10^{4}	1.09×10^{6}	26.3	50.7	0.0	26.1	23.6	25.34
	SpinQuant	1.62×10^{4}	1.14×10^{6}	26.5	50.0	0.0	24.1	24.3	24.98
	FlatQuant	6.22×10^{4}	5.75×10^{6}	26.2	48.4	0.0	26.3	24.3	25.04
	Loopy	28.09	36.36	47.18	51.46	33.61	29.61	24.76	37.32
Recurrent-Llama-3.2-1B	FP ref.	20.72	17.23	47.51	52.49	40.81	29.35	23.00	38.63
	AWQ	6.37×10^{3}	1.14×10^{6}	26.1	49.6	0.0	25.1	24.1	24.98
	SmoothQuant	5.59×10^{4}	1.34×10^{7}	26.2	51.1	0.0	25.0	25.4	25.54
	QuaRot	48.08	139.25	38.0	52.7	17.8	25.4	23.8	31.54
	SpinQuant	41.03	102.34	39.7	50.5	21.5	27.2	23.6	32.50
	FlatQuant	1.04×10^{4}	5.18×10^{6}	26.0	51.0	0.0	25.5	23.2	25.14
	Loopy	42.08	110.83	37.68	52.17	20.64	26.70	23.07	32.0

Table 7: Per-task accuracy (%) for KV quantization. H denotes the Q/K Hadamard transform.

W/A	KV setting	ARC-C	WinoGrande	LAMBADA	HellaSwag
W4A4	BF16 KV	44.20	62.51	45.97	65.43
	BF16 KV+H	44.97	60.85	46.59	65.51
	KV8	45.65	58.96	46.48	65.48
	KV4	40.78	58.64	44.34	64.17
	KV4+H	40.96	61.88	45.68	65.04
W4A8	BF16 KV	51.02	64.56	61.48	69.80
	BF16 KV+H	50.85	64.88	61.56	69.67
	KV8	50.77	65.75	61.77	69.69
	KV4	51.54	65.11	59.71	69.30
	KV4+H	50.51	64.56	61.05	69.53

A.1 KV Quantization Protocol and Per-task Results

We evaluate ByteDance/Ouro-1.4B at $T = 4$, freezing the existing Loopy W4A4 and W4A8 weights and transforms without additional search. The controls are BF16 KV, BF16 KV+H, KV8, KV4, and KV4+H. Evaluation uses full-sequence teacher forcing with `use_cache=False`. K and V are quantized and dequantized for attention computation; these experiments assess prediction quality rather than cached decoding performance.

Following SpinQuant (Liu et al., 2025), the $+H$ variants apply the same normalized H_{128} to Q and K after RoPE and before the attention dot product. K is quantized after this transform; Q receives no additional quantization, and V receives no additional rotation. Otherwise, K is quantized directly after RoPE. V is quantized after its projection and head partitioning. Both K and V use dynamic asymmetric min–max quantization with a separate scale and zero-point for each token and head (128 dimensions), including zero in the range and applying no clipping. KV8 and KV4 quantize both tensors to 8 and 4 bits, respectively. Statistics and quantization arithmetic use FP32, and dequantized outputs are cast to BF16.

WikiText-2 raw test text is concatenated and partitioned into non-overlapping blocks of up to 2048 tokens, retaining the final shorter block. The first token of each block is excluded from loss. We aggregate NLL weighted by the number of valid prediction tokens and report $\mathrm{PPL} = \exp(\mathrm{NLL})$, over 149 blocks and 304,829 valid prediction tokens. This token-normalized protocol differs from the word-normalized 4096-token protocol of the main results. Average accuracy is the unweighted mean over ARC-Challenge, WinoGrande, LAMBADA, and HellaSwag, excluding MMLU.

Table 7 reports per-task accuracy. Under W4A4, Q/K Hadamard transforms improve all four task accuracies relative to KV4, whereas under W4A8, task-level gains and losses largely offset each other.

A.2 Lower-Bit Quantization on Ouro Models

Under W3A4, Loopy achieves WikiText-2 perplexities of 23.77 and 17.65 on Ouro-1.4B and Ouro-2.6B, respectively. W3A3 substantially increases perplexity to 1653.88 and 361.85, revealing severe degradation at this more aggressive precision.

Table 8: **WikiText-2 perplexity under lower-bit quantization.** All quantized results use Loopy. Lower is better.

Model	BF16	W4A4	W3A4	W3A3
Ouro-1.4B	12.03	14.73	23.77	1653.88
Ouro-2.6B	10.51	12.35	17.65	361.85

B Evaluation Protocol

Model identity and scope. We evaluate four checkpoints at depths 4, 4, 32, and 16: Ouro-2.6B, Ouro-1.4B, Huginn-0125, and Recurrent-Llama-3.2-1B. The Ouro sizes share a family. Checkpoint names denote physical parameter count, not unrolled depth. The paper's adapters quantize eligible dense Transformer linears across the prelude, recurrent core, and coda, while Loopy transforms only recurrent-core linears, retaining the AWQ anchor elsewhere. Embeddings, normalization, LM heads, and early-exit gates remain unquantized.

Quantization recipes. LOOPY and AWQ both use symmetric group-wise W4 quantization with a weight group size of 32, together with dynamic symmetric per-token A8 or A4 quantization. In the W4A4 comparison in Table 1, our SmoothQuant adaptation uses group-wise W4 quantization with a weight group size of 32 and dynamic A4 quantization, and our QuaRot adaptation (Ashkboos et al., 2024) combines graph rotations with GPTQ (Frantar et al., 2022), our SpinQuant adaptation (Liu et al., 2025) learns orthogonal transforms before GPTQ and uses symmetric A4. Our FlatQuant adaptation (Sun et al., 2024) uses layerwise affine/diagonal transformations and clipping with W4 RTN. These recurrent-architecture adaptations use the same weight group size of 32. Activation symmetry and calibration compute are not matched across methods. Collapsed adaptations do not establish a failure of every implementation of the original method.

LoopQ comparison. Table 2 reuses the W4A4 Loopy results from Table 1 for Ouro-1.4B and Ouro-2.6B at $T = 4$. LoopQ perplexities and task accuracies are taken from Table 1 of Fang et al. (2026); its average accuracy is computed as the unweighted mean of HellaSwag, WinoGrande, LAMBADA, ARC-Challenge, and MMLU. Quantization settings are aligned, while each method retains its own calibration procedure.

Likelihood and downstream tasks. The full test splits are evaluated without a sample cap, with batch size one. The main WikiText-2 results report word-normalized perplexity with a context length of 4096 tokens, aligned with the LoopQ evaluation protocol. The calibration-depth ablation and numerical-validation experiments use token-normalized perplexity on non-overlapping 2048-token windows, excluding the first position of each window. LAMBADA perplexity and final-word accuracy follow the evaluator's task definition. HellaSwag, WinoGrande, ARC-Challenge, and MMLU use zero-shot answer scoring. MMLU covers all 57 subjects. The paper runtime asserts the requested number of recurrent steps and sets `use_cache=False`. Huginn's request-dependent state initialization is seeded deterministically so that resumed shards use the same state.

B.1 Selection Quality and Calibration Cost

The comparison uses Ouro-1.4B W4A4 at $T = 4$ and Huginn-0125 W4A8 at $T = 32$. Shallow uniform ($T/2$), full-depth uniform (T), shallow-to-full screening ($T/4 \to T/2 \to T$), and LOOPY share the candidate pool, calibrated scaling bases, and quantization recipe within each model. Selection regret is evaluated on the full selection set relative to the same-pool exhaustive optimum at target depth. End-to-end calibration time includes base calibration, candidate construction or restoration, screening, and final model construction, and is normalized by exhaustive search. Figure 6 presents the selection quality–cost comparison.

B.2 Calibration Depth at Fixed Deployment Depth

The following ablations retain the original token-normalized WikiText-2 perplexity protocol with 2048-token windows.

We vary calibration depth while fixing deployment depth for each model. Ouro-2.6B W4A8 and Ouro-1.4B W4A4 use $T_{\text{cal}} \in \{1, 2, 4\}$ and $T_{\text{deploy}} = 4$; Huginn-0125 W4A8 uses $T_{\text{cal}} \in \{8, 16, 32\}$ and $T_{\text{deploy}} = 32$. Within each configuration, the candidate pool and selection windows remain fixed. We reselect candidates by rollout NLL at each calibration depth, and evaluate all selected candidates at the fixed deployment depth.

