SCALING TO TENS OF THOUSANDS OF TEST-TIME ITERATIONS WITH LOOP-NATIVE ATTENTION RESIDUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we argue that looped Transformers need their own residual connections to prevent performance degradation as the number of iterations grows. We observe that increasing loop iterations can reduce reasoning accuracy: noisy state updates overwrite correct intermediate deductions and even undo completed solutions. This leaves subsequent iterations to recover lost information from an already degraded representation: once an error arises in an earlier loop, often as a result of long-range propagation through the recurrence, later loops find it difficult to correct. In this paper, we introduce `InfiLoop`, a loop-native residual connection that learns which past computations to retain and how much to accept from each new update. InfiLoop combines content-based weighting with learned temporal decay to maintain a running summary of recurrent states. An exact streaming recurrence keeps its persistent aggregation memory constant as the loop count grows. The resulting adaptive update suppresses unreliable proposals and preserves useful intermediate states. Across extensive reasoning tasks, a 7M-parameter InfiLoop model outperforms existing recursive architectures, reaching 97.9% exact accuracy on Sudoku-Extreme, and 13.6% pass@2 on ARC-AGI-2. It is worth noting that on Sudoku-Extreme, `InfiLoop` scales test-time looping beyond 20,000 effective steps, effectively translating greater computational depth into higher reasoning capacity.

1 INTRODUCTION

Looped Transformers address complex reasoning tasks by repeatedly applying a shared Transformer block (Dehghani et al., 2019; Wang et al., 2025; Jolicoeur-Martineau, 2025). Each iteration adds another sequence of layer transformations, increasing computational depth without introducing additional parameters. This allows models to devote more computation to reasoning by executing more loops at test time (Saunshi et al., 2025; Geiping et al., 2025). A common way to connect consecutive iterations is to feed the final output of one loop directly into the next, a strategy we refer to as the *carry-last rule*. Although effective, this rule makes the unrolled computation equivalent to a deep network with weights shared across repeated blocks. Increasing the loop count therefore returns us to a familiar question: *does adding more depth lead to more useful computation?*

Standard deep Transformers suffer from the *curse of depth* (Sun et al., 2025), where the contribution of deeper layers progressively diminishes. Repeatedly applying a shared block can compound this limitation, as the same transformations are applied at every iteration and the residual stream grows with the loop count. Related limitations have also been examined in fixed-point looping (Movahedi et al., 2026). Existing remedies use decay or layer scaling to control residual growth and regulate update magnitudes (Sun et al., 2025; Wang et al., 2026; Movahedi et al., 2026). These interventions address how strongly each update affects the representation, but leave a central architectural choice unchanged: the next loop still receives only the latest recurrent state.

We argue that this state-replacement rule is a key bottleneck in deep looped transformers. Under carry-last, earlier computations remain accessible only through the information preserved in the latest state. Standard Pre-LN or Post-LN residual connections provide paths for this information to propagate, but do not explicitly retain and select among outputs from earlier loops. Consequently, a noisy or undesirable iteration can degrade useful representations throughout propagation, leading to sub-optimal performance, as shown in Figure 1. This motivates changing how information is combined across loops, so that further computation can build on useful intermediate states without relying exclusively on the most recent output.

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

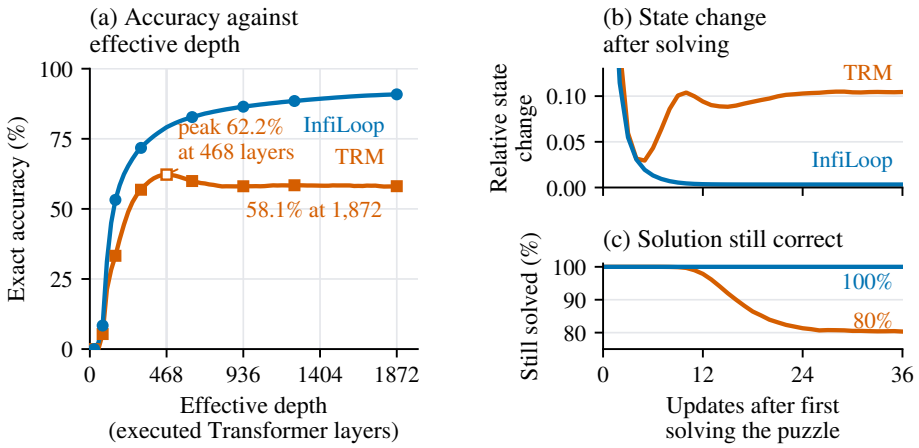

Figure 1: **Carry-last keeps changing its state after solving and loses solutions.** (a) Exact accuracy against effective depth, the number of executed Transformer layers, on Sudoku-Extreme. Each outer update executes 26 layers. TRM peaks at 468 layers and then loses accuracy as depth grows. (b) Relative state change after the update at which each model first solves a puzzle, for the puzzles that both models solve within 36 updates. TRM settles briefly and then moves its state again. (c) Share of those puzzles whose solution is still correct.

To this end, we introduce **InfiLoop**, a loop-native residual connection that replaces the carry-last rule with learned attention over recurrent outputs. A shared pseudo-query scores each candidate when it is produced, while a learned exponential decay discounts earlier contributions. The resulting normalized aggregation forms the input to the next iteration. Crucially, shared scoring and geometric decay admit an exact streaming recurrence, requiring only a weighted-state vector and a normalization scalar per token, with persistent aggregation memory independent of the loop count. This recurrence is equivalent to an adaptive residual update that balances the accumulated state against each new candidate, allowing the model to regulate how strongly subsequent computations alter its reasoning state.

This paper establishes four findings. First, empirical diagnosis reveals that carry-last destabilizes solved states at large depth, while InfiLoop preserves solutions indefinitely. Second, the exact recurrence guarantees constant-memory state combination across arbitrarily long reasoning trajectories. Third, on extensive reasoning benchmarks, InfiLoop surpasses strong damping baselines while requiring substantially fewer layer evaluations. Fourth, inspection of the learned step size reveals an emergent self-verification behavior: the connection discounts outputs produced during unsolved search, accepts the output that completes the solution, and freezes the state once solved.

2 Deeper Loops Undo Earlier Progress

We measure where a looped reasoner breaks as depth grows. On Sudoku-Extreme, we compare InfiLoop with the self-attention TRM, a carry-last model. Both checkpoints are trained with the same loop schedule, and we run both for 72 outer updates. After each outer update, we decode the predicted board and measure the relative change of the outer state, $\|\mathbf{h}_t - \mathbf{h}_{t-1}\|/\|\mathbf{h}_{t-1}\|$. We average this change over initially empty cells. Figure 1 plots exact accuracy against effective depth, the number of executed Transformer layers, and tracks the state change after solving.

Accuracy peaks and then falls with depth. TRM reaches its best exact accuracy of 62.2% at 468 layers, but its accuracy declines to 58.1% at 1,872 layers. InfiLoop keeps improving over the same depth and reaches 90.9%. The predictions show the same pattern. During updates 36 to 72, TRM changes 77.0 predictions per puzzle on initially empty cells, and its number of incorrect cells rises by 0.21 per puzzle. Repeated changes to the same cell count separately. Over the same window, InfiLoop makes 24.0 changes and removes 1.25 incorrect cells per puzzle.

Correct intermediate predictions are lost. TRM overwrites correct predictions in later updates. At update 36, both models predict 1.42 million initially empty cells correctly. During the remaining updates, TRM turns 20.9% of these predictions into an incorrect value at least once, and InfiLoop turns 2.4% of them. The right two panels of Figure 1 align each puzzle to its first solving update.

