STORING IS NOT REMEMBERING: LSTM-UT AND BOUNDED GATED MEMORY FOR LOOPED TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent-depth Transformers reuse one block across many steps, so information needed later must survive repeated rewriting of the hidden state. A natural remedy is to keep more history. We show that, in controlled cellular-automaton tasks, making history available is not the same as making it usable. Using Rule 30, where the correct state is known at every recurrent step, we test depth extrapolation and delayed recall, the recovery of an earlier state after further computation. CoTFormer, which caches keys and values from every earlier step, extrapolates less far and recalls less accurately than a Block Universal Transformer (BUT) that keeps only its current state. Interventions show that its retained history can pull a corrected trajectory back toward failure, and that the cache block written at the requested step is neither necessary nor sufficient for recall. We introduce LSTM-UT, which adds a small, bounded, gated cell state to the shared block. Trained to depth 12, LSTM-UT keeps 99.7% exact-row accuracy at depth 60, where BUT gets no row fully correct, and one checkpoint stays above 99.95% at depth 1,000. It also improves delayed recall over both baselines, and the advantage largely persists at near-matched parameter counts. On these tasks, a small state under learned control proved more useful than a complete but unaddressed history. In OpenWebText2 language modelling, LSTM-UT outperforms BUT and comes within 0.6 perplexity of a parameter-matched CoTFormer, which needs up to 91% more training time per step.

1 INTRODUCTION

Universal and looped Transformers apply one shared block repeatedly to an evolving latent state, so more inference-time iterations need no new layers (Dehghani et al., 2018; Saunshi et al., 2025). Because the block is shared, the learned transition is defined at any depth, but nothing guarantees that it remains correct when composed beyond the depths seen in training. Repeated application also raises a separate memory problem: each iteration rewrites the state, so information needed later must survive being overwritten. CoTFormer (Mohtashami et al., 2025) offers one answer, inspired by chain-of-thought. Just as a model reasoning in text can reread its earlier steps, each CoTFormer repeat attends to cached keys and values of the whole sequence from every earlier repeat. In principle, this lets the model retrieve earlier intermediate results rather than having to preserve them in its current state. But having history available is not the same as being able to use it.

We study this distinction on Rule 30 cellular automata, where the correct state is known at every recurrent step. *Depth extrapolation* asks whether a transition trained up to depth K keeps advancing the automaton correctly beyond K. *Delayed recall* asks a model to keep computing and then recover a specified earlier state. We compare three ways of carrying information across steps. The Block Universal Transformer (BUT) keeps only its current hidden sequence. CoTFormer additionally attends to the cached keys and values of every earlier repeat. LSTM-UT, which we introduce, adds a separate fixed-size cell state with learned retention, writing, and reading.

The results cut against the intuition that more history helps. In depth extrapolation, which asks whether the shared transition keeps working, CoTFormer's cache does not help: despite seeing its full trajectory, it falls to chance at shallower depths than BUT. LSTM-UT extrapolates much further:

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.

trained to depth 12, it keeps 99.7% exact-row accuracy at depth 60, where BUT gets no row fully correct, and one checkpoint stays above 99.95% exact-row accuracy at depth 1,000, $83\times$ its deepest supervised step. In delayed recall, the question is instead whether an earlier state can be retrieved after further computation. CoTFormer again trails BUT, and its recalled row rarely matches the requested state even though that state remains in its cache. LSTM-UT improves recall over both baselines. Finally, we train all three architectures as OpenWebText2 language models: LSTM-UT beats BUT and comes close to CoTFormer in perplexity while training substantially faster per step.

Our contributions are:

- **Availability without use.** CoTFormer's expanding cache does not act as addressable memory. At horizon 12, among its own decoded evolution states, the one at the requested step is the unique closest match to the recalled output in only 0.43% of cases, against 23.7% for BUT. In a representative checkpoint, hiding the cache block written at the requested step changes accuracy by at most 0.05 points. State–cache interventions show retained history pulling corrected trajectories back toward failure.
- **LSTM-UT.** A bounded, gated cell state added to a shared Transformer block, which leaves the block's own proposal intact and reads memory through an additive output gate.
- **Depth extrapolation.** LSTM-UT sustains exact Rule 30 computation far beyond its supervised depth, at depths where BUT and a wider, parameter-matched BUT have collapsed to chance.
- **Delayed recall.** LSTM-UT improves recall over both baselines: 90.4% cell accuracy at horizon 12, against 79.5% for BUT and 60.0% for CoTFormer. With 6.5% fewer parameters than BUT it still reaches 88.9%, and it leads the near-matched BUT by 6.3 to 9.6 points in all three protocols tested.
- **Language modelling.** On OpenWebText2, LSTM-UT beats BUT at every repeat count and comes within 0.6 perplexity of a parameter-matched CoTFormer, which needs up to 91% more training time per step.

2 Background and Related Work

2.1 Recurrent depth and memory

Recurrent-depth Transformers increase computation by repeatedly applying shared layers to an evolving hidden state. Universal Transformers introduced this approach through recurrent refinement and adaptive computation (Dehghani et al., 2018), while Geiping et al. (2025) demonstrate that increasing recurrent depth at inference can improve performance on reasoning benchmarks. Perplexity alone, however, does not fully capture the benefits of recurrence: Saunshi et al. (2025) show that looped models can match or outperform non-looped models on reasoning tasks despite having worse perplexity at the same effective depth. These results motivate evaluating recurrent computation directly, but do not establish whether each additional iteration faithfully performs an intended computational step. We study this question using Rule 30 cellular automata, where the correct state is known at every iteration, and use delayed recall to test whether earlier states remain accessible after further computation. Work on continuous latent reasoning and recursive Transformers has proposed persistent or explicitly managed state to reduce repeated overwriting of a single latent representation (Farhan & Chaudhary, 2026; Yu et al., 2026). We compare three mechanisms that may serve this role: a fixed-size current hidden sequence, an expanding attention history, and a bounded gated cell state.

3 LSTM Universal Transformer: A Bounded Gated Alternative

If a complete history is not reliably used, a small state that the model must actively write to and read from is a natural alternative. The BUT has a fixed-size recurrent state but can overwrite earlier information; CoTFormer exposes an expanding history but did not reliably extrapolate or address it in our experiments. LSTM-UT instead adds a separate, bounded cell state to the shared Transformer block. Related approaches also separate transient computation from persistent memory (Farhan & Chaudhary, 2026; Yu et al., 2026).

3.1 ARCHITECTURE

Let z be the block input and $\bar{h} = \mathcal{T}_\theta(z)$ the ordinary Transformer proposal. A shared memory layer normalisation gives $x = \mathrm{LN}_m(z)$ and $h = \mathrm{LN}_m(\bar{h})$. LSTM-UT updates its cell c and residual stream as

$$f = \sigma(W_{fx}x + W_{fh}h), \qquad i = \sigma(W_{ix}x + W_{ih}h), \tag{1}$$

$$\widetilde{c} = \tanh(W_{cx}x + W_{ch}h), \qquad c^+ = f \odot c + i \odot \widetilde{c}, \tag{2}$$

$$o = \sigma(W_{ox}x + W_{oh}h), \qquad z^+ = \bar{h} + o \odot \tanh(c^+). \tag{3}$$

Thus the Transformer proposal is preserved while the output gate controls an additive read from memory. The cell is initialised to zero for each input, persists across evolution and recall applications, and has fixed sequence length and width. It is not supplied directly to self-attention or the MLP.