As shown in Figure 7, increasing calibration depth lowers deployment perplexity in all three configurations. From the shallowest tested depth to the deployment depth, perplexity decreases from 8.1184 to 8.0367 on Ouro-2.6B, from 12.83 to 11.41 on Ouro-1.4B, and from 9.98 to 9.79 on Huginn. Recorded selection times increase from 96.7 to 296.8 seconds, 62.4 to 171.8 seconds, and

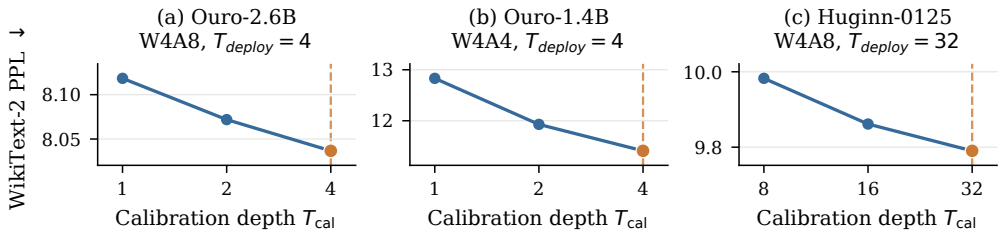


Figure 7: **Calibration depth at fixed deployment depth.** Token-normalized WikiText-2 perplexity with 2048-token windows for (a) Ouro-2.6B W4A8, (b) Ouro-1.4B W4A4, and (c) Huginn-0125 W4A8. Orange markers indicate matched calibration and deployment depths. Each configuration has one run; no error bars are shown. Vertical scales differ across panels.

847.3 to 3164.7 seconds, respectively. Matching deployment depth therefore gives the best perplexity among the tested depths in these runs, at increased selection cost. Additional runs are needed to assess variability.

C Shared Transformations and Recurrent Error Propagation

C.1 Equivalence of Shared Transformations

Proposition 1 (Equivalence under recurrent execution) *Consider a recurrent core F with linear operators $xW_\ell^\top + b_\ell$. For each transformed operator ℓ, let D_ℓ be a positive diagonal matrix and R_ℓ an orthogonal matrix. Replace its input and weight by*

$$\widetilde{x} = xD_\ell^{-1}R_\ell, \qquad \widetilde{W}_\ell = W_\ell D_\ell R_\ell,$$

while retaining its bias and all other operations. These transformations are fixed and shared across recurrent steps. Before quantization and in exact arithmetic, the resulting core $\widetilde{F}$ produces the same hidden-state trajectory as F from the same initial state,

$$\widetilde{h}_t = h_t, \qquad t = 0, \ldots, T,$$

for any finite recurrent depth T. An unchanged output readout consequently produces identical final logits.

Proof. For any input x, orthogonality of R_ℓ and invertibility of D_ℓ give

$$\begin{aligned}\widetilde{x}\widetilde{W}_\ell^\top + b_\ell &= xD_\ell^{-1}R_\ell(W_\ell D_\ell R_\ell)^\top + b_\ell \\ &= xD_\ell^{-1}R_\ell R_\ell^\top D_\ell W_\ell^\top + b_\ell \\ &= xW_\ell^\top + b_\ell.\end{aligned}$$

Each transformed operator thus retains its original output coordinates and values. Since all other operations remain unchanged, $\widetilde{F}(h) = F(h)$ for every input h. Starting from $\widetilde{h}_0 = h_0$, induction yields

$$\widetilde{h}_{t+1} = \widetilde{F}(\widetilde{h}_t) = F(h_t) = h_{t+1}.$$

The final logits are identical under the same output readout. □

The equivalence is algebraic and requires neither contractivity nor differentiability, but does not imply bitwise agreement in finite precision. Quantization can produce representation-dependent perturbations, trajectories, and prediction losses, so pre-quantization equivalence neither guarantees equal quantized quality nor justifies selection by local reconstruction error alone. The next subsection analyzes recurrent perturbation propagation.

C.2 Exact Loss Decomposition and Recurrent Transport

Fix a calibration window x and its initialization, and suppress the dependence of the core maps on x. Let $h_{t+1} = F(h_t)$ and $\hat{h}_{t+1,c} = \widehat{F}_c(\hat{h}_{t,c})$, with $\hat{h}_{0,c} = h_0$. The candidate map $\widehat{F}_c$ is fixed and reused

at every recurrent step. Let $\ell_x(h)$ denote the window-mean next-token NLL after a fixed output computation, so that $L_T(c,x) = \ell_x(\hat{h}_{T,c})$. Candidates and the reference share initialization and non-recurrent computation, which need not be fully unquantized. The reference therefore isolates recurrent-core changes and need not equal the fully floating-point baseline. All losses are assumed finite.

Exact target-depth loss decomposition. Define the local perturbation at the state actually visited by the quantized model and the loss after a reference continuation as

$$z_{t,c} = F(\hat{h}_{t,c}), \qquad e_{t,c} = \widehat{F}_c(\hat{h}_{t,c}) - F(\hat{h}_{t,c}), \qquad V_t^{(T)}(h) = \ell_x\big(F^{T-t}(h)\big), \tag{13}$$

where $0 \le t \le T$ for $V_t^{(T)}$ and F^0 is the identity map.

Proposition 2 (Exact loss decomposition) *For any finite target depth T,*

$$L_T(c,x) - \ell_x(h_T) = \sum_{t=0}^{T-1} \Big[V_{t+1}^{(T)}(z_{t,c} + e_{t,c}) - V_{t+1}^{(T)}(z_{t,c}) \Big]. \tag{14}$$

No differentiability, contractivity, or small-perturbation assumption is required.

Proof. By definition, $V_t^{(T)}(h) = V_{t+1}^{(T)}(F(h))$ and $\hat{h}_{t+1,c} = z_{t,c} + e_{t,c}$. Each summand therefore equals $V_{t+1}^{(T)}(\hat{h}_{t+1,c}) - V_t^{(T)}(\hat{h}_{t,c})$. Summing telescopes to

$$V_T^{(T)}(\hat{h}_{T,c}) - V_0^{(T)}(h_0) = \ell_x(\hat{h}_{T,c}) - \ell_x(F^T(h_0)),$$

which proves the identity. □

The identity requires no quantizer derivatives and applies to discontinuous quantized maps. It evaluates perturbations along the quantized trajectory through reference continuations, rather than by their norms alone, and does not imply monotone error growth.

Optional smooth interpretation. For fixed x, c, t, T, additionally suppose $V_{t+1}^{(T)}$ is continuously differentiable on a neighborhood of the segment $\{z_{t,c} + s e_{t,c} : 0 \le s \le 1\}$, with an $M_{t,c}^{(T)}$-Lipschitz gradient along that segment. The fundamental theorem of calculus gives

$$\begin{aligned} V_{t+1}^{(T)}(z_{t,c} + e_{t,c}) - V_{t+1}^{(T)}(z_{t,c}) &= \Big\langle \nabla V_{t+1}^{(T)}(z_{t,c}), e_{t,c} \Big\rangle + \rho_{t,c,x}^{(T)}, \\ \rho_{t,c,x}^{(T)} &= \int_0^1 \Big\langle \nabla V_{t+1}^{(T)}(z_{t,c} + s e_{t,c}) - \nabla V_{t+1}^{(T)}(z_{t,c}), e_{t,c} \Big\rangle \, ds. \end{aligned} \tag{15}$$

The Lipschitz condition implies $|\rho_{t,c,x}^{(T)}| \le M_{t,c}^{(T)} \|e_{t,c}\|_2^2/2$. Consequently,

$$\begin{aligned} L_T(c,x) - \ell_x(h_T) &= \sum_{t=0}^{T-1} \Big\langle \nabla V_{t+1}^{(T)}(z_{t,c}), e_{t,c} \Big\rangle + R_{T,c,x}, \\ |R_{T,c,x}| &\le \frac{1}{2} \sum_{t=0}^{T-1} M_{t,c}^{(T)} \|e_{t,c}\|_2^2. \end{aligned} \tag{16}$$

When the chain rule applies,

$$\nabla V_{t+1}^{(T)}(z) = \big[D(F^{T-t-1})(z) \big]^\top \nabla \ell_x\big(F^{T-t-1}(z)\big).$$

The first-order effect depends on perturbation direction and reference continuation, with a remainder not assumed small at aggressive precision. A quantized output computation may violate the smoothness assumptions, whereas Equation (14) remains valid without them.

