RECURSIVE REASONING WITH FIXED POINT ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recurrent reasoning increases computational depth without adding parameters, but standard looping repeats the entire Transformer block. Because feedforward layers consume three to four times more compute than self-attention, coupling every attention step to a feedforward update wastes FLOPs. We ask whether a model can reason better by re-examining evidence multiple times before revising its answer. We introduce Fixed Point attention with Reasoning FPSA-R, an answer-conditioned fixed-point reasoner that makes self-attention refinement the unit of adaptive computation. At each stage, the model holds input-derived values and an answer-conditioned anchor fixed while repeatedly updating queries and keys from an evolving latent state. This process refines attention alignment until reaching an approximate equilibrium; only then does a feedforward transformation update the answer representation. The revised answer conditions the next equilibrium, creating a sequence of distinct reasoning problems rather than repeatedly solving a static input-only system. FPSA-R thus separates two forms of depth: adaptive attention refinement within a stage and successive answer revision across stages. Training these conditional equilibria with implicit differentiation and segment-level supervision yields $O(1)$ training memory and reduces computation by 60% compared to standard loop Transformers. We also derive conditional bounds connecting numerical residuals to representation and gradient errors without equating convergence with correctness. On empirical benchmarks, a 7M-parameter FPSA-R achieves 98.5% on Sudoku-Extreme, 95.2% on Maze-Hard, and 44.2% and 15.3% on ARC-AGI-1 and ARC-2. These results demonstrate that multiple rounds of lightweight attention between feedforward transformations provide a far more efficient allocation of reasoning compute than full-block recurrence.

1 INTRODUCTION

Reasoning benefits from computation that grows with the difficulty of the problem. Large language models get this computation by generating chain-of-thought tokens, trading latency and sequence length for accuracy (Wei et al., 2022). Recurrent models spend it in latent space instead: they update an internal state many times before producing an answer (Dehghani et al., 2019; Geiping et al., 2025a). Models of this kind with only a few million parameters now solve Sudoku, maze, and ARC-AGI puzzles that require long chains of dependent computation (Wang et al., 2025; Jolicoeur-Martineau, 2025; Movahedi et al., 2026; Huang et al., 2026), and the strongest of them run their recurrence to a fixed point, stopping once the latent state no longer changes (Movahedi et al., 2026; Huang et al., 2026). Adaptive computation methods go further and let the network decide how much of this computation to spend on each input (Graves, 2016; Banino et al., 2021; Mathur & Plis, 2023; Mathur et al., 2025b).

These models differ in how they train and when they stop, but they agree on what to repeat. Each recurrent step applies a full Transformer block, so every extra step repeats attention, the feedforward network, normalization, and the residual update together. The two main operations do different jobs. Attention moves information between positions, retrieving and relating evidence, while the feedforward network transforms the features at each position. A problem that needs several dependent rounds of evidence gathering therefore pays for a feedforward transformation at every round, even when all it needs is another round of communication. Global attention does not remove this need. One attention layer can see every position, but the evidence it retrieves can change the query the

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.

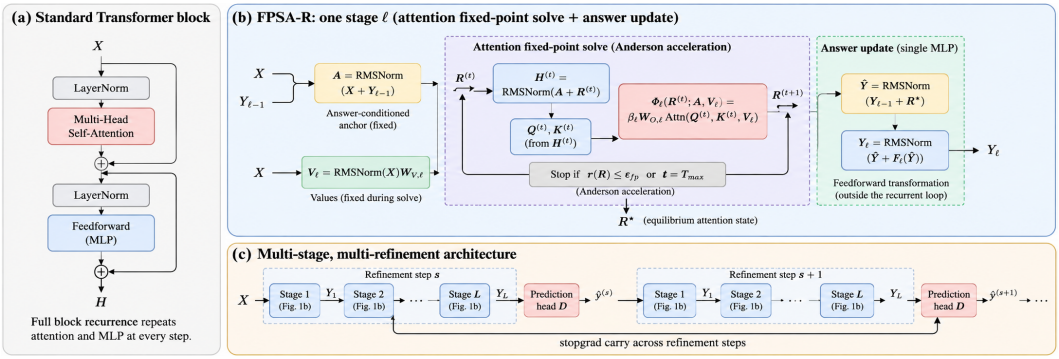

Figure 1: FPSA-R separates two timescales. Within a stage, queries and keys are recomputed from an evolving attention state while the values V and the answer-conditioned anchor A stay fixed, until the attention state reaches equilibrium $R^\star$. A single feedforward update then revises the answer Y, which sets the anchor for the next stage.

model should ask next, and a standard Transformer can only act on that change by moving to another layer (Vaswani et al., 2017). This raises a basic design question for recurrent reasoners: *what should the recurrence repeat?*

We argue that **attention should run on a faster recurrent timescale than feedforward transformation**. In Sudoku, each new assignment changes which constraints carry information. In a maze, progress changes which frontier to explore, and in ARC, a candidate transformation changes which relations among the demonstrations remain unexplained. In each case what the model learns at one step changes what it should attend to at the next, and it may take several such rounds before a feature transformation is worth its cost.