3.2 EXPERIMENTAL SCOPE

We compare LSTM-UT with BUT and CoTFormer on Rule 30 depth extrapolation (Section 4) and delayed recall (Section 5), using identical task definitions and training protocols within each comparison, and on OpenWebText2 language modelling (Section 6). In the cellular-automaton experiments, common-width comparisons use $d = 64$; width-adjusted comparisons approximately match trainable parameter count but not runtime, computation, or memory traffic.

4 RULE 30 DEPTH EXTRAPOLATION

Depth extrapolation asks whether each further application of the shared block still performs one correct Rule 30 update beyond the supervised range.

4.1 TASK AND PROTOCOL

Rule 30 synchronously updates each binary cell from its two periodic neighbours: $\rho(x)_i = x_{i-1} \oplus (x_i \vee x_{i+1})$. Models receive a 64-cell initial row and apply one shared Transformer block per requested update, looped as in Universal and looped Transformers (Dehghani et al., 2018; Saunshi et al., 2025). All models are built on the GPT-2 block (Radford et al., 2019), modified to use bidirectional attention and rotary positions; the principal models use width 64, four heads, an MLP width of 4 times the embedding dimension, and zero dropout. Unlike recurrent-depth language models that surround the looped block with non-recurrent prelude and coda layers (Geiping et al., 2025), our cellular-automaton models contain only the shared block, which we write as a $0/1/0$ (prelude/looped/coda) layout. We write R_K for training that cycles deterministically through depths $1{:}1, \ldots, K{:}K$, one depth per optimizer step. Only the final row is supervised, and no clean intermediate states are supplied. R_6 uses 100,000 training rows and 7,500 steps; R_{12} uses 500,000 rows and 24,000 steps. Validation and final-test splits each contain 4,096 independently generated rows. Checkpoints are selected by validation cell accuracy at an unseen depth (9 for R_6, 15 for R_{12}), with no minimum in-distribution score, and the final-test split is used only for reporting. We report cell accuracy and exact-row accuracy, which requires all 64 cells to be correct.

4.2 DEPTH-EXTRAPOLATION RESULTS

LSTM-UT sustained accurate Rule 30 computation far beyond its supervised depth. Under R_{12} it reached $99.67 \pm 0.56\%$ exact-row accuracy at depth 60, whereas width-64 BUT produced no exactly correct rows and fell to chance-level cell accuracy (Figure 1, right). Both were effectively perfect at the supervised boundary, so the gap lies in extrapolation. Under R_6, LSTM-UT also led BUT at depth 20 (Figure 1, left). Appendix Table 6 and Figure 4 give exploratory, correlational diagnostics from one representative BUT run as it fails at depth.

Parameter count does not explain the gap. Width-80 BUT (77,200 parameters, within 6.3% of width-64 LSTM-UT) still fell to 50.01% cell and 0% exact-row accuracy at depth 60. Nor does extrapolation simply shrink with width. Under R_6, the 20,704-parameter width-32 LSTM-UT, with fewer than half the parameters of width-64 BUT, extrapolated further than width-64 LSTM-UT:

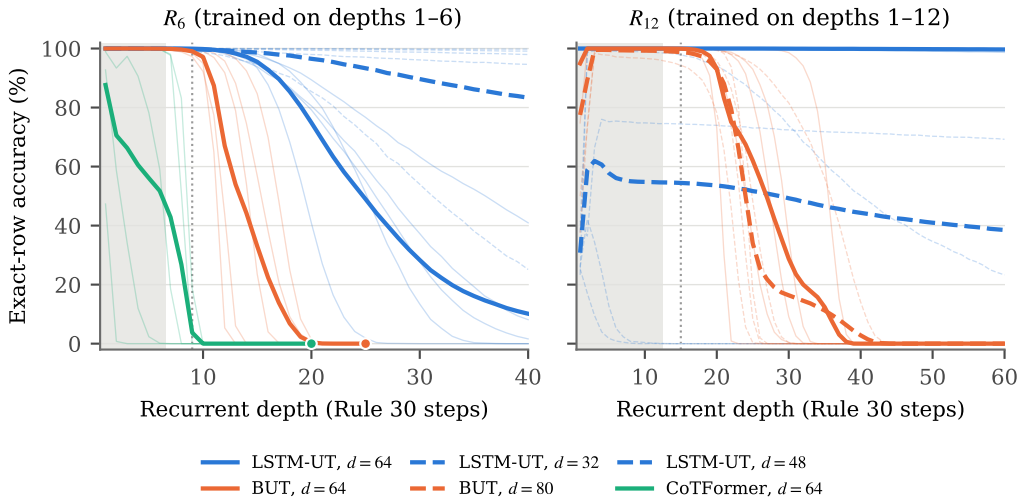

Figure 1: Rule 30 depth extrapolation: final-test exact-row accuracy against recurrent depth for the validation-selected unconstrained extrapolation checkpoints. Bold lines are means over five model-initialisation seeds; faint lines are individual seeds. Grey bands mark the supervised depths and dotted lines the checkpoint-selection depth. Lines ending in a dot were not evaluated further; every seed was already at 0% exact-row accuracy there. Full cell and exact-row values are in Appendix Tables 7 and 8.

$96.5 \pm 6.2\%$ against $74.9 \pm 19.4\%$ exact-row accuracy at depth 20. Under R_{12}, the 46,416-parameter width-48 LSTM-UT can also extrapolate: one seed was exactly correct on every final-test row at every depth from 2 through 60, although at depth 1 it computed the update with non-wrapping row edges (Appendix A.3). It does so unreliably, with two seeds collapsing to chance and two intermediate. Width therefore affects how reliably training finds the extrapolating solution, but not monotonically, and that solution does not require the width-64 parameter count.

Long-horizon stress tests. We ran two further, independently trained LSTM-UT checkpoints far beyond their training depth. An R_8 checkpoint retained 99.95% exact-row accuracy at depth 200. An R_{12} checkpoint was evaluated through depth 1,000 on 15 held-out conditions (three data seeds $\times$ five initial-state densities $p \in \{0.1, \ldots, 0.9\}$). Its exact-row accuracy never fell below 99.95% at any depth or condition; at depth 1,000, $83\times$ its deepest supervised step, only 10 of 61,440 rows were wrong.

4.3 What fails in CoTFormer?

CoTFormer can attend to every earlier step, yet it extrapolated least far; we ask whether that history helps or hurts. Under the R_{12} protocol, CoTFormer showed a gradual failure regime suitable for intervention: its strongest run fell from 99.93% cell accuracy at repeat 15 to 50.49% at repeat 20. Restricting a frozen full-cache checkpoint to recent cached repeats did not rescue it. Training with bounded cache visibility also did not produce a monotonic cache-size benefit: at depth 20, full-cache accuracy was $50.08 \pm 0.01\%$, every recent-2 through recent-8 policy was near chance, while recent-1, functionally equivalent to BUT, retained $99.39 \pm 0.55\%$ (Appendix Table 10).