Relation to local state-error transport. Writing $\delta_{t,c} = \hat{h}_{t,c} - h_t$, a differentiable reference map admits the local expansion

$$\delta_{t+1,c} = J_t \delta_{t,c} + e_{t,c} + r_{t,c}, \qquad J_t = \left. \frac{\partial F}{\partial h} \right|_{h_t}, \tag{17}$$

Here, $r_{t,c}$ is the Taylor remainder. Without controlling it, this recurrence alone gives no quantitative error guarantee for the quantized model.

Connection to depth mismatch. For the fixed calibration set $\mathcal{D}$, define

$$\mathcal{L}_t^{\mathrm{ref}} = \frac{1}{|\mathcal{D}|}\sum_{x\in\mathcal{D}} \ell_x(h_t(x)), \qquad \Delta_t(c) = \mathcal{L}_t(c;\mathcal{D}) - \mathcal{L}_t^{\mathrm{ref}}.$$

Averaging Equation (14) over windows decomposes $\Delta_t(c)$ exactly. If $d_{\tau,T}(c) = \mathcal{L}_\tau(c;\mathcal{D}) - \mathcal{L}_T(c;\mathcal{D})$, then

$$d_{\tau,T}(c) = \mathcal{L}_\tau^{\mathrm{ref}} - \mathcal{L}_T^{\mathrm{ref}} + \Delta_\tau(c) - \Delta_T(c).$$

The reference-only term is independent of c. Hence the depth mismatch in Equation (10) satisfies

$$\beta_{\tau,T} = \max_{c\in\mathcal{C}}\{\Delta_\tau(c) - \Delta_T(c)\} - \min_{c\in\mathcal{C}}\{\Delta_\tau(c) - \Delta_T(c)\}.$$

This expresses depth mismatch through depth-dependent perturbation effects, without guaranteeing small $\beta_{\tau,T}$ or a ranking reversal when $\beta_{\tau,T} > 0$.

C.3 A Contractive Ranking-Reversal Construction

We give an explicit construction showing that contractivity and smaller local perturbation norms do not guarantee agreement between shallow- and target-depth candidate rankings.

Shared contractive dynamics. Let $h_0 = 0$, $F(h) = Ah$, and

$$A = \mathrm{diag}(0.9, 0.1), \qquad b_a = (0.5, 0)^\top, \qquad b_b = (0, 1)^\top, \qquad \widehat{F}_c(h) = Ah + \eta b_c, \quad \eta > 0.$$

Each candidate $c \in \{a, b\}$ reuses the same map at every recurrent step. Since the additive perturbation cancels between inputs and $\|A\|_2 = 0.9$,

$$\|\widehat{F}_c(h) - \widehat{F}_c(h')\|_2 = \|F(h) - F(h')\|_2 = \|A(h - h')\|_2 \le 0.9\|h - h'\|_2.$$

At every state, $\widehat{F}_c(h) - F(h) = \eta b_c$. Thus $\|e_{t,a}\|_2 = 0.5\eta < \eta = \|e_{t,b}\|_2$, both on and off the realized trajectories.

Explicit trajectories and ranking reversal. The reference trajectory is zero. Iterating the candidate recurrence gives

$$\hat{h}_{T,c} = \eta\sum_{j=0}^{T-1} A^j b_c.$$

For binary logits $(0, u(h))$, where $u(h) = h_1 + h_2$, and the first class as target, the NLL is $\ell(h) = \log(1 + \exp(u(h)))$. It is strictly increasing in $u(h)$, and

$$u(\hat{h}_{T,a}) = \frac{\eta}{2}\sum_{j=0}^{T-1} 0.9^j, \qquad u(\hat{h}_{T,b}) = \eta\sum_{j=0}^{T-1} 0.1^j. \tag{18}$$

In particular,

Depth	$u(\hat{h}_{T,a})/\eta$	$u(\hat{h}_{T,b})/\eta$	Lower NLL
1	0.5	1	a
4	1.7195	1.111	b

Writing $L_t(c) = \ell(\hat{h}_{t,c})$, we obtain $L_1(a) < L_1(b)$ but $L_4(a) > L_4(b)$. Both inequalities are strict for every $\eta > 0$, including arbitrarily small perturbations. Candidate a perturbs the more slowly decaying coordinate, so its smaller local errors can have a larger effect after repeated execution.

Selection failure without window-estimation error. For $\mathcal{C} = \{a, b\}$, a singleton calibration set, and target depth $T = 4$, exact depth-1 screening that retains one candidate selects a and eliminates the target-depth optimum b, despite zero window-estimation error. Exact target-depth evaluation of the survivor cannot recover b, yielding

$$\mathrm{Reg}_4(a) = \log(1 + \exp(1.7195\eta)) - \log(1 + \exp(1.111\eta)) > 0. \tag{19}$$

For $\eta = 1$, the target-depth NLLs are approximately 1.884299 and 1.395600, giving regret 0.488699. These are values of the analytical construction, not language-model measurements.

Scope. This abstract construction rules out guarantees based solely on contractivity and local-error norms. It does not specify quantization settings realizing these affine maps or assert contractivity of the evaluated language-model cores. Nor does it imply that shallow screening always fails or that full-depth screening is uniformly superior at equal compute.

C.4 Proof of the Selection-Regret Bound

To prove Equation (11), fix a finite nonempty candidate pool $\mathcal{C}$ and calibration set $\mathcal{D}$, with fixed representations, calibration bases, and window initializations. Write $\mathcal{L}_t(c) = \mathcal{L}_t(c; \mathcal{D})$ and assume finite losses.

Depth mismatch. Set $d_{\tau,T}(c) = \mathcal{L}_\tau(c) - \mathcal{L}_T(c)$. The pairwise mismatch in Equation (10) satisfies

$$\beta_{\tau,T} = \max_{c,c' \in \mathcal{C}} |d_{\tau,T}(c) - d_{\tau,T}(c')| = \max_{c \in \mathcal{C}} d_{\tau,T}(c) - \min_{c \in \mathcal{C}} d_{\tau,T}(c). \tag{20}$$

The maximum pairwise absolute difference equals the range, so candidate-independent loss offsets leave $\beta_{\tau,T}$ unchanged, and $\beta_{T,T} = 0$.

Screening protocol and assumptions. Start with $\mathcal{C}_1 = \mathcal{C}$. At each stage $1 \le r < R$, score the surviving candidates at depth $\tau_r \in \{1, \dots, T\}$ on a nonempty subset $\mathcal{D}_r \subseteq \mathcal{D}$:

$$\widehat{L}_r(c) = \frac{1}{|\mathcal{D}_r|} \sum_{x \in \mathcal{D}_r} L_{\tau_r}(c, x), \qquad c \in \mathcal{C}_r.$$

Retain the K_{r+1} lowest-scoring candidates, where $1 \le K_{r+1} \le |\mathcal{C}_r|$, with fixed tie breaking. Candidate sets are nested and eliminated candidates are never reintroduced. Assume that each screening stage satisfies

$$\max_{c \in \mathcal{C}_r} |\widehat{L}_r(c) - \mathcal{L}_{\tau_r}(c)| \le \varepsilon_r. \tag{21}$$

The final stage evaluates all survivors on the complete calibration set at the target depth and returns $\hat{c}_T \in \arg\min_{c \in \mathcal{C}_R} \mathcal{L}_T(c)$. For $R \ge 2$, we will show

$$0 \le \mathrm{Reg}_T(\hat{c}_T) := \mathcal{L}_T(\hat{c}_T) - \min_{c \in \mathcal{C}} \mathcal{L}_T(c) \le \max_{1 \le r < R} (\beta_{\tau_r,T} + 2\varepsilon_r). \tag{22}$$

Proof. Let $L_T^\star = \min_{c \in \mathcal{C}} \mathcal{L}_T(c)$ and $\mathcal{C}_T^\star = \{c \in \mathcal{C} : \mathcal{L}_T(c) = L_T^\star\}$. If $\mathcal{C}_R \cap \mathcal{C}_T^\star \neq \varnothing$, exact final selection returns a global minimizer and the regret is zero. No uniqueness assumption is needed.