We build this idea into FPSA-R, a recurrent reasoner based on answer-conditioned fixed-point self-attention (Figure 1). Within each stage, FPSA-R holds the values and an anchor built from the input and the current answer fixed, and iterates queries and keys until the attention state reaches equilibrium. Only then does a single feedforward update revise the answer. The revised answer changes the anchor of the next stage, so each stage solves a new routing problem conditioned on the model's current hypothesis rather than repeating the previous one. Where FPRM and EqR drive the whole block to a fixed point, FPSA-R solves for an equilibrium of information routing alone and keeps the feedforward network outside the loop. We train this equilibrium with implicit differentiation (Bai et al., 2019b), so training memory stays nearly constant as the number of attention iterations grows.

Our experiments test whether this placement of recurrence matters. With 7.0M parameters, FPSA-R reaches 98.5, 95.2, 50.6, and 15.3% on Sudoku-Extreme, Maze-Hard, ARC-AGI-1, and ARC-AGI-2 (Chollet, 2019; Chollet et al., 2025). Against a block-loop control that rebuilds FPRM's recurrence in the same framework, it improves by up to 9.6 points at matched parameters or FLOPs, and the gap grows with reasoning depth, reaching 11.6 points on unseen 30×30 mazes. We also find that convergence is a weak signal of correctness: the residual predicts it with an AUROC of only 0.61, and stopping on read-out stability instead matches accuracy with 42% fewer iterations.

The rest of the paper develops and tests this hypothesis. Section 2 introduces FPSA-R, where each reasoning stage solves an attention equilibrium with a fixed anchor and value representation, followed by a single feedforward update. We train the equilibrium with implicit differentiation, avoiding storage of the full recurrent trajectory.

Section 3 evaluates whether the placement of recurrence matters. We compare FPSA-R with published recurrent reasoners and with parameter- and compute-matched block-loop controls (Section 3.2), isolate the components of the recurrent map (Section 3.1), and study test-time depth and out-of-distribution generalization through recurrent budget, maze size, and Sudoku clue count (Section 3.3). We then analyze how predictions change during recurrence and compare numerical and semantic stopping criteria (Section 3.5). Finally, Section 3.6 shows that implicit differentiation matches full BPTT within run-to-run variation while reducing activation memory by $8.6\times$.

2 METHOD: FPSA-R

FPSA-R separates two operations that a Transformer block couples: moving information between positions and transforming features at each position. Section 2.1 defines the answer-conditioned fixed-point attention that does the first and the feedforward update that does the second. Appendix B explains how we solve and differentiate the equilibrium, Appendix C states what a small residual does and does not guarantee, and Section 2.3 gives the cost of each iteration. Algorithm C summarizes one refinement step.

2.1 ANSWER-CONDITIONED FIXED-POINT ATTENTION

An encoder maps the input x to a contextual representation

$$X = E_\psi(x) \in \mathbb{R}^{N\times d}. \tag{1}$$

The model keeps an answer state in $\mathbb{R}^{N\times d}$ and refines it over S answer-refinement steps, each made of L stages. We write $Y_{s,\ell}$ for the answer after stage ℓ of step s, and $Y_{s,0}$ for the answer carried in from the previous step. Each stage holds the answer fixed while it solves an attention subproblem, and only then applies a feedforward update to revise the answer.

Stage ℓ of step s first forms an answer-conditioned anchor

$$A_{s,\ell} = \mathrm{RMSNorm}\left(X + Y_{s,\ell-1}\right), \tag{2}$$

where RMSNorm is root-mean-square normalization (Qiao et al., 2023), and computes evidence values once from the input,

$$V_\ell = \mathrm{RMSNorm}(X)\, W_{V,\ell}. \tag{3}$$

Both stay fixed for the whole attention solve. Fixing the values keeps the evidence stable, so the iterations change where information flows but not what it says.