We therefore crossed two one-time interventions at selected depths: replace the current hidden sequence with the clean Rule 30 embedding, and clear the existing CoT cache. The four paired conditions use the same examples and pre-intervention trajectory; subsequent repeats again accumulate keys and values normally. Table 1 reports mean cell-accuracy changes over the next five predictions, averaged first within each run over tested intervention depths and then across runs.

Correcting the state helped less as the trained cache window grew, while clearing the cache helped increasingly *after* state correction. Cache reset alone had almost no effect. This pattern holds from recent-2 through recent-8: injecting a clean state and resetting the cache together gives the most

Table 1: Five-repeat effects of CoTFormer state–cache interventions on cell accuracy, in percentage points (mean ± sample SD across runs). State is clean-state injection with the old cache retained; reset after clean is the additional effect of clearing that cache; reset alone keeps the corrupted state. Intervention depths were selected around failure and are not balanced across policies; Appendix A.4 defines the effects.

Training cache	State	Reset after clean	Reset alone
Recent 2	18.2 ± 16.2	31.6 ± 16.6	0.1 ± 0.1
Recent 3	8.3 ± 3.7	41.4 ± 3.2	0.1 ± 0.1
Recent 4	3.7 ± 5.7	46.2 ± 5.9	0.0 ± 0.0
Recent 5	1.7 ± 3.1	48.2 ± 3.1	0.0 ± 0.0
Recent 8	-0.9 ± 1.4	50.1 ± 0.1	0.4 ± 0.5
Full	0.2 ± 0.2	49.5 ± 0.3	0.0 ± 0.0

robust rescue, whereas a clean state alone rescues less robustly, temporarily for recent-2 and not at all in the recent-8 example. Even the joint rescue is temporary, fading by depth 40. Appendix Figure 7 shows these post-intervention trajectories.

5 Delayed Cellular-Automaton Recall

Delayed recall tests memory use directly: the answer is a state the model has already computed, so the question is whether it can retrieve it.

5.1 Task and protocol

A model receives a binary initial row $X^{(0)}$, executes H Rule 30 updates, and is then queried for an earlier row $X^{(q)}$, $q \in \{1, \ldots, H\}$. The query specifies the relative age $a = H - q$. Evolution always completes before recall, so the model cannot answer an early-state query merely by stopping at q. BUT must retain or reconstruct the past from its current hidden sequence; CoTFormer can additionally attend to cached keys and values from earlier repeats; LSTM-UT can additionally read its bounded cell state, which persists through the recall applications. All three receive the same learned direction and age controller.[1] For a shared block G_θ, the main experiments use a transient, subtractive controller update $Z' = G_\theta(Z + c) - c$, where $c = e_{\mathrm{dir}} + e_{\mathrm{age}}$. One or two additional shared-block applications produce the recall answer. Training supervises every evolution repeat and the final recall output with cell-wise cross entropy, using $\mathcal{L} = (\mathcal{L}_{\mathrm{evo}} + 2\mathcal{L}_{\mathrm{query}})/3$; the evolution loss averages evenly over the H repeats. Horizons $1{:}1, \ldots, H_{\max}{:}H_{\max}$ and their queries cycle deterministically. The main sweep uses $H_{\max} \in \{6, 12, 20\}$, three model-initialisation seeds, a common data seed, 64-cell rows, width 64, four heads, and the same $0/1/0$ shared-block architecture for all three models. Runs use 5,000 optimizer steps. Training has 100,000 initial rows; validation and test each have 4,096 independent rows. Validation runs every 2,500 steps and selects, among steps 0, 2,500, and 5,000, the checkpoint with the highest ground-truth recall cell accuracy, averaged first over nontrivial queries within each horizon and then equally over horizons. We exclude $q = H$, which requires no delayed recall. All reported main-sweep results use frozen checkpoints on the final-test split. The three architectures use identical data, splits, optimiser settings, training length, and checkpoint candidates, differing only in architecture and model seed.

5.2 Results

Figure 2 and Table 2 compare the three architectures at common width. LSTM-UT led both baselines at every maximum horizon and recall count, and BUT led CoTFormer in every case, by 5.97 to 10.00 percentage points with one recall application and by 6.48 to 19.46 with two. A second recall application was not uniformly beneficial. At $H_{\max} = 12$ with two recall applications, LSTM-UT exceeded BUT by 10.94 cell-accuracy points and 37.20 exact-row points. These are descriptive

[1]The controller adds a two-entry direction embedding and an $(H_{\max} + 1)$-entry age embedding of width d, i.e. $15d$ parameters at $H_{\max} = 12$. Parameter counts in this section therefore exceed those of the same architectures on the Rule 30 task; for example, width-64 BUT has 50,432 parameters here and 49,472 there.

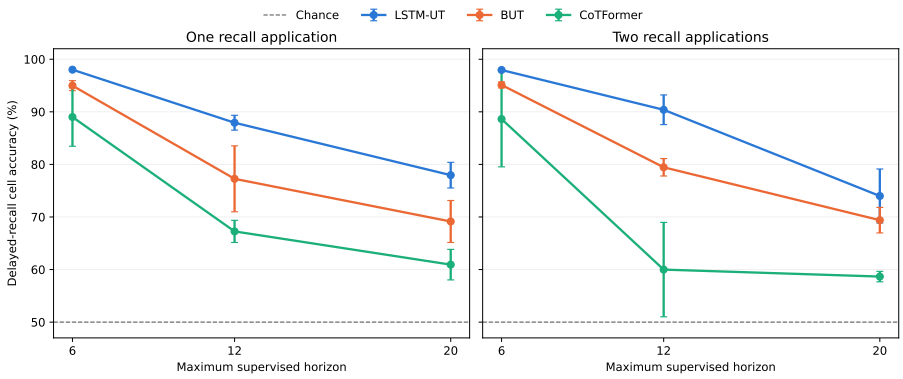

Figure 2: Pair-balanced final-test delayed-recall cell accuracy for width-64 LSTM-UT, BUT, and CoTFormer. Points are means over three model seeds and bars show one sample standard deviation; $q = H$ is excluded. Left and right panels use one and two recall applications. All models share the same data, training schedule, and checkpoint candidates.

results from one shared data seed, and common width does not control parameter count, which Section 5.4 addresses.

Table 2: Width-64 delayed-recall performance for models trained through horizon 12 with two recall applications. Values are percentages reported as mean ± sample standard deviation over three model-initialisation seeds on the final-test split, using each run's validation-selected best delayed recall checkpoint. The macro excludes no-op queries and weights every training horizon equally. "Internal" compares the recalled output with the model's decoded state at the requested repeat; "GT unique best" is the rate at which the recalled decoded row is uniquely closest in cell agreement to the exact state at the requested repeat among all exact trajectory states. Bold marks the best value in each column.