Otherwise, define the stage at which the last global minimizers are discarded:

$$r^\dagger = \min\{r : \mathcal{C}_{r+1} \cap \mathcal{C}_T^\star = \varnothing\}.$$

Since $\mathcal{C}_1 = \mathcal{C}$ and the sets are nested, $1 \le r^\dagger < R$, and some $c^\star \in \mathcal{C}_{r^\dagger} \cap \mathcal{C}_T^\star$ exists. Every retained candidate $d \in \mathcal{C}_{r^\dagger+1}$ satisfies $\widehat{L}_{r^\dagger}(d) \le \widehat{L}_{r^\dagger}(c^\star)$, including under tied scores. Applying Equation (21) to both candidates yields

$$\mathcal{L}_{\tau_{r^\dagger}}(d) - \mathcal{L}_{\tau_{r^\dagger}}(c^\star) \le 2\varepsilon_{r^\dagger}.$$

By the definition of pairwise depth mismatch,

$$\begin{aligned} \mathcal{L}_T(d) - L_T^\star &\le \mathcal{L}_{\tau_{r^\dagger}}(d) - \mathcal{L}_{\tau_{r^\dagger}}(c^\star) + \beta_{\tau_{r^\dagger},T} \\ &\le 2\varepsilon_{r^\dagger} + \beta_{\tau_{r^\dagger},T}. \end{aligned}$$

This holds for every candidate retained at that stage. All later candidate sets are subsets of $\mathcal{C}_{r^\dagger+1}$, so it also holds for $\hat{c}_T$. Thus

$$\mathrm{Reg}_T(\hat{c}_T) \le \beta_{\tau_{r^\dagger},T} + 2\varepsilon_{r^\dagger}. \tag{23}$$

Taking the maximum over screening stages proves the upper bound. Nonnegativity follows from $\hat{c}_T \in \mathcal{C}$. □

Full-depth consequence. For LOOPY, $\tau_r = T$ at every screening stage, so $\beta_{\tau_r,T} = 0$ and

$$\mathrm{Reg}_T(\hat{c}_T) \le 2 \max_{1 \le r < R} \varepsilon_r,$$

recovering Equation (12). If all global minimizers are eliminated, the refined bound is $2\varepsilon_{r^\dagger}$. The stage-maximum bound also implies the weaker sum bound because all terms are nonnegative. Accumulation across stages is unnecessary for nested top-K screening: later elimination cannot escape the loss bound already satisfied by every retained candidate. If $R = 1$, there is no screening and exact full-set selection has zero regret.

Assumptions and limits. This deterministic guarantee is conditional on the fixed-pool protocol and stagewise error bounds above. It requires neither a unique optimum nor a positive gap, and assumes no window independence, unbiasedness, bounded NLL, or distributional model for screening errors. The schedule does not certify ε_r, and the proof supplies no concentration or sample-complexity guarantee. The result concerns the fixed pool and calibration set, not all quantizers or held-out generalization. Nonzero depth mismatch neither lower-bounds regret nor necessarily reverses rankings, and removing it does not guarantee superiority at equal compute. The next proposition gives sufficient conditions for exact optimum recovery.

C.5 OPTIMAL-CANDIDATE PRESERVATION

Complementing the regret bound in Appendix C.4, we give a sufficient condition for exact recovery of a unique optimum under full-depth screening. Retain its fixed candidate pool, calibration set, and representations, and write $L(c) = \mathcal{L}_T(c; \mathcal{D})$.

Proposition 3 (Optimal-candidate preservation) *Assume that L has a unique minimizer $c^\star$ over the finite nonempty pool $\mathcal{C}$. Start with $\mathcal{C}_1 = \mathcal{C}$. At each screening stage $1 \le r < R$, score candidates at the target depth T using estimates $\widehat{L}_r(c)$ and retain the K_{r+1} lowest-scoring candidates, with fixed tie breaking and $1 \le K_{r+1} \le |\mathcal{C}_r|$. Eliminated candidates are not reintroduced. Suppose*

$$\max_{c \in \mathcal{C}_r} |\widehat{L}_r(c) - L(c)| \le \varepsilon_r. \tag{24}$$

Let $L_{r,(j)}$ be the j-th smallest full-set loss among candidates in $\mathcal{C}_r$, counted with multiplicity. At every stage that eliminates candidates, assume

$$L_{r,(K_{r+1}+1)} - L(c^\star) > 2\varepsilon_r. \tag{25}$$

Then $c^\star$ survives all screening stages. Exact final evaluation of the survivors on $\mathcal{D}$ at depth T returns $c^\star$, and the selection regret is zero.

Proof. Initially, $c^\star \in \mathcal{C}_1$. Suppose $c^\star \in \mathcal{C}_r$. If $K_{r+1} = |\mathcal{C}_r|$, no candidate is discarded and the claim is immediate. Otherwise, write $K = K_{r+1}$. For every candidate satisfying $L(c) \ge L_{r,(K+1)}$, Equations (24) and (25) imply

$$\widehat{L}_r(c) \ge L(c) - \varepsilon_r > L(c^\star) + \varepsilon_r \ge \widehat{L}_r(c^\star).$$

Consequently, any candidate whose estimated loss is no greater than that of $c^\star$ must have full-set loss strictly below $L_{r,(K+1)}$. There are at most K such candidates, including $c^\star$ itself. Thus at most $K-1$ other candidates can precede $c^\star$, even with tied estimated scores, and $c^\star$ is retained. Induction proves survival through every stage. Since the unique global minimizer belongs to $\mathcal{C}_R$, exact final evaluation returns it. □

Interpretation and relation to the regret bound. The gap compares the optimum with the first loss outside the retention budget, not necessarily the second-best candidate. For a fixed candidate set, retaining more candidates weakens the separation requirement. More windows may improve estimation but need not monotonically reduce realized error. The strict gap guarantees survival under any fixed tie-breaking rule.

Unlike the regret bound in Appendix C.4, exact recovery here requires uniqueness and separation. The regret bound remains applicable without either condition, including after all optima are discarded. Failure of Equation (25) does not imply elimination of the optimum, since the condition is sufficient, not necessary.

Table 9: Complete H20 decoding throughput (tokens/s, ↑). The last two rows add online rotations to Loopy (offline).

	W4A4		W4A8	
Method	B1/128	B4/2048	B1/128	B4/2048
QuaRot	9.08*	35.14*	8.80	34.75
SpinQuant	9.16*	35.85*	9.14	35.78
Loopy (online)	10.01	40.93	10.22	41.17
Loopy (offline)	**11.21**	45.45	**11.37**	**45.83**
+ down rotation	11.17	**45.56**	**11.37**	45.60
+ down/QKV rotation	11.15	45.26	11.25	45.66

Scope. The guarantee is conditional on the stated uniform estimation-error bounds and candidate separation, which the fixed schedule does not certify. It does not establish recovery under arbitrary shallow-depth scores, optimality beyond the fixed candidate pool, held-out generalization, or superiority at equal compute. Empirical oracle recovery and selection regret characterize complementary aspects of approximation quality but do not verify these stagewise conditions.

D H20 Decoding Throughput

Evaluation protocol. We benchmark Ouro-1.4B at $T = 4$ on a NVIDIA H20 GPU, using batch size/input length pairs of 1/128 and 4/2048. Each run generates 128 tokens without EOS stopping, with the first token produced during prefill and the remaining 127 through cached decoding. Aggregate batch throughput is $127B/t_{\mathrm{decode}}$, measured with CUDA-synchronized wall-clock timing over the complete decoding loop, including Python scheduling but excluding prefill and initialization. All methods use fresh BF16 KV caches with 96 recurrent slots, Flash SDPA, and eager execution without CUDA Graphs or profiling. We report the arithmetic mean of ten throughput measurements, collected in two blocks of three warm-up runs followed by five fixed-input measurements.

Offline and hybrid configurations. Loopy (offline) combines Q/K/V- and gate/up-shared input scaling absorbed into RMSNorm and linear weights, head-wise V–O Hadamard rotation with block size 128 absorbed into paired weights, and up–down scaling that does not cross the SiLU gate. These transforms are shared across recurrent calls and require no explicit online transformation, while activation quantization remains dynamic. This constrained configuration differs from the online Loopy checkpoint. The two hybrid configurations add down-input or down- and Q/K/V-input rotations using seed 42 and block size 128 with matching weights. All three configurations reuse compatible input quantization results across Q/K/V and gate/up without fusing their matrix multiplications.

Kernel differences and diagnostic entries. Loopy uses group-128 Triton packed-W4 kernels, whereas the adapted QuaRot and SpinQuant kernels use per-output-channel weight scales, with additional asymmetric activation zero-point correction for SpinQuant. Attention and timing scope are shared, so the results compare implementations rather than isolate transformation overhead. Entries marked $*$ denote W4A4 integer paths whose NLL differences from BF16 evaluation exceed the predefined tolerance; their throughput cannot be paired directly with BF16-path quality results. W4A8 passes this check on four calibration windows, but full quality evaluation was not repeated on the timed execution paths.