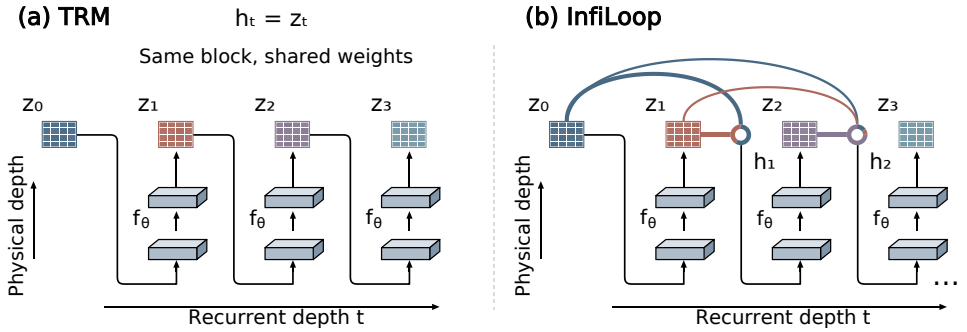

Figure 2: **Connecting successive applications of a shared block.** (a) TRM uses the carry-last rule: the candidate z_t becomes the next recurrent state, $h_t = z_t$. (b) InfiLoop combines earlier candidates into the state h_t passed to the next block application, using content weights and exponential temporal decay. Colors identify candidates from different iterations; the illustrated contributions are schematic. The same block parameters are reused at every recurrent step.

Within 5 updates after solving, TRM reduces its state change to 3% of the state norm, but the change then climbs back to 10% per update and stays there. TRM begins to lose solutions after 9 updates and has lost 19.7% of them after 36 updates. The state change of InfiLoop instead falls to 0.3% per update, and every puzzle solved by InfiLoop remains solved 36 updates later.

Learning what to carry forward. A newer output is not necessarily a better output, yet carry-last passes every new block output to the next iteration in full. Earlier outputs reach later iterations only through the newest output, so the connection between iterations decides what the loop keeps. We therefore replace carry-last with a learned residual connection. A running summary holds the contributions of earlier outputs, and the connection learns how much of each new output to add to it. The summary becomes the input to the next iteration, so the learned connection acts during reasoning and not on predictions after reasoning ends. A shared scorer weights each output once, at its own iteration, and a learned decay reduces the contributions of older outputs. The weighted summary has an exact recurrent form, and its persistent memory does not grow with the number of iterations. Section 3 defines the connection and derives its recurrence.

3 Loop-Native Attention Residuals

A looped Transformer applies a shared block f_θ for T recurrent steps, conditioned on an input representation x. Under the commonly used *carry-last* rule, each step produces a candidate state z_t and passes it directly to the next iteration:

$$z_t = f_\theta(h_{t-1}; x), \qquad h_t = z_t, \tag{1}$$

where h_0 is the initial recurrent state. Earlier computations remain accessible only through the information retained in the latest state. Figure 2 contrasts this connection with the learned aggregation used by InfiLoop.

3.1 Attention Residuals Across Recurrent Steps

Attention Residuals constructs the residual stream by attending over representations from earlier layers (Kimi Team et al., 2026). We adapt this principle to recurrent steps, combining content-based weighting with temporal decay. The aggregation operates independently at each token position; below, we omit the token index and write $z_i, h_t \in \mathbb{R}^d$, where d is the hidden dimension. Let $z_0 = h_0$ denote the initial representation and z_i the candidate produced at step $i \geq 1$. The state passed to the next iteration is

$$h_t = \sum_{i=0}^{t} \alpha_{t,i} z_i, \qquad \alpha_{t,i} = \frac{\exp(s_i - \lambda(t-i))}{\sum_{j=0}^{t} \exp(s_j - \lambda(t-j))}. \tag{2}$$

Here $\alpha_{t,i}$ is the normalized weight assigned to candidate z_i at step t. Its content score is $s_i = q^\top \mathrm{RMSNorm}(z_i)$, where $q \in \mathbb{R}^d$ is a learned pseudo-query shared across iterations and $\mathrm{RMSNorm}(\cdot)$ denotes RMS normalization (Zhang & Sennrich, 2019). Each candidate is therefore scored once, when it is produced. The temporal bias $-\lambda(t-i)$ discounts older candidates using a

learned decay rate $\lambda > 0$. The unnormalized weight factors as $\exp(s_i)\exp(-\lambda(t-i))$, combining content-based relevance with recency. In the no-decay case $\lambda = 0$, the aggregation recovers the Full Attention Residuals rule over recurrent outputs, with the pseudo-query tied across iterations.

Equivalently, the aggregated state can be expressed as the ratio of an unnormalized weighted sum $N_t \in \mathbb{R}^d$ to a normalization mass $D_t \in \mathbb{R}_{>0}$:

$$N_t = \sum_{i=0}^{t} \exp(s_i - \lambda(t-i)) z_i, \qquad D_t = \sum_{i=0}^{t} \exp(s_i - \lambda(t-i)), \qquad h_t = \frac{N_t}{D_t}. \tag{3}$$

These quantities admit an exact streaming update, derived below.

3.2 Exact Recurrence with Constant Memory

Directly evaluating the sums in Equation 3 would require retaining the full candidate history. The shared scorer and exponential temporal decay make this unnecessary: historical content scores remain unchanged, and their weights decay by the same factor at each step. Specifically, let $e_i = \exp(s_i)$ and $\beta = e^{-\lambda} \in (0, 1)$, so that $\exp(s_i - \lambda(t-i)) = e_i \beta^{t-i}$. Separating the newest contribution gives

$$\begin{aligned} N_t &= \sum_{i=0}^{t-1} \beta^{t-i} e_i z_i + e_t z_t = \beta N_{t-1} + e_t z_t, \\ D_t &= \sum_{i=0}^{t-1} \beta^{t-i} e_i + e_t = \beta D_{t-1} + e_t. \end{aligned} \tag{4}$$

We initialize $N_0 = e_0 z_0$ and $D_0 = e_0$, which yields $h_0 = z_0$, and recover the state at each step as $h_t = N_t / D_t$. The recurrence is algebraically identical to Equation 2; no approximation to the attention weights is introduced.

3.3 Interpretation as an Adaptive Residual Update

Substituting $N_{t-1} = D_{t-1} h_{t-1}$ into Equation 4 and dividing by D_t yields

$$h_t = h_{t-1} + \gamma_t (z_t - h_{t-1}), \qquad \gamma_t = \frac{e_t}{\beta D_{t-1} + e_t} \in (0, 1). \tag{5}$$

The attention aggregation thus implements an adaptive step toward the new candidate. The step size γ_t compares the candidate's content weight e_t with the retained historical mass βD_{t-1}. When $e_t \gg \beta D_{t-1}$, γ_t approaches one and the new candidate dominates. When $e_t \ll \beta D_{t-1}$, γ_t approaches zero and the previous state is largely preserved. The learned scorer can therefore suppress low-scoring proposals while accepting high-scoring ones. Because the attention weights are positive and sum to one, h_t lies in the convex hull of the candidate states, i.e., $\|h_t\| \le \max_{0 \le i \le t} \|z_i\|$. Thus, aggregation does not amplify the norm beyond that of the largest candidate. A uniform bound over arbitrarily many iterations additionally requires the candidate norms to remain bounded.