Architecture	Cell	Internal	Exact row	GT unique best
LSTM-UT	$\mathbf{90.39 \pm 2.82}$	$\mathbf{90.50 \pm 2.73}$	$\mathbf{48.40 \pm 15.13}$	$\mathbf{83.06 \pm 5.10}$
BUT	79.45 ± 1.66	79.79 ± 1.54	11.20 ± 8.78	62.05 ± 2.47
CoTFormer	59.99 ± 8.98	76.91 ± 6.30	1.69 ± 2.93	21.32 ± 5.64

5.3 Why CoTFormer's cache does not help

At $H_{\max} = 12$, one-application forward accuracy at repeat 12 was 87.1% for BUT but approximately chance for CoTFormer, so ground-truth recall alone conflates computation and retrieval errors. We therefore also compare recall with the model's own decoded row at the requested repeat. Raw internal agreement can be misleading if recurrent states are similar; *changed-cell* internal accuracy restricts attention to cells on which the requested and current decoded rows differ. These diagnostics use queries at evaluation horizon $H = 12$ of the $H_{\max} = 12$ protocol, whereas Table 2 averages over every horizon up to 12, so the two sets of values differ. With one recall application, BUT and CoTFormer had similar raw internal accuracy (64.8% and 64.1%), but changed-cell accuracy was 63.5% versus 21.0%. The requested repeat was the unique closest logit-space match in 23.7% versus 0.43% of cases (Appendix Figure 5). With two recall applications, CoTFormer had higher raw internal agreement at this horizon (71.1% versus 65.9%) but much lower changed-cell agreement (12.0% versus 67.9%); averaged over all horizons, BUT's raw internal agreement was instead the higher (Table 2). Thus high agreement with its own trajectory did not imply selection of the requested past state. CoTFormer's gap between internal and ground-truth accuracy in Table 2 makes the same point.

If CoTFormer used its history as memory, the cache block written at the requested step should matter for recall. Post-hoc validation diagnostics tested whether the controller-indexed evolution cache block was used as a lookup target. Permuting that block's values or masking it tests its influence or necessity; leaving it as the only visible evolution block tests sufficiency, while recall-phase blocks

remain visible. For a four-head horizon-20 checkpoint, hiding the nominal block changed cell accuracy by -0.01 points at query 10 and $+0.05$ at query 19. Leaving only that evolution block instead changed accuracy by -3.88 and -15.60 points. The nominal block was neither necessary nor sufficient for those answers, though the latter result shows that other cached blocks mattered collectively. The block written at update q precedes the completed state Z_q, so these tests do not exclude an offset or distributed use of other blocks.

Table 3: Four-head CoTFormer cache interventions at horizon 20, using a selected validation checkpoint. Parentheses show cell-accuracy changes from normal recall in percentage points.

Condition	Query 10	Query 19
Normal recall	56.74%	62.25%
Permute requested block's values	56.72% (-0.02)	60.89% (-1.36)
Hide requested evolution block	56.73% (-0.01)	62.30% $(+0.05)$
Hide every other evolution block	52.87% (-3.88)	46.65% (-15.60)

Six separately trained single-head models gave the same broad diagnosis. Across seven non-current horizon-20 queries, the requested block was most attended in $4.66 \pm 8.07\%$ of observations with one recall application and $2.31 \pm 2.20\%$ with two, compared with 5% under uniform selection among 20 evolution blocks. Masking it changed mean accuracy by -0.110 ± 0.183 and -0.011 ± 0.013 percentage points respectively. These are selected-checkpoint, validation-set diagnostics, not independent confirmatory tests. They used manual attention because the trained SDPA implementation did not expose probabilities; prediction agreement with the fused version was 97.29% for the four-head checkpoint and 96.93%–99.93% across single-head checks. The result concerns learned behaviour on this task, not a formal inability of attention to retrieve past states.

5.4 Parameter-matched comparison

The fixed-width comparison gives LSTM-UT more parameters than BUT. We therefore repeated delayed recall with widths chosen to approximately match trainable parameter count. Table 4 gives the cleanest central comparison: at $H_{\max} = 12$ with two recall applications, width-48 LSTM-UT has 6.54% fewer parameters than width-64 BUT, but leads it on all four recall measures.

Table 4: Near-parameter-matched delayed recall at $H_{\max} = 12$ with two recall applications. Accuracy values are percentages reported as mean $\pm$ sample standard deviation over three model-initialisation seeds. Bold marks the better value in each accuracy column.

Architecture	Parameters	Cell	Internal	Exact row	GT unique best
LSTM-UT, $d = 48$	47,136	$\mathbf{88.94 \pm 1.41}$	$\mathbf{89.90 \pm 1.23}$	$\mathbf{18.82 \pm 7.89}$	$\mathbf{84.41 \pm 2.87}$
BUT, $d = 64$	50,432	79.45 ± 1.66	79.79 ± 1.54	11.20 ± 8.78	62.05 ± 2.47

The same pair was compared in two further protocols. With one recall application, width-48 LSTM-UT led width-64 BUT by 9.57 cell-accuracy points at $H_{\max} = 12$ and by 6.27 points at $H_{\max} = 20$, where the parameter gap differs by at most 0.19 percentage points from the horizon-12 gap.

Extra parameters therefore do not explain the LSTM-UT advantage, although matching parameter count does not also match computation, memory traffic, or runtime.

6 Language Modelling

The cellular-automaton tasks isolate state tracking; we also test whether LSTM-UT trains competitively as a language model and at what cost.

6.1 Setup

All models are GPT-2-style causal language models (Radford et al., 2019) trained on OpenWebText2 (Gao et al., 2020), following the CoTFormer protocol (Mohtashami et al., 2025). Unlike the cellular-automaton models, each has a 2-layer prelude, a 9-layer core looped $R \in \{5, 8, 10\}$ times, and

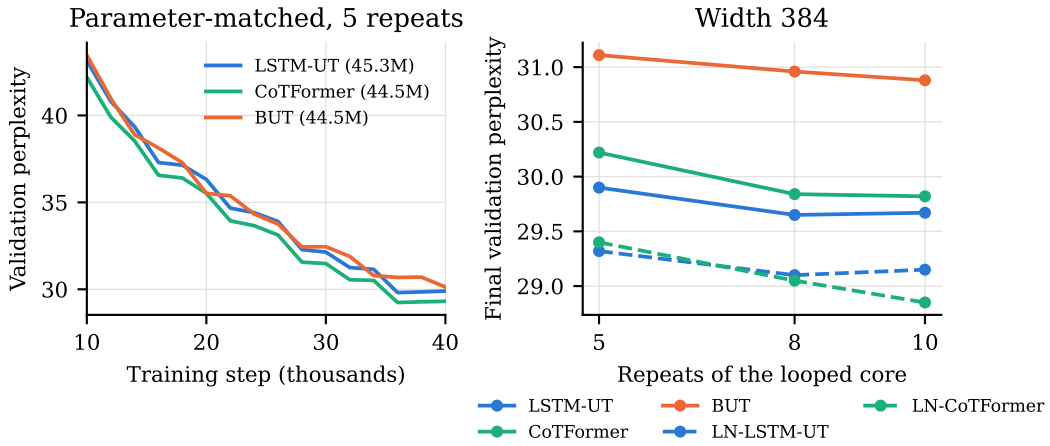

Figure 3: OpenWebText2 language modelling. Left: validation perplexity during training for the parameter-matched models with five repeats (width-408 BUT and CoTFormer, width-384 LSTM-UT). Right: validation perplexity at step 40,000 against the number of repeats at width 384; dashed lines add a layer normalisation after every repeat. One model seed per configuration.

a 1-layer coda (effective depth 48, 75, and 93), with width 384, six heads, rotary positions, and sequence length 256. Training uses 40,000 AdamW steps at batch size 256 (about 2.6B tokens), learning rate 10^{-3} with 20% warmup and cosine decay, and one model seed per configuration. In LSTM-UT the cell persists through the prelude, the loop, and the coda: each prelude and coda block, and each complete application of the looped core, is followed by one gated update whose previous input is the block's or core's input and whose proposed input is its output. "LN-" variants apply a layer normalisation after every repeat, as in LN-CoTFormer (Mohtashami et al., 2025). For a parameter-matched comparison at $R = 5$, BUT and CoTFormer were also trained at width 408 (44.51M parameters, against 45.28M for LSTM-UT). We report validation perplexity at step 40,000 and the median time per training step after step 1,000, comparing times only between runs at the same width and repeat count on the same GPU type.