E Evidence and Reproduction

The reported results reflect the current experimental pipeline. The main Loopy quantization results use configurations selected by progressive full-depth evaluation, with full-precision results provided as references. Supporting materials include evaluation reports, JSON records for diagnostic figures, and complete-test records for the rebuilt packed checkpoint.

Figure inputs are stored as paper-local snapshots with source hashes. The accompanying records document the configurations, evaluation protocols, and execution paths associated with the reported

results. Hashes identify data and checkpoint versions; numerical agreement and prediction quality are assessed through the corresponding experiments. The supplementary evidence ledger records the scope of these checks.

F Activation Distributions Across Transformer Blocks

To examine whether the activation behavior illustrated in Figure 3(a) extends beyond the displayed block, we visualize the attention and FFN inputs of all 24 physical Transformer blocks in the recurrent core of Ouro-1.4B at recurrent step 4. All views use the same 2048-token training window, identity scaling, and block-Hadamard rotation with block size 128 and seed 42. Across all 48 block–input pairs, rotation reduces both the maximum absolute activation and the mean per-token maximum-to-RMS ratio. The figures retain all 2048 channels and display maximum absolute values within consecutive 16-token bins; vertical axes are scaled independently, while color ranges are shared within each before–after pair. Blocks are shown in order without omission, extending the block-22 example in the main text; these observations concern the fixed window and recurrent step evaluated here.

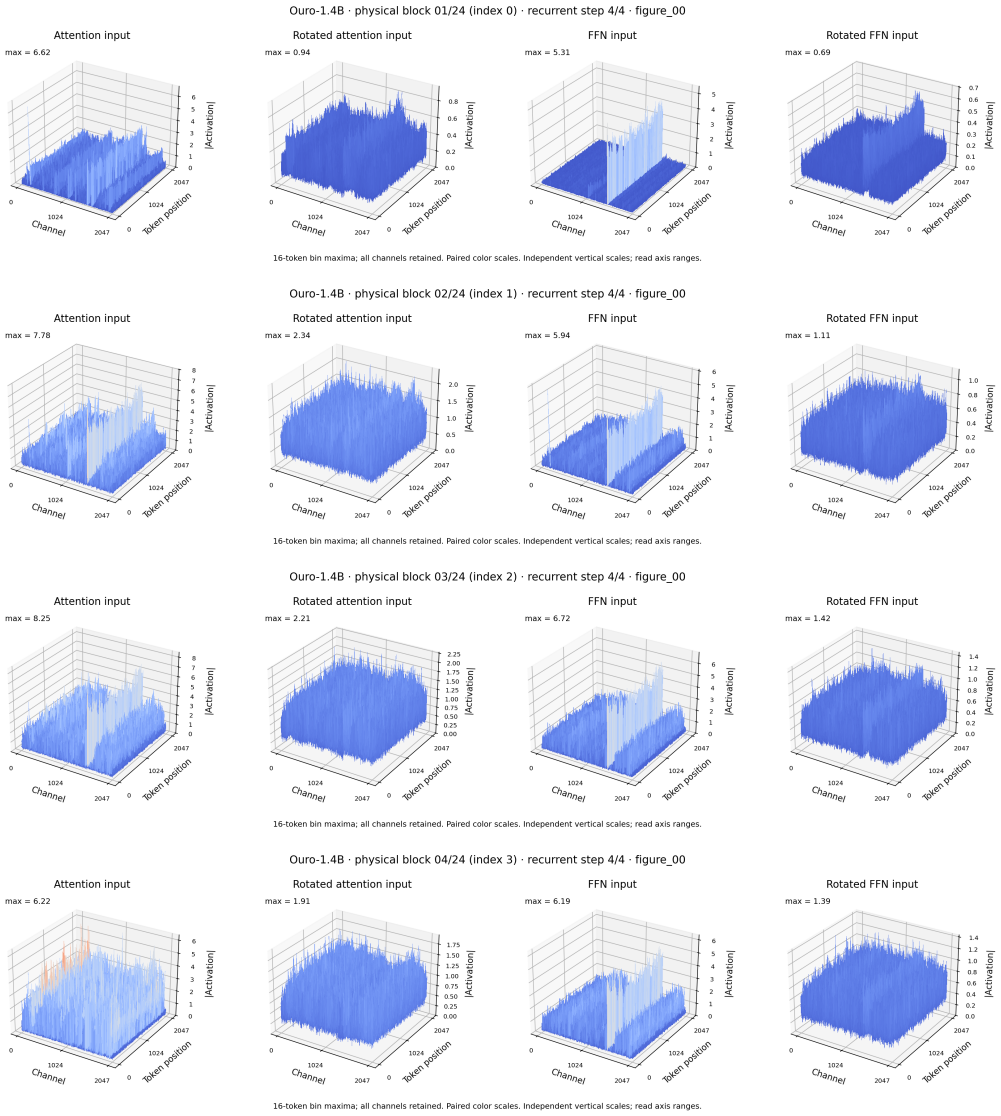


Figure 8: **Activation distributions in blocks 1–4 of Ouro-1.4B at recurrent step 4.** Each row shows attention input, rotated attention input, FFN input, and rotated FFN input, from left to right.

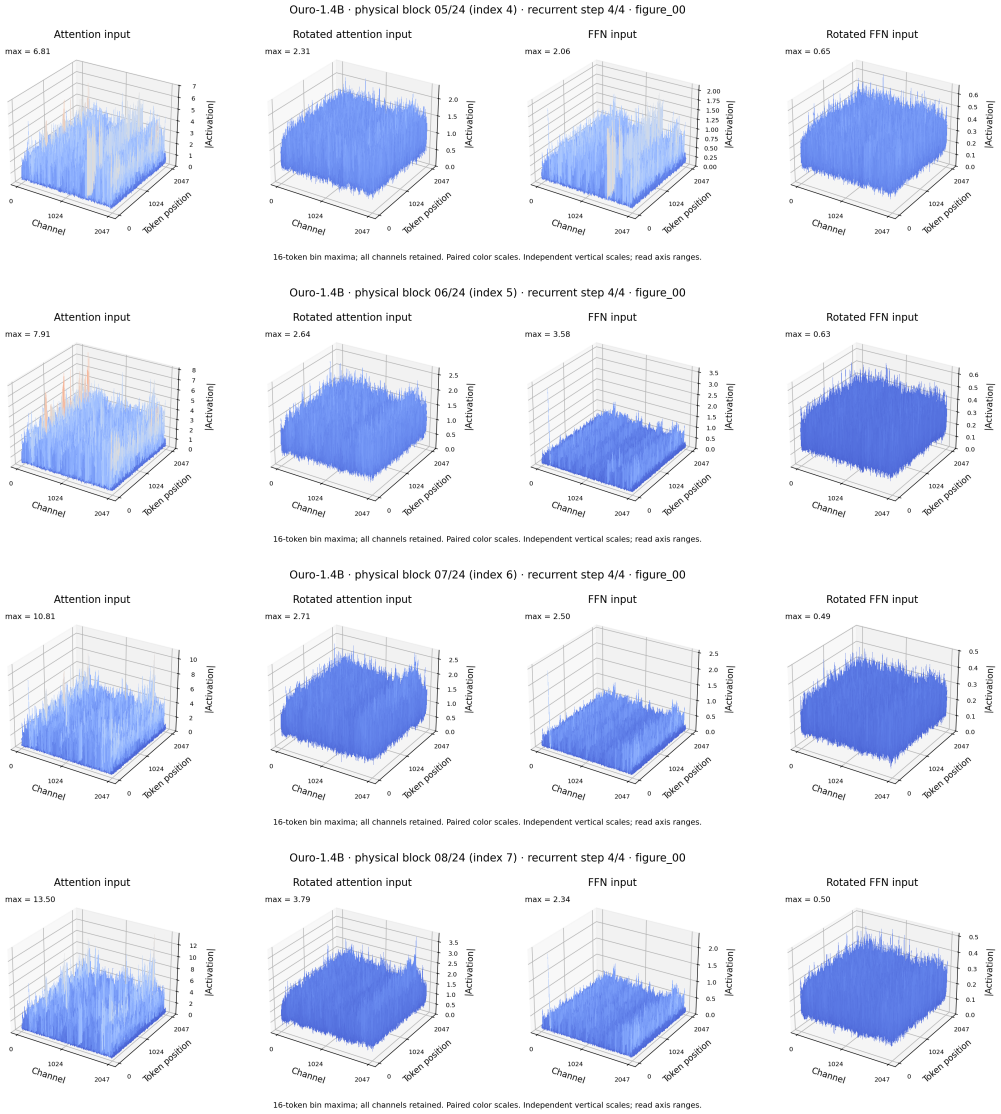


Figure 9: **Activation distributions in blocks 5–8 of Ouro-1.4B at recurrent step 4.** Each row shows attention input, rotated attention input, FFN input, and rotated FFN input, from left to right.