Moreover, our framework applies to both standard looped Transformers (Dehghani et al., 2019) and nested recurrent reasoners like TRM (Jolicoeur-Martineau, 2025) and HRM (Wang et al., 2025). For a standard looped Transformer, we replace the carry-last assignment with Equation 4. For nested looped architectures, we apply the same rule independently to each recurrent axis. The inner and outer states each have a separate scorer–decay pair (q, β) and accumulators (N, D), adding $d + 1$ learned parameters per axis while keeping the aggregation state independent of the loop count.

3.4 Computational and Memory Complexity

Our aggregation stores only a vector $N_t \in \mathbb{R}^d$ and a scalar D_t per token, requiring $O(d)$ persistent memory and $O(d)$ computation per iteration. Over T iterations, its total aggregation cost is $O(Td)$, while its memory footprint remains independent of the loop count. In contrast, Full Attention Residuals with iteration-specific queries stores and attends over all preceding representations, requiring $O(td)$ memory and computation at step t (Kimi Team et al., 2026). Our recurrence therefore reduces peak persistent memory from $O(Td)$ to $O(d)$ and total aggregation cost from $O(T^2 d)$ to $O(Td)$.

4 Experiments

We evaluate InfiLoop against recursive and looped reasoning models across symbolic puzzle benchmarks and abstract visual transformation tasks. Our implementation builds upon the public TRM

codebase (Jolicoeur-Martineau, 2025) and adopts its deep-supervision training procedure. We first describe the benchmarks and evaluation protocol, then report complete system performance, examine test-time compute and depth scaling, and analyze the emergent verification dynamics of the learned connection.

4.1 Dataset description

In the following, we provide a structured description of each dataset used in this paper. We refer the reader to the cited literature for further information.

Sudoku-Extreme. The task consists of exceptionally challenging, partially filled 9×9 Sudoku puzzles with unique solutions, introduced by Wang et al. (2025). Each board is flattened into a sequence of length 81. The training set contains 1,000 unique base puzzles, each augmented 1,000 times through digit permutations and spatial symmetries, yielding approximately 1M training samples. The test data contains 422,786 unique puzzles. The evaluation metric is exact sequence accuracy, requiring all 81 board cells to match the ground truth.

Maze-Hard. The task consists of 30×30 shortest-path maze puzzles, introduced by Wang et al. (2025). Each grid contains open navigable cells, impassable walls, an origin, and a destination. The model predicts a flattened grid marking the reference shortest path. The training and test sets each contain 1,000 unique samples. Following Wang et al. (2025), we apply no data augmentation to this task. The evaluation metric is exact sequence accuracy.

ARC-AGI-1 and ARC-AGI-2. The Abstraction and Reasoning Corpus assesses whether a reasoning system can induce an underlying transformation rule from minimal demonstrations (Chollet, 2019; Chollet et al., 2024). ARC-AGI-1 provides two or more demonstration pairs (typically three) of coloured 2D grids with variable dimensions up to 30×30, and requires generating the output grid for a held-out test input. ARC-AGI-2 keeps this format while presenting substantially more intricate problems that compose multiple transformations and resist brute-force search (Chollet et al., 2025). ARC-AGI-1 has 400 public training tasks and 400 public evaluation tasks; ARC-AGI-2 has 1,000 public training tasks and 120 public evaluation tasks.[1] The evaluation metric is pass@2 accuracy, selecting the top two candidate outputs aggregated across augmented test-time views. Pretrained reasoning language models with chain-of-thought struggle on both benchmarks: DeepSeek-R1 with 671B parameters attains 15.8% on ARC-AGI-1 and 1.3% on ARC-AGI-2, while Claude 3.7 Sonnet (16K) attains 28.6% and 0.7%, respectively, as reported in Table 1.

4.2 Evaluation protocol

Model architecture. We evaluate a compact 7M-parameter recursive reasoner that implements the nested looping architecture in Section 3.3. Its shared recursive block contains two Transformer layers with hidden dimension 512, eight attention heads, SwiGLU activations (Shazeer, 2020) with an expansion factor of four, and a depthwise 3×3 convolution. Across both the inner and outer recurrent loops, the model replaces standard carry-last state replacement with our loop-native attention residual connection, parameterized by a 513-parameter scorer–decay pair (q, β) per axis.

Baselines. We benchmark InfiLoop against the primary looped reasoning paradigms: (1) **TRM-Att** (7M parameters) (Jolicoeur-Martineau, 2025), the standard self-attention recursive reasoner using carry-last state propagation; (2) **FPRM** (7M parameters) (Movahedi et al., 2026), a released looped model that stabilizes iterations via pre-norm, learned residual scaling, and damped fixed-point updates; (3) **HRM** (Wang et al., 2025), **Attractor Models** (Fein-Ashley & Rashidinejad, 2026), **EqR** (Huang et al., 2026), and **URM** (Gao et al., 2025), spanning parameter budgets from 7M to 27M; and (4) **Frontier reasoning LLMs** using test-time chain-of-thought. For direct controlled comparisons, we evaluate the official released checkpoint of FPRM on Sudoku-Extreme and retrain the TRM baseline with the authors' published recipe.

Training configuration. For Sudoku-Extreme, we train InfiLoop on 1,000 base puzzles with 1,000 augmentations per puzzle. We optimize using AdamW (Loshchilov & Hutter, 2019) with peak learning rate 10^{-4}, weight decay 1.0, and batch size 768. The training schedule executes $H = 3$ outer loops and $L = 6$ inner updates across 12 deep-supervision segments. Each segment executes 21 shared-block applications, with gradients backpropagated through the final seven. For Maze-Hard and

[1] Official ARC-AGI-2 dataset: `https://github.com/arcprize/ARC-AGI-2`.

Table 1: Reasoning benchmark results (%) for complete systems under their respective inference procedures. Sudoku and Maze use exact solution accuracy. ARC uses pass@2. Best results are bold, and the best among the listed 7M-parameter models are underlined. Results for DeepSeek R1 (DeepSeek-AI, 2025), Claude 3.7 Sonnet (Anthropic, 2025), o3-mini (OpenAI, 2025), and Gemini 2.5 Pro (Comanici et al., 2025) follow Jolicoeur-Martineau (2025); URM and EqR parameter counts follow Movahedi et al. (2026). ‡URM reports ARC pass@1. †Reproduced by us: TRM retrained on Sudoku-Extreme with the authors' published recipe, and the released ARC-AGI-2 verification checkpoint evaluated under our protocol.

		Puzzles (exact acc.)		ARC-AGI (pass@2)	
Method	Params	Sudoku-Ext.	Maze-Hard	ARC-1	ARC-2
Large language models					
DeepSeek R1	671B	0.0	0.0	15.8	1.3
Claude 3.7 Sonnet	–	0.0	0.0	28.6	0.7
o3-mini-high	–	0.0	0.0	34.5	3.0
Gemini 2.5 Pro (32K)	–	–	–	37.0	4.9
Reproduced by us					
TRM†	7M	73.6	–	–	4.6
Recursive reasoning models (reported)					
HRM	27M	55.0	74.5	40.3	5.0
Attractor Model	27M	91.4	**93.1**	–	–
URM	14M	77.6	–	$\geq$**53.8**‡	$\geq$**16.0**‡
TRM	7M	74.7	85.3	44.6	7.8
EqR	7M	93.0	–	–	–
Attractor Model	7M	54.3	46.7	–	–
FPRM	7M	94.2	87.0	47.5	6.2
InfiLoop (ours)	7M	**97.9**	87.4	49.1	13.6

ARC benchmarks, training employs $H = 3, L = 6$ across up to 16 supervision segments with peak learning rate 3×10^{-4}. Appendix A provides complete hyperparameter and resource accounting.