Let $R^{(t)}$ denote the recurrent attention state at inner iteration t, where we drop the indices s and ℓ when the stage is clear. The model builds each attention pattern from

$$H^{(t)} = \mathrm{RMSNorm}\left(A_{s,\ell} + R^{(t)}\right), \tag{4}$$

and recomputes queries and keys from this evolving representation with rotary position embeddings (Su et al., 2021),

$$Q_h^{(t)} = \mathrm{RoPE}\left(H^{(t)} W_{Q,\ell}^h\right), \qquad K_h^{(t)} = \mathrm{RoPE}\left(H^{(t)} W_{K,\ell}^h\right). \tag{5}$$

We normalize queries and keys within each head (Hendrycks et al., 2021) and compute

$$P_h^{(t)} = \mathrm{softmax}\left(\frac{\widehat{Q}_h^{(t)} \widehat{K}_h^{(t)\top}}{\tau_{\ell,h}}\right), \tag{6}$$

where $\tau_{\ell,h} > 0$ is a learned head-specific temperature. Because the normalized queries and keys have unit norm, every attention logit lies in $[-1/\tau_{\ell,h},\, 1/\tau_{\ell,h}]$, which Section 2.2 uses. The recurrent map is

$$\Phi_\ell(R; A, V) = \beta_\ell\, W_{O,\ell}\, \mathrm{Concat}_h\left[P_h(R)\, V_\ell^h\right], \tag{7}$$

where $\beta_\ell \in (0, 1)$ is a learned gain that scales the attention output. FPSA-R seeks the fixed point

$$\boxed{R_{s,\ell}^\star = \Phi_\ell\left(R_{s,\ell}^\star; A_{s,\ell}, V_\ell\right).} \tag{8}$$

Equation 8 captures the central asymmetry of FPSA-R. The values and the anchor stay fixed during a solve, while the recurrent state, the queries and keys, and so the attention pattern change. Evidence retrieved at one iteration can therefore change the alignment computed at the next without changing the evidence itself.

Once the attention state converges, the model folds it into the current answer and applies a single feedforward transformation,

$$\widetilde{Y}_{s,\ell} = \mathrm{RMSNorm}\left(Y_{s,\ell-1} + R_{s,\ell}^\star\right), \tag{9}$$

$$Y_{s,\ell} = \mathrm{RMSNorm}\left(\widetilde{Y}_{s,\ell} + F_\ell\left(\widetilde{Y}_{s,\ell}\right)\right). \tag{10}$$

The MLP F_ℓ runs once per stage, after the attention equilibrium, rather than once per attention update. A stage can therefore perform many rounds of information routing and a single nonlinear answer transformation.

The updated answer changes the anchor of the next stage through Eq. 2, so successive stages solve different fixed-point problems. Across refinement steps, the final answer of step s initializes the next,

$$Y_{s+1,0} = \mathrm{stopgrad}\,(Y_{s,L})\,, \tag{11}$$

and the recurrent states carry over as warm starts for the next solves, as in the detached carry of recent recursive reasoners (Wang et al., 2025; Jolicoeur-Martineau, 2025). FPSA-R thus alternates between solving an attention problem under the current answer hypothesis and revising that hypothesis with a feedforward transformation.

Relation to other fixed points and iterative attention. FPRM and EqR iterate a full Transformer block to a fixed point (Movahedi et al., 2026; Huang et al., 2026), so the converging state includes the feedforward output and every iteration pays for the MLP. Iterative attention has a longer history: a single attention step performs one update of a modern Hopfield network retrieving against fixed stored patterns (Schöne et al., 2025), and Slot Attention and the Perceiver iterate attention to refine a set of latents (Locatello et al., 2020; Jaegle et al., 2021). FPSA-R shares their fixed evidence but differs in two ways. Its equilibrium is conditioned on the current answer through the anchor, so each stage solves a different problem, and it places this equilibrium inside a two-timescale reasoner where the feedforward network runs only between solves.

2.2 What the Residual Guarantees

The solver reports a residual, and training uses the gradient at the returned state. We now state what a small residual implies about both. All norms are Frobenius norms on $\mathbb{R}^{N\times d}$, and proofs appear in Appendix C.

Assumption 1 (Local contraction)**.** *There is a set $\mathcal{U}$ that contains the solver iterates and a fixed point $R^\star$ on which $\Phi_\ell(\cdot\,; A, V)$ is γ-Lipschitz for some $\gamma < 1$.*

The next lemma shows why this assumption is reasonable for FPSA-R but not for standard attention.