6.2 Results

Figure 3 and Table 5 summarise the runs. At width 384, LSTM-UT reached lower perplexity than BUT at every repeat count, by 1.2 to 1.3 points, and slightly lower perplexity than CoTFormer (29.90, 29.65, and 29.67 against 30.22, 29.84, and 29.82). At matched parameter count, however, CoTFormer was better (29.31 against 29.90, with BUT at 30.13), and with a layer normalisation after every repeat the two were close at $R = 5$ (29.32 for LN-LSTM-UT, 29.40 for LN-CoTFormer) while LN-CoTFormer was lower at $R = 8$ and $R = 10$.

LSTM-UT was substantially cheaper to train than CoTFormer. On the same GPUs, CoTFormer took 22.5%, 60.4%, and 90.7% more time per step than LSTM-UT at $R = 5$, 8, and 10, and 23.0%, 60.2%, and 89.4% more for the LN variants; total job wall-clock shows the same ratios. The overhead grows with depth because each CoTFormer repeat attends over keys and values cached from every earlier repeat, whereas the LSTM-UT cell has a fixed size. LSTM-UT cost 10.8% to 13.6% more per step than BUT. With one seed per configuration, perplexity differences of a few tenths should be treated as indicative. The evidence is that bounded gated memory is competitive with an expanding cache for language modelling at a fraction of its training cost, not that it improves perplexity over CoTFormer.

7 Discussion and Limitations

LSTM-UT keeps BUT's fixed-size recurrent interface but adds a separate state that learned gates retain, update, and expose. This separation may reduce competition between transforming the residual

Table 5: OpenWebText2 validation perplexity at step 40,000 and median time per training step at width 384, one model seed per configuration. Times at $R = 5$ were measured on A100 GPUs, except for the LN variants; all other times were measured on H200 GPUs. Bold marks the lowest perplexity in each column.

		Validation perplexity			Time per step (ms)		
Model	Parameters	$R = 5$	$R = 8$	$R = 10$	$R = 5$	$R = 8$	$R = 10$
BUT	40.56M	31.11	30.96	30.88	411	350	427
CoTFormer	40.56M	30.22	29.84	29.82	572	627	902
LSTM-UT	45.28M	29.90	29.65	29.67	467	391	473
LN-CoTFormer	40.56M	29.40	**29.05**	**28.85**	331	631	907
LN-LSTM-UT	45.28M	**29.32**	29.10	29.15	269	394	479

stream and preserving information for later use; the results are consistent with that interpretation but do not establish it as the mechanism.

The internal role of the LSTM cell remains unresolved. Aggregate recall accuracy does not reveal whether the cell stores compressed snapshots of the trajectory, features from which earlier rows can be reconstructed, a controller-dependent summary, or information that merely improves the residual stream's own recurrent dynamics. Unlike the CoTFormer analysis, the present study does not intervene directly on the LSTM cell, inspect its gates across depth, or test whether a particular cell state is necessary or sufficient for recall. Targeted cell-state replacement and ablation, gate-activation analysis, and probes for individual trajectory states would be needed to support a mechanistic account.

Finally, the evidence about memory use is restricted to deterministic binary cellular automata and their delayed-recall extension; the language-modelling runs test training quality and cost, not memory use, and use one seed per configuration. The cellular-automaton experiments use a limited number of model-initialisation seeds and a shared data seed. Within each comparison, models share training schedules, data splits, and checkpoint candidates; the only difference in evaluation coverage is that final-test evaluation of R_6 BUT and CoTFormer stopped at depths 25 and 20, after every seed had reached 0% exact-row accuracy. Rule 30 provides a precise test of repeated transition and recall, but it does not reproduce the semantic variability, partial observability, stochastic dynamics, or open-ended outputs of natural-language reasoning and learned world models. The results therefore justify treating bounded gated memory as a strong candidate for those settings, not assuming that its advantage will transfer without direct evaluation.

8 Conclusion

Recurrent-depth models need their transition to keep working and their memory to stay usable as computation deepens. On Rule 30, a bounded gated cell provided both. Trained only to depth 12, every width-64 LSTM-UT seed kept at least 98.7% exact-row accuracy at depth 60, where BUT, even widened to match its parameter count, produced no correct rows. One checkpoint stayed above 99.95% through depth 1,000, $83\times$ its deepest supervised step. The same cell improved delayed recall over both baselines, including with 6.5% fewer parameters than BUT.

CoTFormer shows why memory *availability* is not memory *use*. Its cache holds every earlier state, yet it extrapolated least far, rarely selected the requested state during recall, and could pull a corrected trajectory back toward failure. In language modelling, LSTM-UT beat BUT and came within 0.6 perplexity of a parameter-matched CoTFormer, while CoTFormer needed up to 91% more training time per step.

What the cell stores remains open. Direct interventions on the cell, more seeds, compute-matched baselines, and tasks with stochastic or partially observed dynamics are needed to establish how far these advantages extend.

AI use statement

We used generative AI tools (large language model writing and coding assistants) in preparing this paper. For writing, they helped restructure and edit the manuscript, including drafting revised versions

of the abstract, introduction, section openings, and conclusion from our results, and checking the text for internal consistency. For code and analysis, they wrote the plotting scripts for Figures 1–3 and assisted with verification: recomputing reported statistics from logged run metrics, regenerating the Rule 30 data splits to test for overlap between training, validation, and test data, and the boundary-cell analysis of early-depth behaviour in Appendix A.3. AI coding assistants also helped implement parts of the training and evaluation code. All reported numbers come from our own training runs; AI-generated code and analyses were verified by comparing their outputs against the logged metrics and the reported tables. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Mujtaba Farhan and Maheep Chaudhary. Why limit the residual stream to layers and not tokens? persistent memory for continuous latent reasoning. *arXiv preprint arXiv:2606.07720*, 2026. URL https://arxiv.org/abs/2606.07720.

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

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In *Advances in Neural Information Processing Systems*, volume 38, pp. 41340–41391, 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Abstract-Conference.html.

Amirkeivan Mohtashami, Matteo Pagliardini, and Martin Jaggi. CoTFormer: A chain of thought driven architecture with budget-adaptive computation cost at inference. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=7igPXQFupX.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

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*, 2025. URL https://arxiv.org/abs/2502.17416.

Chengting Yu, Xiaobo Shu, Yadao Wang, Yizhen Zhang, Haoyi Wu, Jiaang Li, Rujiao Long, Ziheng Chen, Yuchi Xu, Wenbo Su, and Bo Zheng. MeSH: Memory-as-state-highways for recursive transformers. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=IhTrFvY7p3.

A ADDITIONAL EXPERIMENTAL DETAILS

Figure 4 visualises the increasing similarity of late recurrent states for the representative BUT run in Table 6.