4.3 Reasoning performance across puzzle benchmarks

The complete InfiLoop reasoner achieves the highest performance among the 7M-parameter configurations listed in Table 1 across the four benchmarks. On Sudoku-Extreme, InfiLoop achieves 97.9% exact accuracy, surpassing FPRM at 94.2% and reported TRM-Att at 74.7%. On Maze-Hard, InfiLoop achieves 87.4% accuracy, exceeding both TRM-Att at 85.3% and FPRM at 87.0%. On ARC-AGI-1, InfiLoop reaches 49.1% pass@2, improving upon FPRM at 47.5% and TRM-Att at 44.6%. On the challenging ARC-AGI-2 benchmark, InfiLoop attains 13.6% pass@2, nearly doubling TRM-Att at 7.8% and outperforming FPRM at 6.2%. We also reproduce the TRM baseline ourselves, marked † in Table 1. Retraining TRM on Sudoku-Extreme with the authors' published recipe gives 73.6%, close to the 74.7% they report. Evaluating the released ARC-AGI-2 verification checkpoint under our protocol yields 4.6% pass@2. All baseline comparisons in the following sections use the retrained checkpoint.

4.4 Test-time depth and compute efficiency

We next evaluate how InfiLoop translates recurrent computation into reasoning accuracy. We conduct controlled evaluations on a fixed sample of Sudoku-Extreme test puzzles across varying layer budgets without retraining. With two Transformer layers per shared block, executing S supervision segments, H outer cycles, and L inner updates corresponds to $2SH(L + 1)$ layer evaluations.

Comparison with FPRM under test-time scaling. Figure 3 (left) compares InfiLoop against the released FPRM checkpoint and retrained TRM across executed layers up to 24,960. At 1,872 layer evaluations, InfiLoop achieves 90.9% exact accuracy overall. This exceeds FPRM's 90.1% accuracy at 3,744 layer evaluations, cutting the required computation in half. When matched at 3,744 layer evaluations, InfiLoop reaches 92.4% overall, maintaining a steady advantage over FPRM at 90.1%. Furthermore, the accuracy that InfiLoop achieves at 936 layer evaluations (86.5%) is never matched by TRM even when evaluated at 24,960 layers (83.65%), more than 26 times deeper. At 24,960 layer evaluations, exact accuracy is 92.74% for InfiLoop, 91.63% for FPRM, and 83.65% for TRM.

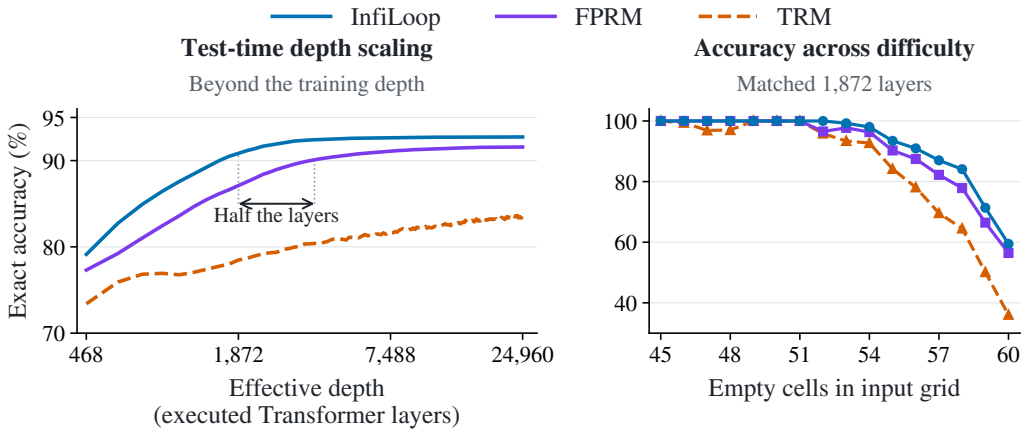

Figure 3: **Test-time scaling and accuracy across puzzle difficulty. Left:** Exact accuracy against executed Transformer layers (logarithmic axis), with all three models evaluated to 24,960 layers. InfiLoop reaches 90.9% at 1,872 layers, exceeding FPRM's 90.1% at 3,744 layers with half the layer evaluations. **Right:** Exact accuracy by empty-cell count at matched 1,872 layers.

Accuracy across puzzle difficulties. Figure 3 (right) evaluates accuracy as a function of the number of empty cells in the input grid, an established proxy for Sudoku difficulty. We evaluate all difficulty levels from 45 to 60 empty cells on the 32,768 test puzzles at matched 1,872 layer evaluations. For easy puzzles with 45 to 51 empty cells, both InfiLoop and FPRM achieve 100.0% exact accuracy. From 52 empty cells onward, the margin of InfiLoop over FPRM stays between 1.5 and 6.2 percentage points, while the margin over TRM rises from 4.2 percentage points at 52 empty cells to 23.4 at 60. At 52 empty cells, InfiLoop attains 100.0% accuracy against 96.5% for FPRM (+3.5 percentage points) and 95.8% for TRM (+4.2 percentage points). At 55 empty cells, InfiLoop scores 93.4% against 90.3% for FPRM (+3.2 percentage points) and 84.2% for TRM (+9.3 percentage points). At 58 empty cells, the margin over FPRM peaks at $+6.2$ percentage points (84.1% versus 77.9%), while the margin over TRM widens to $+19.6$ percentage points. At the most difficult count of 60 empty cells, InfiLoop maintains 59.5% accuracy against 56.5% for FPRM (+3.0 percentage points) and 36.0% for TRM (+23.4 percentage points). Learned state reuse shields intermediate deductions precisely when search paths are deep and complex.

4.5 What the connection learns to carry

Section 2 demonstrated that carry-last destabilizes solutions because later iterations overwrite earlier correct states. Equation 5 reveals how InfiLoop regulates state propagation through the adaptive step size $\gamma_t = e_t/(\beta D_{t-1} + e_t)$. If the scorer assigned equal weight to every block output, γ_t would converge to $1 - \beta = 0.56$. Any divergence from 0.56 reflects learned content preferences. Figure 4 analyzes how the step size and write scores behave across test puzzles.

The step size surges when the solution appears. Figure 4a aligns puzzles solved at update 13 or later to their solving update. Prior to finding the solution, the mean step size remains suppressed at approximately 0.40, below the equal-weight baseline of 0.56. At the exact update that completes the puzzle, the step size surges to 0.91, giving dominant weight to the newly found solution. Within six subsequent updates, it returns to the baseline value of 0.56. This surge is strictly contingent on finding the solution; for puzzles that remain unsolved, the step size stays suppressed across all iterations.

The content score reflects solution correctness. The scorer receives no external hints or task labels during reasoning. Figure 4b plots the clipped content score $\ln e_t = \mathrm{clip}(s_t, -8, 8)$ for outputs after update 12 against the number of wrong cells in the decoded board. The candidate states that decoding to a fully correct board achieves a median score of 7.25, close to the upper clipping bound of 8.0. The assigned score drops monotonically as prediction errors increase: the median score falls to 2.75 at 20 wrong cells, and drops below zero to -0.58 at 33 wrong cells. Valid candidate states receive high write weights and dominate the recurrent summary. Candidate states with errors receive lower weights, limiting their influence on the reasoning trajectory. Appendix D shows that removing the content score, during training or at test time, lowers accuracy by 6 to 7 percentage points.