Lemma 1 (Normalized attention is Lipschitz)**.** *Suppose that on $\mathcal{U}$ the inputs to every normalization in Eqs. 2–6 have row norms bounded below by $\nu > 0$. Then Φ_ℓ is Lipschitz on $\mathcal{U}$, with a constant that grows linearly in $\beta_\ell \|W_{O,\ell}\|_2 \|V_\ell\|_2$ and in $1/\min_h \tau_{\ell,h}$.*

Unnormalized dot-product attention has no finite global Lipschitz constant (Kim et al., 2021). Query-key normalization and the temperature bound the logits, and β_ℓ scales the whole map, so contraction is a property the model can learn and the stability control can enforce.

Proposition 1 (Residual bounds the state error)**.** *Under Assumption 1, $R^\star$ is the unique fixed point in $\mathcal{U}$, and for every $R \in \mathcal{U}$*

$$\|R - R^\star\| \le \frac{\|\Phi_\ell(R) - R\|}{1-\gamma} \le \frac{r(R)\,c(R)}{1-\gamma}, \tag{12}$$

where $c(R) = \big(\sum_i (\|\Phi_\ell(R)_i\|_2 + \|R_i\|_2 + \epsilon)^s 2\big)^{1/2}$ converts the normalized residual of Eq. 7 into an absolute one.

Proposition 2 (Residual bounds the gradient error)**.** *Under Assumption 1, suppose further that Φ_ℓ, $\mathcal{L}$, and their first derivatives are Lipschitz on $\mathcal{U}$. Let $\hat{R} \in \mathcal{U}$ with $\delta = \|\hat{R} - R^\star\|$, and let $\hat{u}$ solve Eq. 10 at $\hat{R}$ with relative GMRES residual at most η. Then the gradient $\hat{g}$ computed by Eq. 10 at $(\hat{R}, \hat{u})$ satisfies*

$$\|\hat{g} - \nabla_\theta \mathcal{L}\| \le C\left(\frac{\delta}{(1-\gamma)^2} + \frac{\eta}{1-\gamma}\right), \tag{13}$$

where C depends only on the Lipschitz and norm bounds.

Together, the two propositions say that a small residual certifies an accurate state and an accurate implicit gradient, and that both guarantees weaken as γ approaches 1. They lapse entirely without contraction, which is why the stability control matters.

Remark 1. Neither bound involves the task. A small residual shows that the solver has found the equilibrium of Eq. 8, not that this equilibrium decodes to the correct answer. Section 3.5 measures this gap directly.

2.3 Cost per Iteration

We count two FLOPs per multiply-accumulate (Geiping et al., 2025b). A block-loop iteration with a $4\times$ MLP costs $8Nd^2$ for the query, key, value, and output projections, $16Nd^2$ for the MLP, and $4N^2d$ for the attention scores and weighted sum, for $24Nd^2 + 4N^2d$ in total. An FPSA-R inner iteration computes only the query, key, and output projections and the attention, for $6Nd^2 + 4N^2d$, plus $O(mNd)$ for an Anderson history of size m, which is small when $m \ll d$. The values and the MLP cost $18Nd^2$ once per stage. The per-iteration ratio

$$\frac{6d + 4N}{24d + 4N} \tag{14}$$

approaches $1/4$ when $N \ll d$ and rises toward 1 as N grows. At $d = 512$, for example, it is 0.27 for Sudoku ($N = 81$) and 0.42 for 30×30 mazes ($N = 900$). Looping attention alone therefore saves the most on short inputs, where the MLP dominates the cost of a block.

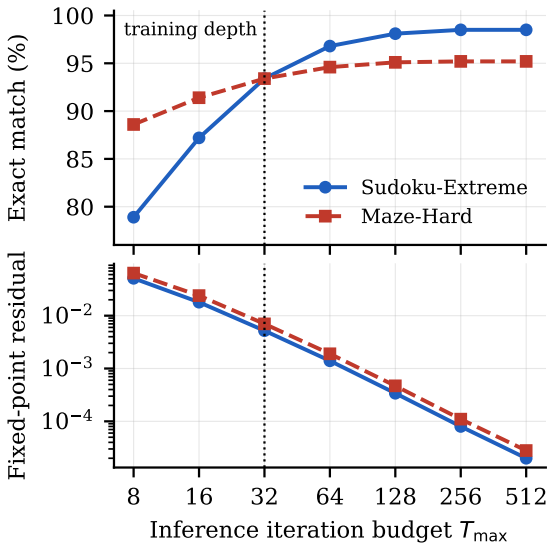

Figure 2: Accuracy against realized inference GFLOPs for models trained with $T = 32$, evaluated from one checkpoint at increasing iteration budgets on (a) Sudoku-Extreme and (b) Maze-Hard. Dashed lines mark the training budget ($T = 32$) and the largest evaluation budget ($T = 512$). The star marks $T = 1$. Shaded regions show one standard deviation over 5 seeds.