A.1 TRAINING AND CHECKPOINT PROVENANCE

The Rule 30 initial rows are independent Bernoulli(0.5) samples with periodic boundaries. Training rows are materialised once from a fixed data seed and shuffled with a fixed shuffle seed; target rows are recomputed exactly for the selected depth. Each validation and final-test split contains 4,096

Table 6: A representative R_4 BUT's eventual failure (model seed 2, final test). Diagonal accuracy compares the decoded row with the exact target; transition consistency compares adjacent decoded rows under one exact Rule 30 update.

Repeat	Cell accuracy	Predicted ones	Hidden cosine	Transition consistency
5	100.00%	50.02%	0.5415	100.00%
15	95.84%	50.25%	0.7656	98.05%
28	49.99%	64.29%	0.9827	55.11%
36	49.76%	77.41%	0.9902	33.38%
40	50.06%	95.89%	0.9959	10.07%

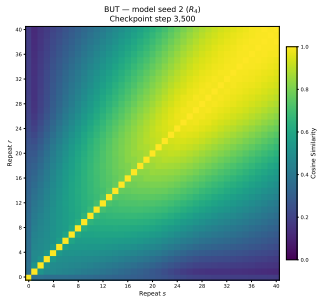

Figure 4: Pairwise cosine similarity between complete recurrent hidden states for the representative R_4 BUT run in Table 6. Late-depth states become increasingly similar as accuracy and decoded-transition consistency collapse. Similarity alone does not establish a latent attractor.

independently generated rows. Model-initialisation seeds vary within a comparison, while data, shuffle, validation, and test seeds are held fixed. Depth pairs are cycled round-robin by optimizer step rather than mixed within a batch. The R_4 comparison used learning rate 10^{-3}, weight decay 0.15, and gradient clipping at 0.9; the deeper regimes changed weight decay to 0.1. All runs used zero dropout. The unconstrained extrapolation checkpoints used in Tables 7–8 maximise validation cell accuracy at depth 9 for R_6 and at depth 15 for R_{12}. The final-test set did not select checkpoints. All depth-extrapolation results are reported over five model-initialisation seeds.

Table 7: R_6 Rule 30 depth extrapolation using the unconstrained extrapolation-selected checkpoint. Each result reports cell accuracy and exact-row accuracy, in percent, as mean $\pm$ sample standard deviation over five model-initialisation seeds. Depth 6 is the supervised boundary, depth 9 is used for checkpoint selection, and depths 12 and 20 test further extrapolation.

Architecture	d	Seeds	Metric	Depth 6	Depth 9	Depth 12	Depth 20
LSTM-UT	32	5	Cell	99.998 ± 0.004	99.990 ± 0.015	99.975 ± 0.041	99.790 ± 0.361
			Exact	99.922 ± 0.124	99.741 ± 0.422	99.370 ± 1.043	96.514 ± 6.249
LSTM-UT	64	5	Cell	100.000 ± 0.000	99.999 ± 0.001	99.977 ± 0.018	98.774 ± 1.042
			Exact	100.000 ± 0.000	99.951 ± 0.057	99.131 ± 0.647	74.863 ± 19.437
BUT	64	5	Cell	99.999 ± 0.002	99.980 ± 0.025	98.546 ± 2.264	65.705 ± 19.859
			Exact	99.956 ± 0.074	99.062 ± 1.181	67.056 ± 41.173	0.449 ± 0.655
CoTFormer	64	5	Cell	88.254 ± 20.780	64.430 ± 20.683	50.002 ± 0.057	50.001 ± 0.078
			Exact	51.787 ± 50.042	3.735 ± 8.353	0.000 ± 0.000	0.000 ± 0.000

We also ran an intermediate R_8 regime (width 64, five seeds). It showed the same ordering: at depth 60, LSTM-UT reached $73.9 \pm 42.8\%$ exact-row accuracy against 0% for BUT. We omit its table for space.

The long-horizon R_8 checkpoint (model seed 0) was selected at training step 8,250 by the strict extrapolation criterion; at depth 200 it retained 99.989% cell accuracy, corresponding to 29 incorrect cells in two of 4,096 rows. The R_{12} checkpoint (model seed 4) was selected at step 21,500 and evaluated with initial-state probabilities $p \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$ and 4,096 rows per condition.

Table 8: R_{12} Rule 30 depth extrapolation using the unconstrained extrapolation-selected checkpoint. Results are percentages reported as mean $\pm$ sample standard deviation over five model-initialisation seeds. Depth 12 is the supervised boundary, depth 15 selects the checkpoint on validation data, and depths 40 and 60 are beyond both training and checkpoint selection. The width-80 BUT and width-48 LSTM-UT rows approximately match the parameter count of width-64 LSTM-UT and BUT, respectively.

Architecture	d	Parameters	Seeds	Metric	Depth 12	Depth 20	Depth 40	Depth 60
LSTM-UT	64	82,368	5	Cell	100.000 ± 0.000	100.000 ± 0.000	99.971 ± 0.056	99.916 ± 0.152
				Exact	100.000 ± 0.000	99.995 ± 0.011	99.868 ± 0.230	99.673 ± 0.557
BUT	64	49,472	5	Cell	100.000 ± 0.001	99.783 ± 0.383	55.354 ± 10.616	50.053 ± 0.098
				Exact	99.995 ± 0.011	93.149 ± 11.626	0.000 ± 0.000	0.000 ± 0.000
BUT	80	77,200	5	Cell	99.933 ± 0.140	99.658 ± 0.556	57.979 ± 17.762	50.010 ± 0.037
				Exact	99.185 ± 1.690	92.959 ± 8.831	2.812 ± 6.289	0.000 ± 0.000
LSTM-UT	48	46,416	5	Cell	95.291 ± 5.414	91.484 ± 9.549	80.624 ± 18.941	72.009 ± 22.109
				Exact	54.644 ± 50.855	53.594 ± 49.864	44.316 ± 44.147	38.506 ± 44.505

At depth 1,000, its cell accuracy ranged from 99.973% to 100% and its exact-row accuracy from 99.951% to 100%; pooled across the 15 conditions, 342 of 3,932,160 cells and 10 of 61,440 rows were incorrect. Decoded recurrence exact-row accuracy was 100% at depth 1,000 in every condition.

For delayed recall, all models use AdamW with learning rate 10^{-3}, weight decay 0.15, gradient-norm clipping at 0.9, batch size 128, and no learning-rate schedule. Every training batch contributes evolution and query losses; intermediate evolution targets are exact Rule 30 rows, but clean states are never fed into the recurrence. The main sweep covers maximum horizons 6, 12, and 20 with three model-initialisation seeds. The source data seed is shared across seeds and architectures. The primary checkpoint maximises validation recall cell accuracy under the pair-balanced macro, with worst-query cell accuracy, exact-row accuracy, forward accuracy, and loss as deterministic tie-breakers. Test data do not select checkpoints.