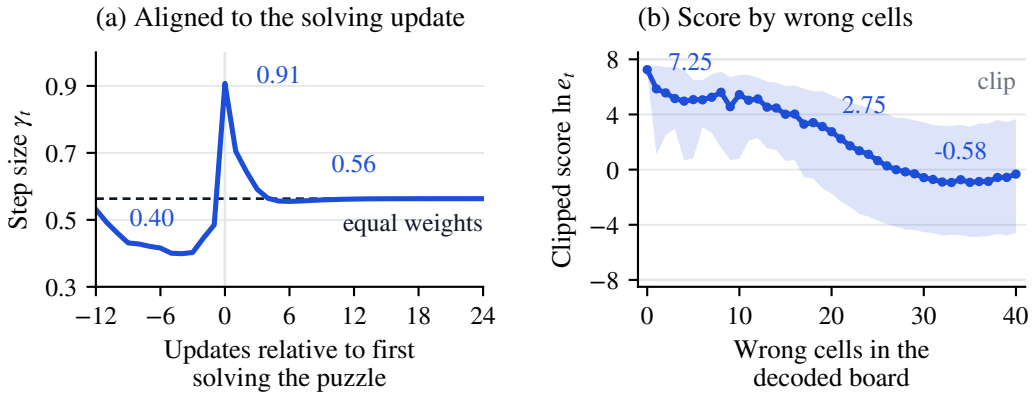

Figure 4: **Learned adaptive updates reflect solution milestones and board correctness. (a)** Mean adaptive step size γ_t aligned to the update at which the puzzle is first solved (for puzzles solved at update 13 or later). The step size remains suppressed ($\gamma_t \approx 0.40$) during search, peaks at 0.91 when the puzzle is completed, and settles to the equal-weight baseline $1 - \beta = 0.56$ thereafter. **(b)** Median clipped content score $\ln e_t = \text{clip}(s_t, -8, 8)$ for block outputs produced after update 12, plotted against the number of incorrect cells in the decoded board. The shaded band denotes the interquartile range. High weights are assigned exclusively to valid configurations without explicit correctness supervision.

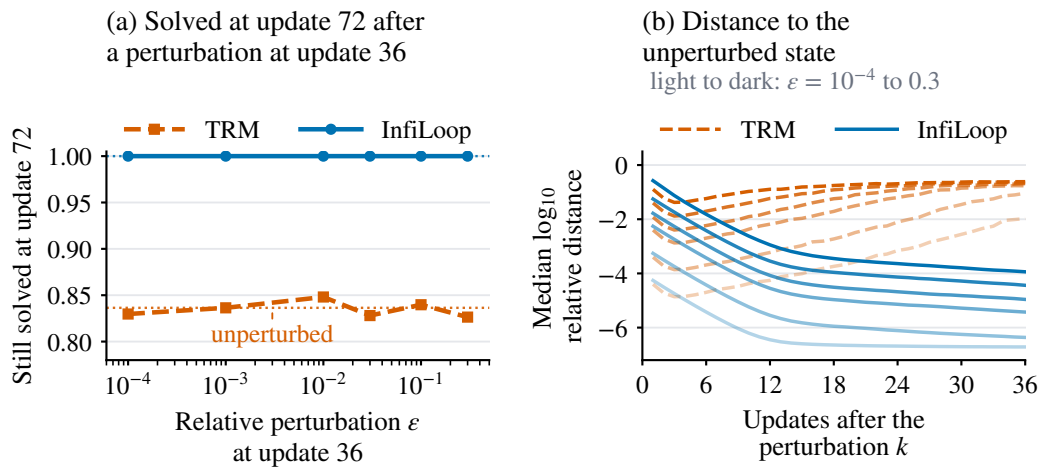

Figure 5: **State stability and solution retention under recurrent state perturbation.** At outer update 36, a relative perturbation $\epsilon \in [10^{-4}, 0.3]$ is injected into the recurrent state, followed by 36 additional updates. **(a)** Proportion of puzzles solved at update 36 that remain solved at update 72 as a function of perturbation magnitude ϵ. The horizontal dashed line marks the unperturbed baseline. **(b)** Evolution of the median relative Euclidean distance between the perturbed and unperturbed states, $\log_{10}(\|h'_{36+k} - h_{36+k}\|/\|h_{36+k}\|)$, across subsequent updates k. Different line shades denote perturbation magnitudes ϵ. While perturbations in TRM quickly diverge to $O(1)$ relative distance, InfiLoop constrains trajectory drift and retains all solved states.

State stability under recurrent perturbation. Because candidate states with constraint violations receive small content weights e_t, the recurrence limits the influence of noisy updates on the accumulated state. To test whether this mechanism stabilizes the trajectory against perturbations, we inject a relative perturbation $\epsilon \in [10^{-4}, 0.3]$ into the recurrent state at update 36 and unroll the model for another 36 updates. As shown in Figure 5a, InfiLoop retains 100% of its solved puzzles at update 72 across all tested perturbation scales, while TRM loses 15–20% of solutions even without perturbation. Figure 5b tracks the relative Euclidean distance to the unperturbed trajectory over subsequent updates k. For TRM, even a minimal perturbation of $\epsilon = 10^{-4}$ causes the trajectory to diverge to a relative distance of order one within 12 updates. In contrast, InfiLoop keeps the displacement bounded throughout subsequent iterations, showing that the learned aggregation prevents chaotic drift in the recurrent state.

Table 2: **Component and initialization ablations on Sudoku-Extreme.** Exact solution accuracy (%) on 7,680 test puzzles. Top: placement of the connection across recurrent axes at training depth $H3\,L6$ and three larger budgets. Bottom: temporal decay on the two-axis model. The left group fixes β during training. The right group learns β from the initialization β_0.

Connection placement across budgets				
Configuration	$H3\,L6$	$H3\,L12$	$H3\,L96$	$H32\,L6$
Inner axis alone	72.6	75.2	77.2	79.4
Outer axis alone	75.1	76.3	76.2	80.0
Both axes (ours)	**82.1**	**85.3**	**87.4**	**89.7**

Temporal decay	Fixed β			Learned β_0		
Budget	Reverse 1.82	Uniform 1	Forward 0.55	0.10	0.90	0.55 (ours)
$H3\,L6$	2.0	38.4	74.0	66.3	63.3	**82.1**
$H32\,L6$	2.0	40.2	77.9	72.2	66.0	**89.7**

4.6 Connection placement and decay initialization

We evaluate the design choices underlying InfiLoop: the placement of the learned connection across recurrent loops, and the initialization of the temporal decay β_0. Table 2 summarizes both ablations on 7,680 puzzles sampled uniformly at random from the Sudoku-Extreme test set.

Applying the connection to both axes. The nested reasoner maintains two distinct recurrent trajectories: an inner loop for iterative candidate refinement, and an outer loop for context updates. Table 2 compares applying InfiLoop to the inner loop alone, the outer loop alone, and both loops simultaneously. The dual-axis configuration achieves the highest accuracy across all evaluated budgets. At the training depth of $H3\,L6$, dual-axis aggregation reaches 82.1% accuracy, compared with an inner connection alone at 72.6% and an outer connection alone at 75.1%. This margin widens as the reasoning budget expands to $H32\,L6$, reaching 89.7% against 79.4% and 80.0%. Both recurrent trajectories benefit from learned state reuse.