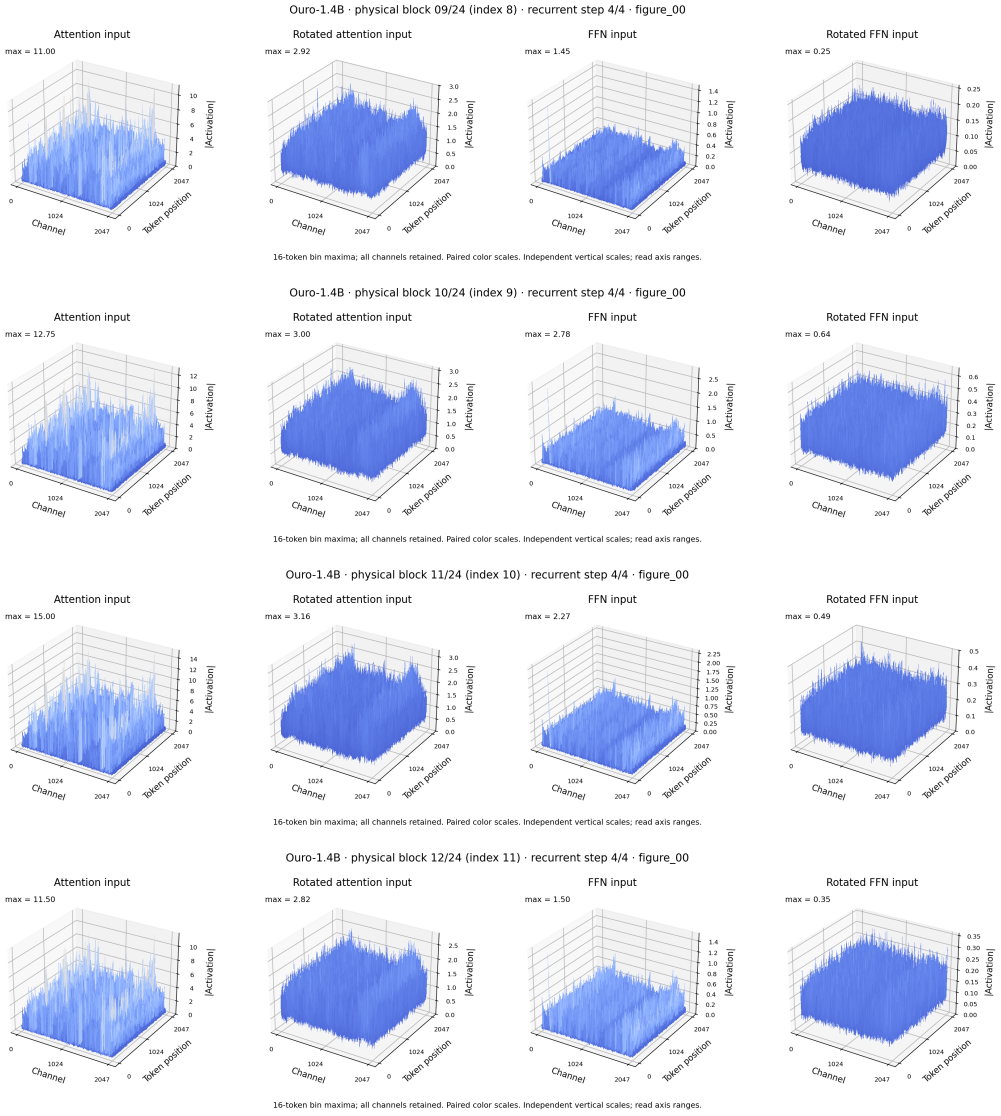


Figure 10: **Activation distributions in blocks 9–12 of Ouro-1.4B at recurrent step 4.** Each row shows attention input, rotated attention input, FFN input, and rotated FFN input, from left to right.

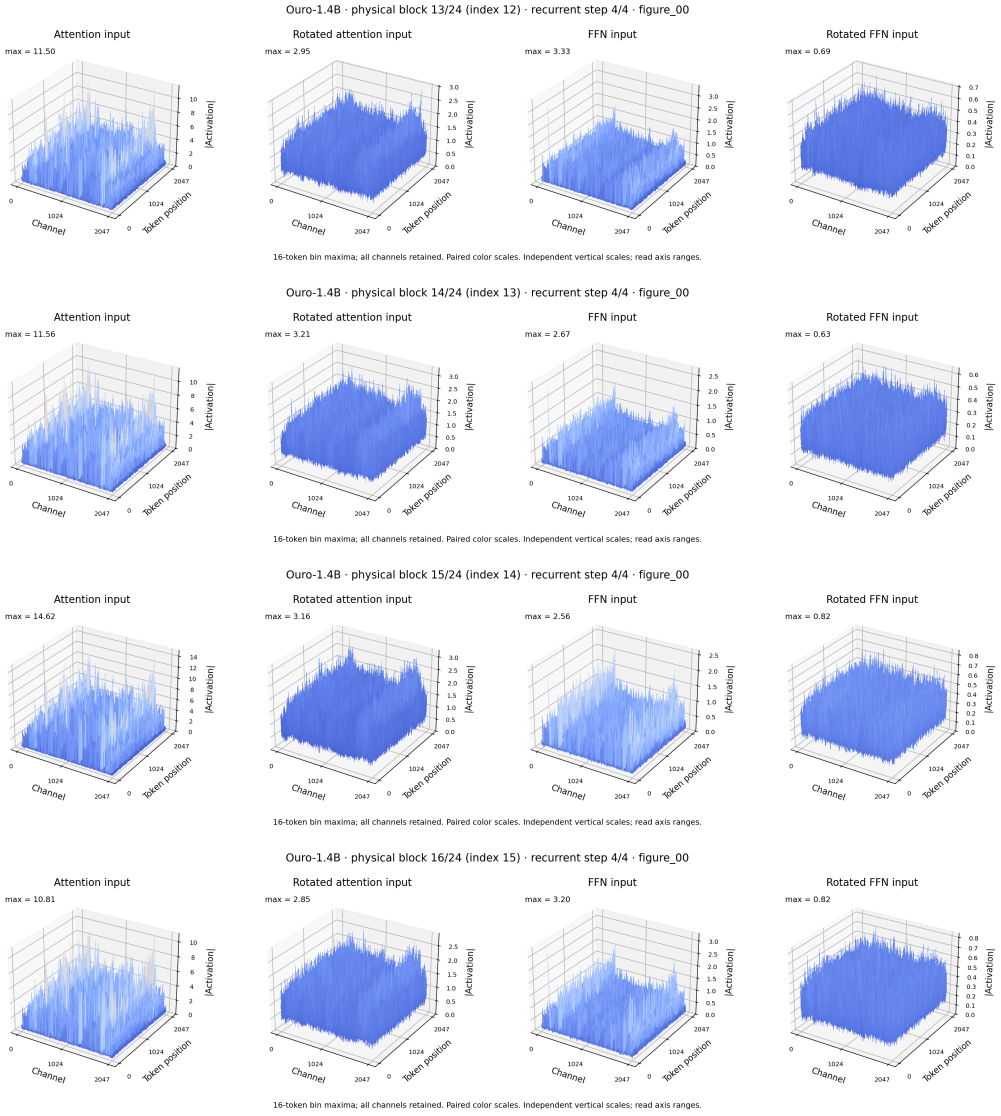


Figure 11: **Activation distributions in blocks 13–16 of Ouro-1.4B at recurrent step 4.** Each row shows attention input, rotated attention input, FFN input, and rotated FFN input, from left to right.