A.2 Evaluation diagnostics

Early-depth non-monotonicity. Exact-row accuracy measures each depth separately and need not decrease monotonically. For the R_{12}, width-48 LSTM-UT checkpoint with model seed 0, accuracy rose from 23.29% at depth 1 to 100% at depth 2 on 4,096 final-test rows. All depth-1 errors occurred at the two boundary positions; interior accuracy was 100%. The same output head is used at every depth, while recurrence propagates continuous hidden and cell states rather than decoded bits. Appendix A.3 shows that these depth-1 outputs match Rule 30 computed with fixed zero edges instead of periodic ones, so the first repeat produces a consistent update that omits the wrap-around rather than unstructured readout errors. Why the wrap-around is present from the second repeat onward remains open.

Cell accuracy is the fraction of correctly predicted cells; exact-row accuracy requires all 64 cells to match. For delayed recall, queries are averaged within each horizon and the horizon means are weighted equally, excluding the no-op query $q = H$. Internal accuracy compares the recalled row with the model's own decoded row at requested repeat q, rather than the exact automaton row. It must be interpreted alongside ground-truth accuracy because a collapsed recurrent trajectory can yield high self-agreement without correct recall.

Changed-cell internal accuracy restricts that comparison to positions at which the model's requested-repeat and current-repeat decoded rows differ. On these cells, matching the current binary value necessarily fails to match the requested one. This makes the diagnostic more sensitive to selection than raw internal accuracy, though it can be unstable when few cells differ. The unique-best statistic compares the recalled output with every decoded evolution state in logit space and counts q only when it is strictly closer than every competing repeat. Ties are not successes. These diagnostics help separate forward-computation failure from failure to select an earlier computed state, but they do not reveal a unique internal mechanism.

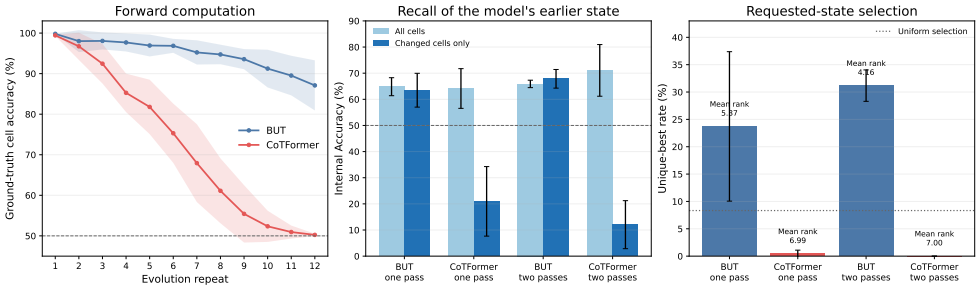

Figure 5: Horizon-12 diagnostics for one-application runs: same-checkpoint forward accuracy (left), raw and changed-cell internal recall accuracy (centre), and frequency that the requested repeat is the unique closest logit-space match (right). Means and sample SDs use three model seeds.

A.3 Early-depth non-monotonicity in Rule 30 extrapolation

In Figure 1 (right), the width-48 LSTM-UT and width-80 BUT means are lower at depths 1–2 than at later supervised depths. Each depth is a separate forward pass of exactly d block applications from the initial row, decoded only at the end; decoded rows are never fed back into the recurrence, so a depth-1 output error need not appear at depth 2.

The evaluator records accuracy at the two boundary cells (positions 0 and 63, whose periodic neighbours wrap around the row) separately from the 62 interior cells. Table 9 shows that every depth-1 error in the affected runs lies at a boundary cell: interior accuracy is exactly 100% over all 4,096 final-test rows. The errors match Rule 30 computed with fixed zero edges instead of periodic ones. Applied to the same final-test rows, that rule gives 23.36% exact-row accuracy, 61.68% boundary accuracy, and 0.766 errors per row; seed 0 of width-48 LSTM-UT gives 23.3%, 61.57%, and 0.769. Its logged test example is wrong only at cell 63, as the zero-edge rule predicts for that row. Seeds 1 and 3 match equally closely, while the remaining affected seeds lie between the zero-edge rule and the correct periodic update. From depth 2 onward, width-48 seed 0 is exact at both boundary and interior cells through depth 60.

Table 9: Depth-1 behaviour of the R_{12} runs in Figure 1 that are not exact at depth 1 (final test, unconstrained extrapolation checkpoint, 4,096 rows). Boundary accuracy averages cells 0 and 63; interior accuracy averages cells 1–62. All width-64 LSTM-UT seeds, width-80 BUT seeds 0–2, and width-64 BUT seeds 1–4 are exact at depth 1. The final row applies Rule 30 with zero edges to the same test rows.

Run	Depth-1 exact	Boundary	Interior	Errors/row	Depth-2 exact
LSTM-UT $d = 48$, seed 0	23.3	61.57	100.00	0.769	100.0
LSTM-UT $d = 48$, seed 1	26.0	62.77	100.00	0.745	42.2
LSTM-UT $d = 48$, seed 2	37.7	63.21	100.00	0.736	98.6
LSTM-UT $d = 48$, seed 3	25.4	61.88	100.00	0.762	17.1
LSTM-UT $d = 48$, seed 4	41.7	62.84	100.00	0.743	36.5
BUT $d = 80$, seed 3	50.0	62.39	100.00	0.752	72.9
BUT $d = 80$, seed 4	37.5	68.42	100.00	0.632	73.9
BUT $d = 64$, seed 0	74.8	87.40	100.00	0.252	100.0
Zero-edge Rule 30	23.36	61.68	100.00	0.766	–

The behaviour is not an artefact of checkpoint selection: the in-distribution-selected checkpoints of the same runs show the same depth-1 deficit. It is also present during training. In width-48 seed 0, the depth-1 training loss at step 23,000 is 0.0166, against 3.2×10^{-4} at depth 2, and validation exact-row accuracy at depth 1 stays between roughly 24% and 51% throughout training while depths 2 and 12 approach 100%. The R_6 width-48 LSTM-UT runs, which are not plotted, show the same pattern (seed 0: 25.2% depth-1 exact-row accuracy, 62.39% boundary accuracy, 100% interior accuracy).

We also checked that these curves are not affected by data leakage. Regenerating the training, validation, and final-test splits from their recorded seeds shows no shared rows among them, for both

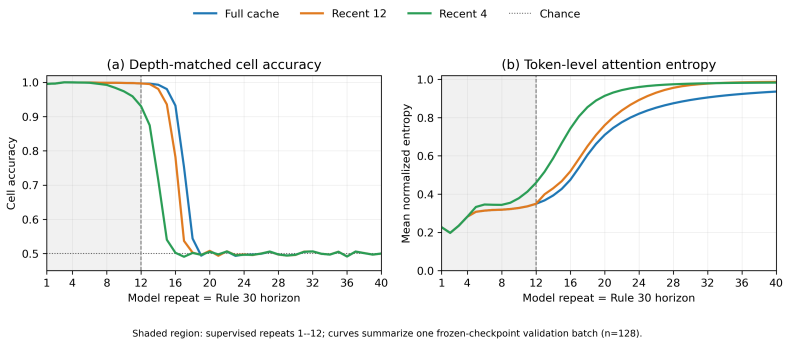

Figure 6: Representative CoTFormer accuracy and normalised attention entropy under inference-time cache masking. The frozen checkpoint was not retrained for the restricted windows, so this is a diagnostic intervention rather than a trained-policy comparison.

the 100,000-row R_6 and 500,000-row R_{12} training sets. No validation or test row occurs on any training row's Rule 30 trajectory through depth 60. The regenerated test and validation rows match the examples printed in the run logs. Checkpoint selection uses only the validation split, and targets at every depth are computed with the same periodic Rule 30 update in training and evaluation. These checks rule out leakage and target mismatch as explanations. They do not explain why these models omit the wrap-around at the first application but include it from the second onward.