Forward decay over the loop trajectory. The decay $\beta = e^{-\lambda}$ sets how fast older block outputs lose weight in the aggregated state. Forward decay uses $\beta < 1$, so older outputs weigh less. Uniform weights use $\beta = 1$, which is the Full Attention Residuals case $\lambda = 0$. Reverse decay uses $\beta > 1$, so older outputs weigh more. We first fix β during training for the two-axis model. Reverse decay with $\beta = 1.82$ solves only 2.0% of the puzzles at $H3\,L6$. Uniform weights reach 38.4%, and forward decay with $\beta = 0.55$ reaches 74.0%. The gap grows with the budget: at $H32\,L6$, forward decay reaches 77.9%, uniform weights reach 40.2%, and reverse decay stays at 2.0%. Learning β from $\beta_0 = 0.55$ gives the best accuracy, 82.1% at $H3\,L6$ and 89.7% at $H32\,L6$. The initialization also matters: $\beta_0 = 0.10$ reaches 66.3% and $\beta_0 = 0.90$ reaches 63.3% at $H3\,L6$. We therefore learn a forward decay from $\beta_0 = 0.55$ on all benchmarks.

5 Conclusion

We discuss related work in Appendix B. In this paper, we identify state replacement under the carry-last rule as a primary bottleneck in deep looped Transformers: repeatedly replacing the recurrent state with the latest block output overwrites correct intermediate deductions and destabilizes solved configurations. To address this, we introduce InfiLoop, a loop-native residual connection that combines content-based pseudo-query scoring with exponential temporal decay across iterations. Its exact recurrence needs only $O(d)$ persistent memory per token and keeps the aggregated state within the convex hull of the candidate representations. On Sudoku-Extreme, Maze-Hard, ARC-AGI-1, and ARC-AGI-2, InfiLoop outperforms carry-last and fixed-point damping baselines, and it reaches their accuracy with fewer layers under test-time depth scaling. Furthermore, analysis of the learned connection reveals that it operates as an adaptive residual update: the model suppresses proposals with constraint violations, increases update magnitude upon reaching a valid solution, and resists state drift under perturbations. These findings show that learning how to selectively aggregate recurrent states is an effective mechanism for scaling test-time reasoning depth.

AI USE STATEMENT

Generative AI tools were used only to polish the writing. They were not used for research ideation, experimental design, analysis, code, or figure generation.

ETHICS STATEMENT

This work studies recurrent reasoning architectures on logic puzzles and abstract visual reasoning tasks. All models are trained from scratch on the public Sudoku-Extreme, Maze-Hard, ARC-AGI-1, and ARC-AGI-2 datasets and evaluated on their public test or evaluation splits. These datasets contain no personal information, and the research does not involve human subjects or crowdsourcing. We do not release new datasets.

REPRODUCIBILITY STATEMENT

Source code for training and evaluation is included as supplementary material. Section 4.1 describes the datasets, and Section 4.2 gives the architecture, baselines, and training configuration. Section 4.4 and Appendix A define how layer evaluations are counted, and Appendix A also gives the FLOP and memory accounting. Appendix D describes the content-score ablations. For baselines, we retrain TRM on Sudoku-Extreme with the authors' published recipe, and we evaluate the released FPRM checkpoint and the ARC Prize verification checkpoint of TRM under the protocols above.

REFERENCES

Anthropic. Claude 3.7 Sonnet system card. Official system card, 2025. URL https://www-cdn.anthropic.com/9ff93dfa8f445c932415d335c88852ef47f1201e/claude-3-7-sonnet-system-card.pdf.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. ReZero is all you need: fast convergence at large depth. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161, pp. 1352–1361. PMLR, 2021. URL https://proceedings.mlr.press/v161/bachlechner21a.html.

Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron C. Courville, and Se-Young Yun. Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/8b08bbf8b420faa6eeb4020720582ec7-Abstract-Conference.html.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/01386bd6d8e091c2ab4c7c7de644d37b-Abstract.html.

Andrea Banino, Jan Balaguer, and Charles Blundell. PonderNet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, 2021. URL https://openreview.net/forum?id=1EuxRTe0WN.

Arpit Bansal, Avi Schwarzschild, Eitan Borgnia, Zeyad Emam, Furong Huang, Micah Goldblum, and Tom Goldstein. End-to-end algorithm synthesis with recurrent networks: Extrapolation without overthinking. In *Advances in Neural Information Processing Systems*, volume 35, pp. 20232–20242. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/7f70331dbe58ad59d83941dfa7d975aa-Abstract-Conference.html.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014. URL https://aclanthology.org/D14-1179/.

François Chollet. On the measure of intelligence. arXiv preprint arXiv:1911.01547, 2019. URL https://arxiv.org/abs/1911.01547.

François Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. ARC prize 2024: Technical report. arXiv preprint arXiv:2412.04604, 2024. URL https://arxiv.org/abs/2412.04604.

François Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. ARC-AGI-2: A new challenge for frontier AI reasoning systems. arXiv preprint arXiv:2505.11831, 2025. URL https://arxiv.org/abs/2505.11831.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261, 2025. URL https://arxiv.org/abs/2507.06261.

DeepSeek-AI. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. arXiv preprint arXiv:2501.12948, 2025. URL https://arxiv.org/abs/2501.12948.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/25cc3adf8c85f7c70989cb8a97a691a7-Abstract-Conference.html.

Jacob Fein-Ashley and Paria Rashidinejad. Solve the loop: Attractor models for language and reasoning. In *First FoGen Workshop at ICML 2026*, 2026. URL https://openreview.net/forum?id=AlZtpLVXKf.

Zitian Gao, Lynx Chen, Yihao Xiao, He Xing, Ran Tao, Haoming Luo, Joey Zhou, and Bryan Dai. Universal reasoning model. arXiv preprint arXiv:2512.14693, 2025. URL https://arxiv.org/abs/2512.14693.

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, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Abstract-Conference.html.

Angeliki Giannou, Shashank Rajput, Jy-Yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. In *International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/61915.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4700–4708, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html.

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

Kimi Team, Guangyu Chen, Yu Zhang, Jianlin Su, Weixin Xu, Siyuan Pan, Yaoyu Wang, Yucheng Wang, Guanduo Chen, Bohong Yin, Yutian Chen, Junjie Yan, Ming Wei, Y. Zhang, Fanqing Meng, Chao Hong, Xiaotong Xie, Shaowei Liu, Enzhe Lu, Yunpeng Tai, Yanru Chen, Xin Men, Haiqing Guo, Y. Charles, Haoyu Lu, Lin Sui, Jinguo Zhu, Zaida Zhou, Weiran He, Weixiao Huang, Xinran Xu, Yuzhi Wang, Guokun Lai, Yulun Du, Yuxin Wu, Zhilin Yang, and Xinyu Zhou. Attention residuals. arXiv preprint arXiv:2603.15031, 2026. URL https://arxiv.org/abs/2603.15031.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Sajad Movahedi, Vera Milovanović, Shlomo Libo Feigin, Alexander Theus, Thomas Hofmann, Valentina Boeva, T. Konstantin Rusch, and Antonio Orvieto. Fixed-point reasoners: Stable and adaptive deep looped transformers. arXiv preprint arXiv:2606.18206, 2026. URL https://arxiv.org/abs/2606.18206.

OpenAI. OpenAI o3-mini system card. Official system card, 2025. URL https://openai.com/index/o3-mini-system-card/.

Matteo Pagliardini, Amirkeivan Mohtashami, Francois Fleuret, and Martin Jaggi. DenseFormer: Enhancing information flow in transformers via depth weighted averaging. In *Advances in Neural Information Processing Systems*, volume 37, pp. 136479–136508. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/f67449c7ab72f441d3a713b046c6818c-Abstract-Conference.html.

Ofir Press, Noah A. Smith, and Mike Lewis. Train short, test long: Attention with linear biases enables input length extrapolation. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=R8sQPpGCv0.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/2676109d49d1eb26d6bc584a8f556305-Abstract-Conference.html.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 6695–6706. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/3501672ebc68a5524629080e3ef60aef-Abstract.html.