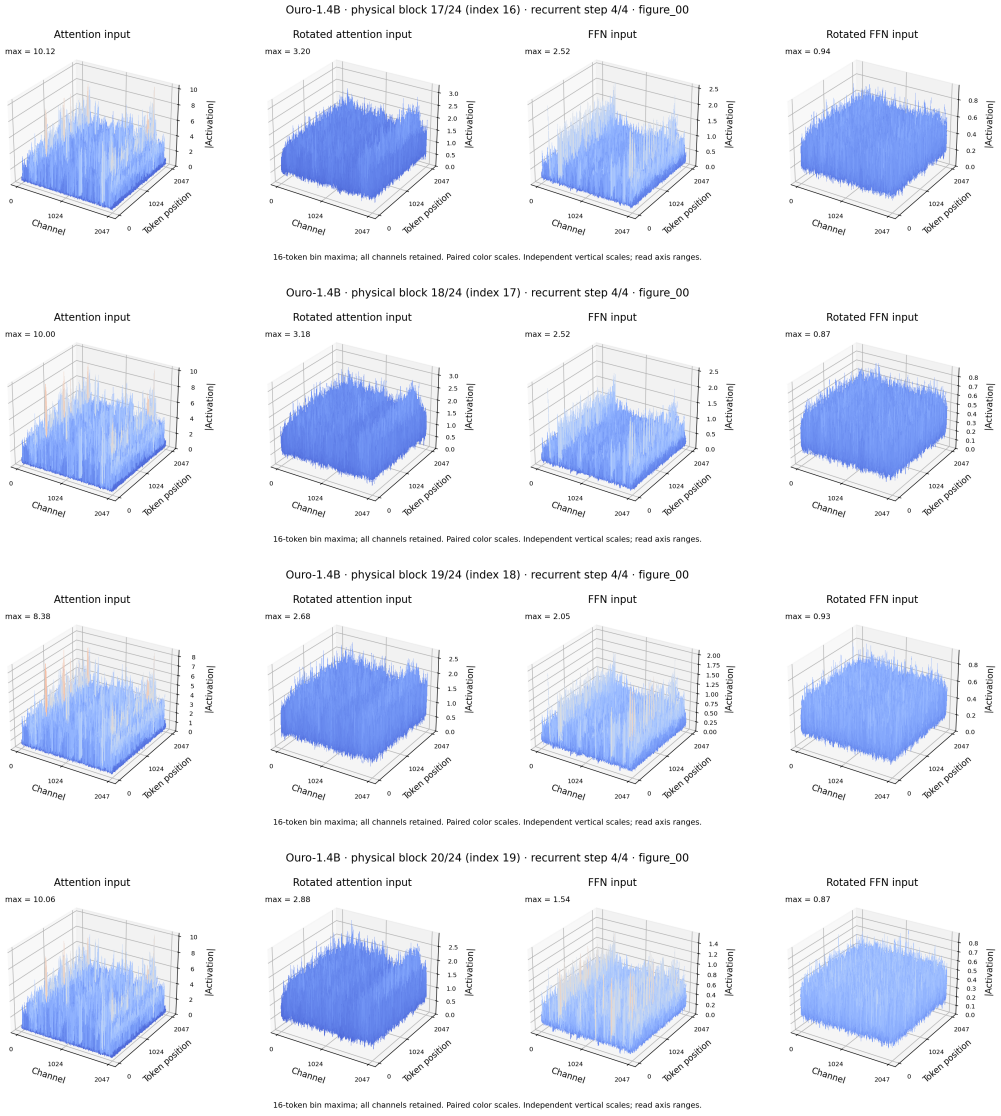


Figure 12: **Activation distributions in blocks 17–20 of Ouro-1.4B at recurrent step 4.** Each row shows attention input, rotated attention input, FFN input, and rotated FFN input, from left to right.

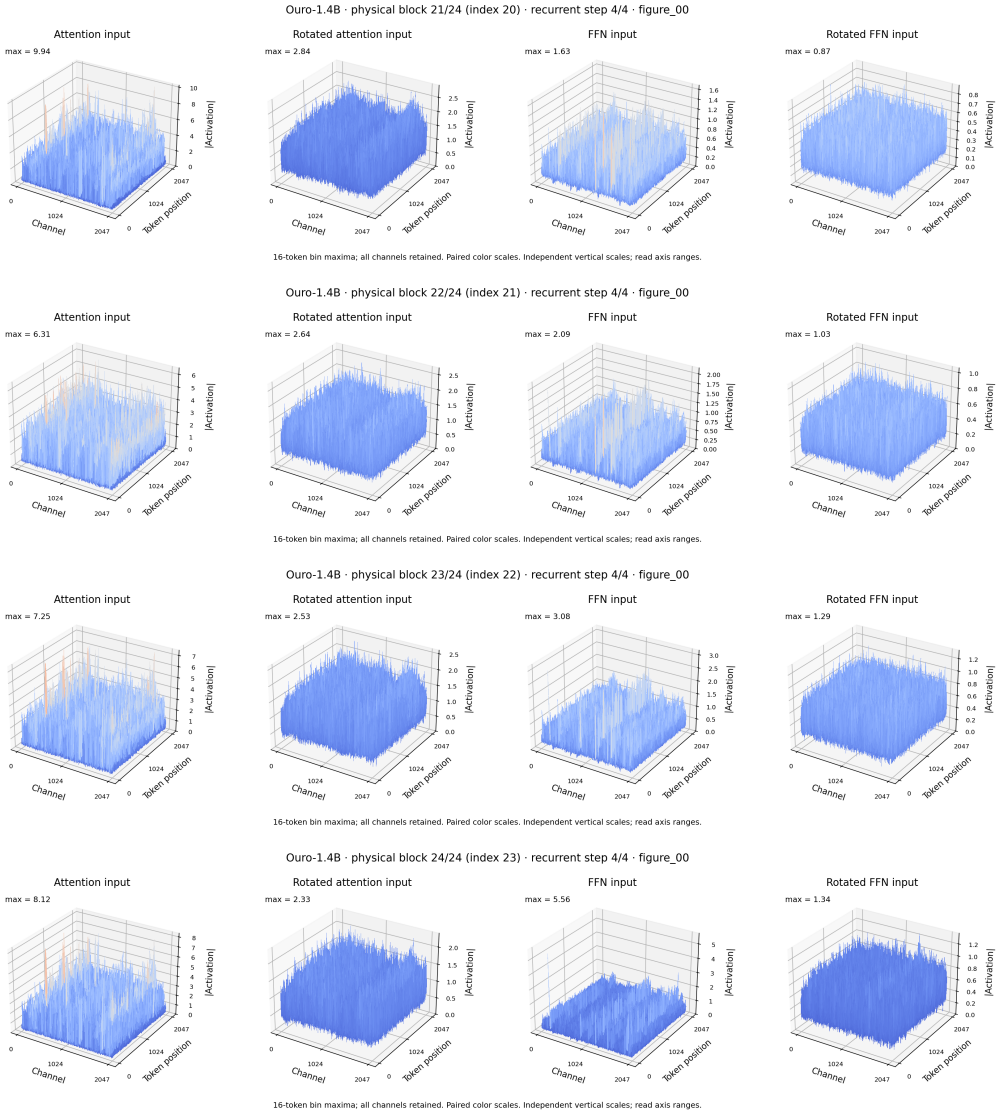


Figure 13: **Activation distributions in blocks 21–24 of Ouro-1.4B at recurrent step 4.** Each row shows attention input, rotated attention input, FFN input, and rotated FFN input, from left to right.

G Activation Distributions Across Recurrent Steps

G.1 Activation Profiles

To complement the comparisons across physical blocks in Appendix F, we visualize the input activations to the query projection in physical block 12 of Ouro-1.4B across four recurrent calls, using the same input window and the same model state as in the preceding activation analysis. The panels exhibit similar overall structure but differ in their local peaks and maximum magnitudes, illustrating that repeated use of a shared module does not imply identical activation patterns. These views provide a qualitative illustration of activation evolution during recurrent execution.