A.4 Further CoTFormer checks

The R_{12} CoTFormer runs were trained for 24,000 optimizer steps on 500,000 initial rows, supervising depths 1 through 12. In one manual-attention checkpoint, the newest three cache blocks received 75.04% of attention mass at repeat 16, the earliest three received 16.24%, and the ten intermediate blocks only 8.72%. From repeat 12 to 40, normalised token entropy rose from 0.349 to 0.936 while mean maximum attention probability fell from 0.432 to 0.0027. These correlations do not establish whether attention flattening caused the representational failure or followed it. Masking the frozen checkpoint to recent-12 or recent-4 blocks failed to rescue extrapolation (Figure 6); this inference-time distribution shift is why separately trained cache-window models were also tested (Table 10).

Table 10: CoTFormer cell accuracy (%) under cache policies active throughout R_{12} training, mean $\pm$ sample SD. Checkpoints were selected by unseen-depth validation accuracy.

Training cache	Seeds	Depth 12 (trained)	Depth 17	Depth 20
Full	2	99.74 ± 0.09	65.63 ± 13.41	50.08 ± 0.01
Recent 1	3	99.99 ± 0.02	99.88 ± 0.14	99.39 ± 0.55
Recent 2	3	99.86 ± 0.10	62.36 ± 10.09	49.97 ± 0.10
Recent 3	3	83.25 ± 28.72	62.37 ± 18.85	49.95 ± 0.14
Recent 4	3	99.73 ± 0.40	75.94 ± 15.64	$50.08 \pm <0.01$
Recent 5	3	99.22 ± 1.21	70.33 ± 25.49	49.99 ± 0.12
Recent 6	3	83.15 ± 28.75	56.29 ± 10.25	50.04 ± 0.06
Recent 7	3	95.52 ± 6.99	64.74 ± 17.05	49.99 ± 0.08
Recent 8	3	83.23 ± 28.82	60.55 ± 10.69	49.99 ± 0.08

Factorial effects. Each intervention starts from a frozen checkpoint at source depth t and runs four paired conditions on the same 4,096-row validation split, sharing the free-running trajectory through depth t and continuing to depth 40: free baseline (FB: model state and cache kept), clean baseline (CB: hidden sequence replaced by the clean Rule 30 embedding, cache kept), free reset (FR: model state kept, cache cleared), and clean reset (CR: both). Clearing is a one-time operation; from repeat $t+1$ onward every condition accumulates keys and values normally under the model's trained cache policy. Writing $A_c(t, t+j)$ for cell accuracy of condition c at depth $t+j$, the three columns of

Table 1 are the five-repeat paired differences

$$\Delta_{\text{state}} = \tfrac{1}{5}\sum_{j=1}^{5}\big[A_{\text{CB}}(t,t+j) - A_{\text{FB}}(t,t+j)\big],$$
$$\Delta_{\text{reset|clean}} = \tfrac{1}{5}\sum_{j=1}^{5}\big[A_{\text{CR}}(t,t+j) - A_{\text{CB}}(t,t+j)\big],$$
$$\Delta_{\text{reset|free}} = \tfrac{1}{5}\sum_{j=1}^{5}\big[A_{\text{FR}}(t,t+j) - A_{\text{FB}}(t,t+j)\big],$$

labelled "State", "Reset after clean", and "Reset alone". The total effect of the joint intervention is therefore $\Delta_{\text{state}} + \Delta_{\text{reset|clean}}$. Each run's effects are first averaged over its tested source depths, and the table reports the mean and sample standard deviation across runs, so runs with more tested depths do not receive more weight. The rows contain 3 runs and 16 interventions (recent 2), 2 and 10 (recent 3), 3 and 9 (recent 4), 3 and 10 (recent 5), 2 and 7 (recent 8), and 2 and 6 (full cache). One recent-8 run is excluded because its selected checkpoint was already at chance within the supervised depths, so it had no extrapolating trajectory to rescue. No factorial grid was collected for recent-1, recent-6, or recent-7 models.

For the state–cache factorial, the first post-intervention prediction occurs at depth $t+1$. A recent-2 run that had collapsed before intervention at $t = 23$ regained 99.33% cell accuracy at depth 24 from state correction alone and retained 96.36% at depth 35; joint state correction and cache reset reached 99.82% at depth 35. A recent-8 run differed: at $t = 18$, neither state correction nor reset alone helped, whereas their combination reached perfect cell and exact-row accuracy at depth 19 and 99.88% cell accuracy at depth 30. Both interventions eventually failed by depth 40. Intervention depths were chosen near observed failure regions and are not balanced across cache policies.

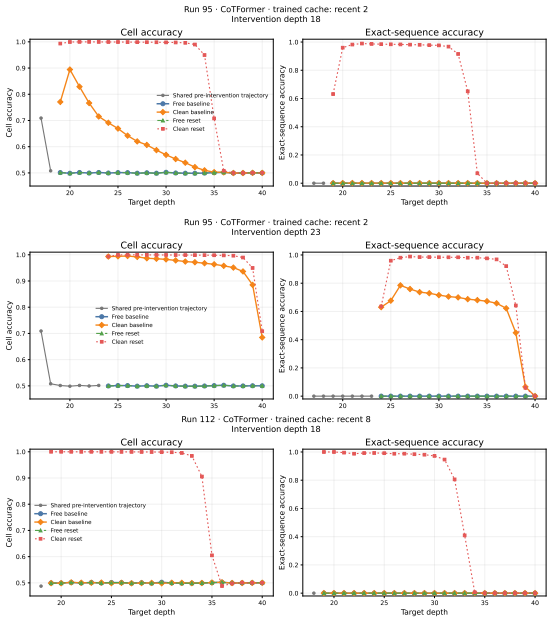

Figure 7: Representative factorial state–cache interventions: recent-2 run 95 at depths 18 and 23 (top and middle), and recent-8 run 112 at depth 18 (bottom). Clean-state injection rescues the recent-2 model temporarily; the recent-8 model requires both state correction and cache reset. These are selected examples, and neither intervention makes the transition indefinitely stable.

In the delayed-recall cache analysis, a single attention head did not recover requested-repeat addressing. Table 11 averages seven non-current horizon-20 queries within each seed and then across three seeds. With two recall applications, internal agreement could exceed 95% for near-current queries even when ground-truth accuracy was approximately 50%, reinforcing the need for selection-sensitive diagnostics. These attention and intervention measurements use selected validation checkpoints and a manual-attention implementation; they are mechanistic evidence for this setting, not population-level estimates.

Table 11: Attention to the controller-indexed evolution block in single-head CoTFormer, mean $\pm$ sample SD over three seeds. Uniform selection among 20 evolution blocks would yield 5% most-attended frequency and mean rank 10.5.

Recall applications	Requested mass	Most attended	Mean rank
1	$4.73 \pm 1.04\%$	$4.66 \pm 8.07\%$	11.80 ± 1.76
2	$3.28 \pm 0.75\%$	$2.31 \pm 2.20\%$	13.37 ± 0.63