Noam Shazeer. GLU variants improve transformer. arXiv preprint arXiv:2002.05202, 2020. URL https://arxiv.org/abs/2002.05202.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/hash/215a71a12769b056c3c32e7299f1c5ed-Abstract.html.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

Wenfang Sun, Xinyuan Song, Pengxiang Li, Lu Yin, Yefeng Zheng, and Shiwei Liu. The curse of depth in large language models. In *Advances in Neural Information Processing Systems*, volume 38, pp. 163104–163136. Curran Associates, Inc., 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/eeb57fdf745eb31a3c7ef22c59a4661d-Abstract-Conference.html.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. arXiv preprint arXiv:2307.08621, 2023. URL https://arxiv.org/abs/2307.08621.

Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 32–42, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Touvron_Going_Deeper_With_Image_Transformers_ICCV_2021_paper.html.

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

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. DeepNet: Scaling transformers to 1,000 layers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(10):6761–6774, 2024. URL https://ieeexplore.ieee.org/document/10496231.

Shaowen Wang, Bingrui Li, Ge Zhang, Wenhao Huang, and Jian Li. On the residual scaling of looped transformers: Stability and transferability. In *Latent & Implicit Thinking Workshop at ICLR 2026*, 2026. URL https://openreview.net/forum?id=bj8l2FYjSd.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, 2020. URL https://proceedings.mlr.press/v119/xiong20b.html.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/b8402301e7f06bdc97a31bfaa653dc32-Abstract-Conference.html.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Defa Zhu, Hongzhi Huang, Zihao Huang, Yutao Zeng, Yunyao Mao, Banggu Wu, Qiyang Min, and Xun Zhou. Hyper-connections. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/f1e8ff97057e0c3f6bcb7018c78d4df1-Abstract-Conference.html.

A Compute and Persistent-State Accounting

We quantify the computation and persistent state of InfiLoop. It stores 582.8 KiB of persistent state per puzzle, independent of the loop budget.

Layer evaluations. Each outer cycle applies the shared block L times to the inner state and once to the outer state. With S supervision segments, H outer cycles per segment, and K layers per block, the number of Transformer layer evaluations is

$$C(H, L) = SH(L + 1)K. \tag{6}$$

Training uses S=12, H=3, L=6, and K=2, which gives 504 layer evaluations. The depth study in Section 4.4 fixes H=6 and L=12, so each segment executes 156 layers, and varies S from 1 to 160, which gives 156 to 24,960 layer evaluations. This count omits the connection and the prediction heads, which we add below. The embedding is a table lookup and adds no multiply-accumulates. FPRM also uses a two-layer block with width 512 and the same sequence length. Layer evaluations therefore match the dominant Transformer arithmetic across the two models. FPRM's solver operations are outside the InfiLoop FLOP accounting below. These counts do not compare measured runtime.

Profiling setup and FLOP convention. We profile the InfiLoop Sudoku checkpoint at batch size one under the 12-segment inference protocol. Each sequence has 81 board cells and 16 prefix positions, so the shared block processes n=97 positions. The SwiGLU intermediate width is m=1,536: the implementation takes two thirds of the expansion $4d$ and rounds it up to a multiple of 256. We count two FLOPs per multiply-accumulate in linear projections, attention matrix products, the depthwise convolution, and the content scores of the connection. Normalization, activation functions, rotary embeddings (Su et al., 2024), other elementwise arithmetic, and data movement are not counted.

Counted FLOPs. Let d be the hidden width, g the number of board cells, and k_c the convolution kernel width. One application of the shared block costs

$$F_{\text{block}} = K(8nd^2 + 4n^2d + 6ndm) + 2gdk_c^2, \tag{7}$$

where the convolution applies once per block to the g board cells and the attention and feed-forward projections apply to all n positions. For n=97, d=512, m=1536, K=2, g=81, k_c=3, this is 1.36154 GFLOPs per block application in InfiLoop. The prediction head has V=11 outputs per position. The halting head of the adaptive computation procedure has two outputs and reads one position. With T=$SH(L + 1)$ block applications, the counted total is

$$F_{\text{InfiLoop}} = TF_{\text{block}} + S(2ndV + 4d) + 2nd(T + I), \qquad I = SH + 2S. \tag{8}$$

The term $2ndT$ is the content score of every block output. The term $2ndI$ counts the content scores of initialization writes. These comprise one write per outer cycle and two writes per segment in the carry reset. The carry reset replaces the recurrent state of a puzzle that halted in the previous segment with the learned initial state. The carry reset is scored for every puzzle, because the implementation builds both the reset and the retained state before selecting one. We count the operators of one segment in the InfiLoop checkpoint and multiply by S=12. This count matches the formula at four profiled depths: 0.34, 0.92, 2.55, and 25.6 TFLOPs per puzzle at H=3, L=6, H=8, L=6, H=12, L=12, and H=32, L=48. At the depth-study setting H=6, L=12, the formula gives 1.27 TFLOPs at 1,872 layers and 17.0 TFLOPs at 24,960 layers.

The content scores are the only operations of the connection in the totals above. The remaining arithmetic of the connection is elementwise. We charge each arithmetic primitive or unary function once and exclude comparisons, casts, indexing, and memory traffic. Under this convention, the remaining arithmetic adds $T[n(7d + 6) + 1] + In(4d + 4)$ scalar operations.

Persistent state. The shared block attends bidirectionally over the whole input and recomputes keys and values at every application. InfiLoop keeps no KV cache between iterations. What persists between block applications is the recurrent carry. At batch size one, the two BF16 carries occupy $2nd \cdot 2$ bytes, or 194.0 KiB. A carry-last TRM keeps the same two carries. InfiLoop adds an FP32 numerator and denominator per position on each axis, $2n(d + 1) \cdot 4$ bytes or 388.8 KiB, for a total of 582.8 KiB. None of these sizes depends on H, L, or S. They scale linearly with batch size and sequence length. Model weights, input tensors, and temporary operator workspaces are not part of this accounting.

B Related Work

Reasoning through recurrent depth. Universal Transformers apply one shared block repeatedly (Dehghani et al., 2019). Later work studied the computational capabilities of looped Transformers and their use for latent reasoning (Giannou et al., 2023; Saunshi et al., 2025; Geiping et al., 2025). Looped Transformers have also been studied as learners of iterative algorithms (Yang et al., 2024) and for generalization to longer input sequences (Fan et al., 2025). HRM and TRM build compact reasoners from nested latent updates (Wang et al., 2025; Jolicoeur-Martineau, 2025). URM studies architecture and optimization choices for recursive reasoning (Gao et al., 2025). Adaptive Computation Time, PonderNet, and Mixture-of-Recursions learn how much computation to spend (Graves, 2016; Banino et al., 2021; Bae et al., 2025). InfiLoop modifies the connection between iterations of a nested TRM-style reasoner.

Overthinking in recurrent networks. Schwarzschild et al. (2021) showed that recurrent networks trained on easier problems can generalize to harder ones by taking more iterations at test time. Bansal et al. (2022) identified degradation under excessive iteration and addressed it with input recall and progressive training. Their recall connection preserves access to the original problem instance, while InfiLoop aggregates candidate states produced during reasoning. Both address behavior beyond the training iteration budget, but act on different information in the recurrence.