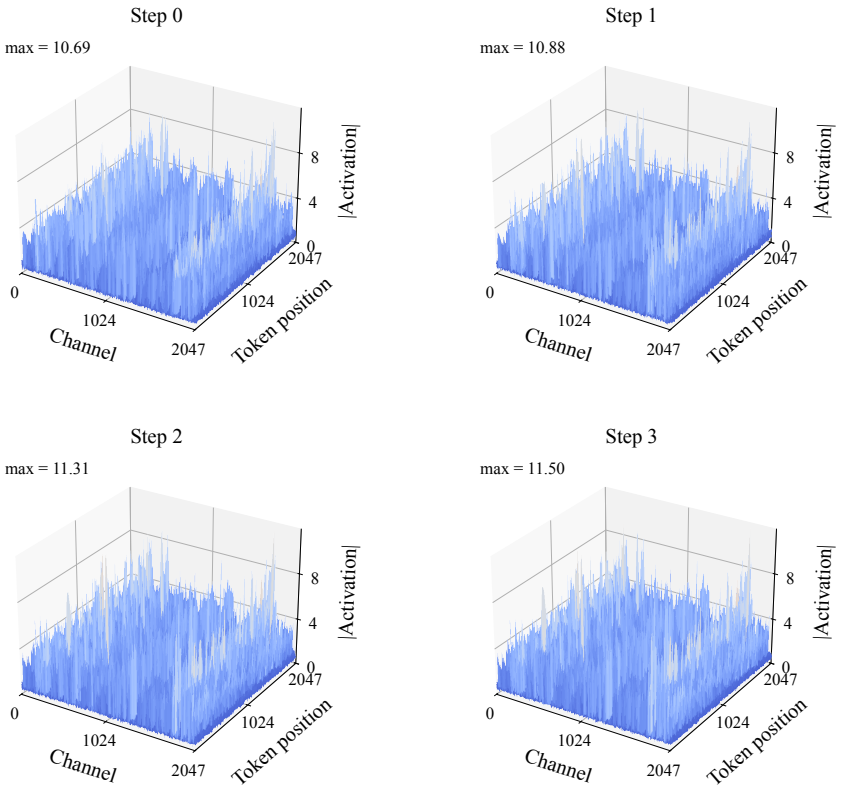


Figure 14: **Activation distributions across recurrent steps.** Input activations to the query projection in physical block 12 of Ouro-1.4B for the same input window. Panels labeled Step 0–3 correspond to the first through fourth recurrent calls. The overall activation structure remains similar, while local peaks and maximum magnitudes vary across calls despite shared weights.

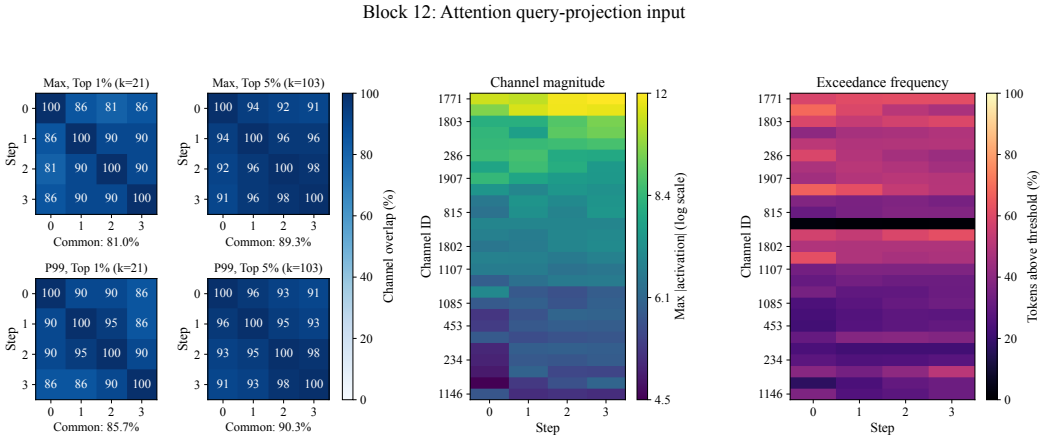


Figure 15: **Outlier-channel stability at the block-12 query-projection input.** Left to right, step-wise Top-1%/5% channel overlap, maximum absolute activation for fixed channels (logarithmic color scale), and token exceedance frequency. The frequency threshold is 1.84375; the four-step common Max Top-1% fraction is 81.0%.

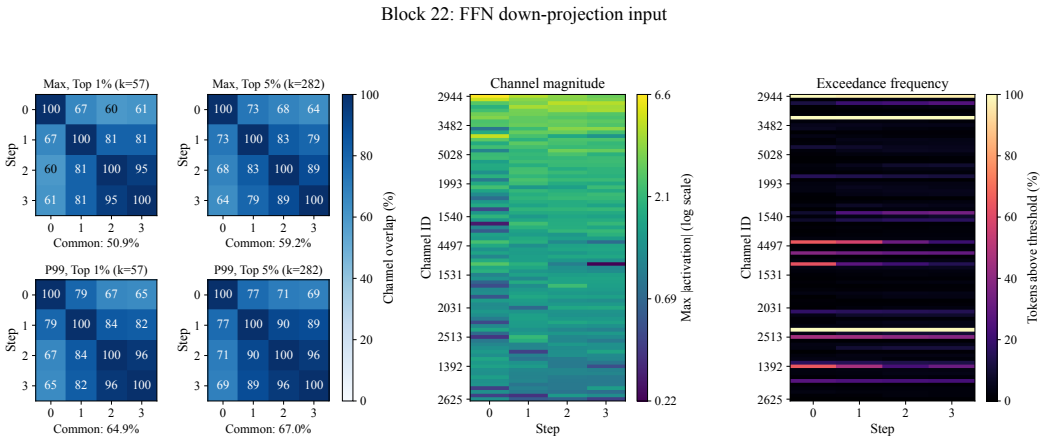


Figure 16: **Outlier-channel stability at the block-22 down-projection input.** Panels follow the same layout as Figure 15. The frequency threshold is 0.11963. The four-step common Max Top-1% fraction is 50.9%, indicating greater channel-set variation than in the block-12 query case.

G.2 Outlier-Channel Stability

We examine inputs to the attention query projection in physical block 12 and the FFN down projection in physical block 22 of ByteDance Ouro-1.4B. Both cases use the same 2048-token window and an unquantized, unrotated BF16 forward pass. Steps 0–3 denote four recurrent calls. Channels are ranked separately at each step by their maximum (Max) or 99th-percentile (P99) absolute activation over tokens. Common denotes the four-step intersection size divided by the per-step Top-k set size. The magnitude and frequency panels fix the union of the four Max Top-1% sets, ordered by mean Max across steps (27 channels for block 12 and 84 for block 22). Frequency measures the fraction of tokens exceeding a case-specific threshold, fixed across steps at the pooled absolute-activation P99. The common Max Top-1% fraction is 81.0% in block 12 and 50.9% in block 22. These cases show partial persistence of high-magnitude channels alongside changes in channel membership, magnitude, and exceedance frequency.

H CONFIGURATION RANKINGS ACROSS RECURRENT DEPTHS

Figure 17 complements the absolute accuracies in Figure 4 with within-depth rankings for the same 20 configurations of Ouro-1.4B W4A4 on WinoGrande. The fixed column order makes changes in relative performance visible as recurrent depth increases. The configuration $(p, s) = (0.8, 2026)$ ranks first at depths 1, 3, and 4, while the marked winner at depth 2 is $(0.9, 2026)$. Thus, the ordering varies across depths even though the shallowest and final-depth winners coincide in this experiment. Rank changes should be interpreted alongside absolute accuracy differences rather than as evidence of their statistical significance.

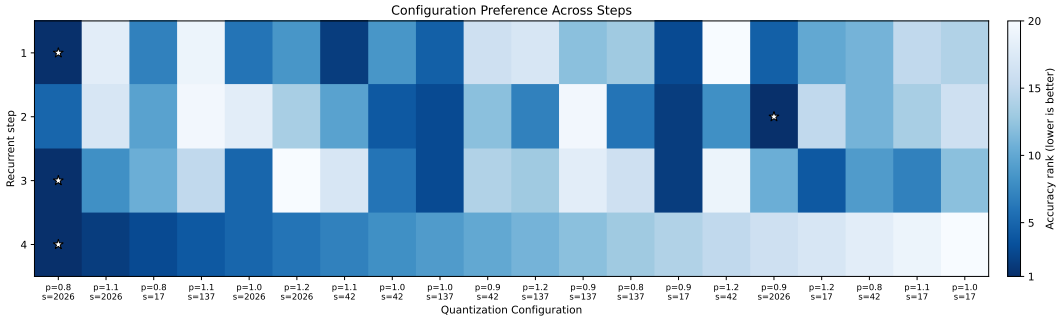


Figure 17: **Within-depth configuration rankings.** WinoGrande accuracy ranks for Ouro-1.4B W4A4, using the same candidate pool and column order as Figure 4. Darker cells indicate better ranks (1 is best); stars mark the displayed top-ranked configuration at each depth.