Stabilizing deep and looped Transformers. A looped Transformer at many iterations is a very deep weight-shared network. Section 2 measures one consequence: carry-last TRM accuracy peaks and then falls with depth. Normalization placement affects gradient stability at initialization (Xiong et al., 2020), while depth-dependent normalization scaling controls activation growth (Sun et al., 2025). ReZero uses a zero-initialized residual gate (Bachlechner et al., 2021), LayerScale learns channel-wise residual scaling (Touvron et al., 2021), and DeepNet combines residual scaling with a tailored initialization (Wang et al., 2024). Residual scaling for looped Transformers controls residual growth under weight sharing (Wang et al., 2026). Deep Equilibrium Models formulate weight-tied networks through fixed-point root finding and implicit differentiation (Bai et al., 2019). FPRM combines pre-norm, learned residual scaling, and damped fixed-point updates, with convergence-based halting (Movahedi et al., 2026). Attractor Models and equilibrium reasoners use fixed-point formulations for latent reasoning (Fein-Ashley & Rashidinejad, 2026; Huang et al., 2026). At a fixed point the update size is zero. These approaches stabilize recurrent computation through normalization, update scaling, or fixed-point formulations. InfiLoop instead learns from the task loss which earlier outputs the next state keeps and by how much.

Learned residual connections. ResNet propagates features through identity skip connections (He et al., 2016), while DenseNet concatenates features from preceding layers (Huang et al., 2017). DenseFormer learns weights for averaging current and earlier layer outputs (Pagliardini et al., 2024). Hyper-Connections learn how to mix features across depth and parallel residual streams (Zhu et al., 2025). These works establish learned cross-layer information flow as a broader design space. InfiLoop specializes this idea to a shared recurrent block: content-dependent weights and exponential age decay admit a running numerator and denominator whose size is independent of the number of iterations.

Gated recurrent updates. The interpolation in Equation 5 has the same form as an update gate in a GRU (Cho et al., 2014) or the coupled transform and carry gates in a Highway Network (Srivastava et al., 2015). The distinction is the gate parameterization: γ_t is the ratio of the new content weight to the sum of that weight and the decayed historical mass, rather than the output of a separate gating network. One shared content scorer and one decay parameter add $d + 1$ parameters per recurrence axis. This parameterization gives the exact normalized history aggregation in Equation 2.

Relation to Attention Residuals. Full Attention Residuals learns input-dependent weights over the outputs of earlier layers (Kimi Team et al., 2026). Each layer has its own pseudo-query and attends to the stored outputs of earlier layers, so memory grows with depth. A blockwise variant stores block-level summaries to reduce this history. A looped model has no fixed number of iterations, and a loop-native connection therefore cannot store the full activation history. InfiLoop scores each output once, at its own iteration, with one scorer shared by every iteration. A learned exponential decay then ages this weight. The connection keeps only a running weighted sum and a running normalization mass, so its memory does not grow with the loop count and the same rule applies at any depth. Retention networks also combine exponential age weighting with a recurrent form (Sun

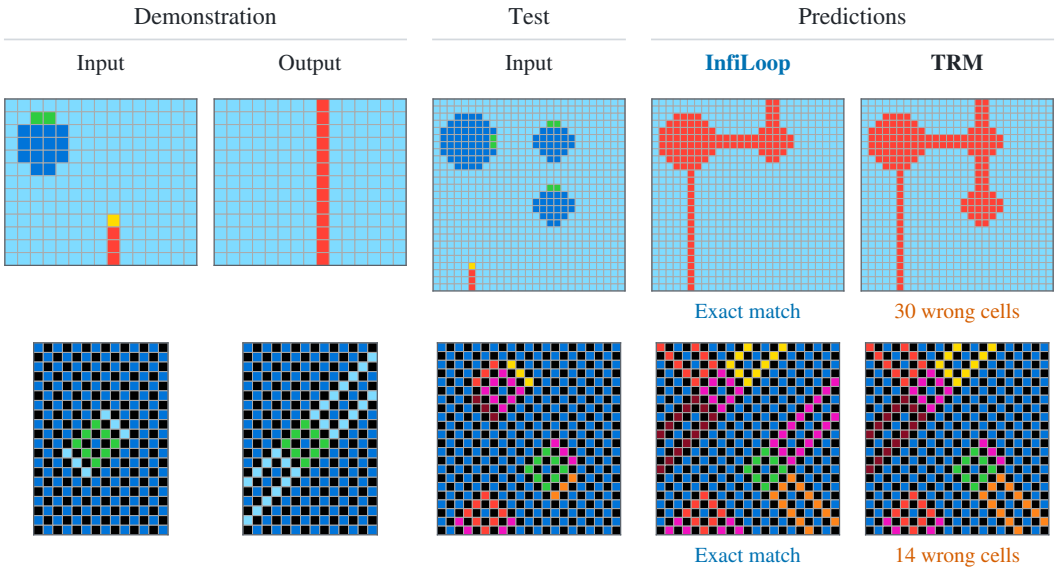

Figure 6: **Qualitative comparison on ARC-AGI-2.** Each row shows a demonstration input–output pair, a test input, and the predictions of InfiLoop and TRM. The top and bottom rows correspond to tasks `5961cc34` and `80a900e0`, respectively. As in the pass@2 evaluation, we show the best of each model's two submitted attempts. InfiLoop matches the target exactly in both cases; TRM makes 30 and 14 cell errors, respectively.

et al., 2023), and ALiBi demonstrates input-length extrapolation with linear relative-position biases (Press et al., 2022). In InfiLoop, the recurrence runs over reasoning iterations and yields the next recurrent state.

C Qualitative analysis on ARC-AGI-2.

Figure 6 visualizes the qualitative predictions of InfiLoop and TRM on two held-out ARC-AGI-2 evaluation tasks. In Task `5961cc34`, which demands geometric expansion of spatial components, InfiLoop produces the exact target structure, while TRM produces 30 erroneous cells with distorted boundaries. In Task `80a900e0`, the model must extend colored diagonal rays across an alternating checkerboard background. InfiLoop preserves the underlying background regularity while drawing sharp diagonal rays. TRM introduces 14 corrupted cells that break the checkerboard pattern and fragment the rays. These qualitative results demonstrate that learned state reuse provides stability across complex spatial rules where carry-last propagation degrades structural coherence.

D Removing the Content Score

Section 4.5 shows that the content score tracks board correctness. We test whether this score also matters for accuracy by removing it in two ways. Setting $q = 0$ gives every candidate the same score $s_i = 0$, and Equation 5 then reduces the connection to an exponential moving average whose step size approaches $1 - \beta$. All results in this section use the Sudoku-Extreme test puzzles of Section 4.4, evaluated at $H6\,L12$ with S=12, or 1,872 layers. Table 3 summarizes them.

Table 3: Exact accuracy (%) on Sudoku-Extreme at $H6\,L12$, S=12, or 1,872 layers.

InfiLoop	$q = 0$ in training	$q = 0$ at test time	$\beta = 0$ at test time
90.9	84.1	84.8	1.5

Removing the score during training. We train a second model with the same recipe, seed, and number of training steps, and with q fixed to zero on both axes. It reaches 84.1% against 90.9% for InfiLoop, so the content score is worth 6.8 percentage points when the model is trained with it.

Removing the score at test time. We also set q to zero in the trained InfiLoop checkpoint without retraining, and accuracy falls from 90.9% to 84.8%. Setting β to zero instead turns the connection

into carry-last, and accuracy falls to 1.5%, because the block weights are trained to operate on the aggregated state rather than on raw block outputs.

Both ways of removing the score lower accuracy by 6 to 7 percentage points. The gain of InfiLoop therefore does not come from damping alone: a fixed exponential average also damps the updates but lacks the content score, and it is less accurate.