This formulation gives three separate kinds of computational depth: the number of attention iterations that solve a stage, the number of stages that transform the answer within a step, and the number of answer-refinement steps. FPSA-R can therefore spend more on attention without spending proportionally more on feedforward computation.

3 Experiments

Our experiments ask whether moving recurrence from the full Transformer block into attention produces a better reasoner, and whether the gain comes from where recurrence is placed rather than from extra parameters or compute. We compare FPSA-R with published recurrent reasoners and with parameter- and compute-matched controls (Section 3.2), test how the advantage changes as problems demand more reasoning depth (Section 3.3), check whether fixed-point attention transfers to pretrained language models (Section 3.4), and examine what the recurrence does and when it should stop (Section 3.5). Appendix E isolates the components of the recurrent map.

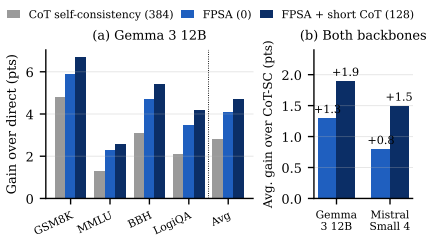

Figure 3: FPSA against chain-of-thought at a matched budget of $1.5\times$ direct-answer FLOPs. Parentheses give reasoning tokens decoded per example. (a) Gain over direct answer on Gemma 3 12B. (b) Average gain over CoT self-consistency on both backbones. Full results in Appendix F.

3.1 Setup

We evaluate on Sudoku-Extreme, Maze-Hard, ARC-AGI-1, and ARC-AGI-2 with the data splits, augmen-

Table 1: Main reasoning results. Published rows use the values and protocols of the original papers; EqR reports mazes on a different benchmark (Maze-Unique) and does not report full ARC-AGI. FPSA-R and the block-loop controls share one training and evaluation pipeline and differ only in the recurrent operator. Latency is wall-clock time per Sudoku-Extreme puzzle on one A100 80GB at batch size 256. Reproduced results report mean ± standard deviation over 5 seeds.

				Accuracy (%)			
Method	Params	Rel. FLOPs	Latency (ms)	Sudoku-Ext.	Maze-Hard	ARC-1	ARC-2
Published reference results							
HRM (Wang et al., 2025)	27M	–	–	55.0	74.5	40.3	5.0
TRM-Att (Jolicoeur-Martineau, 2025)	7M	–	–	74.7	85.3	44.6	7.8
FPRM (Movahedi et al., 2026)	7M	–	–	94.2	87.0	47.5	6.2
EqR (Huang et al., 2026)	5.0M	–	–	**98.7**	–	–	–
Controlled comparison							
Block loop, parameter-matched	7.0M	1.22×	3.9	93.8 ± 0.4	87.2 ± 0.5	47.1 ± 0.9	6.4 ± 0.6
Block loop, FLOP-matched	7.0M	1.00×	3.2	92.1 ± 0.5	85.9 ± 0.6	45.6 ± 1.0	5.7 ± 0.7
FPSA-R	7.0M	1.00×	3.3	98.5 ± 0.2	$\mathbf{95.2 \pm 0.4}$	$\mathbf{50.6 \pm 0.8}$	$\mathbf{15.3 \pm 0.7}$

tation, and evaluation protocols of Wang et al. (2025) and Jolicoeur-Martineau (2025). Sudoku and Maze use exact-solution accuracy, so a prediction counts only when the full output grid matches the target. For both ARC benchmarks we report task-level pass@2 with voting across augmented views, and Appendix A gives the number of views and every other detail of the protocol.

HRM, TRM-Att, FPRM, and EqR serve as published reference methods. Our controlled comparison is a block-loop model that rebuilds FPRM's fixed-point recurrence inside our framework: it applies the complete attention–MLP block at every recurrent step and shares FPSA-R's embeddings, data, augmentation, supervision schedule, optimizer, EMA, seeds, Anderson solver, stopping rule, and implicit gradient. The two models therefore differ only in what they loop. We train one block loop matched to FPSA-R in parameters and one matched in realized inference FLOPs.

Adaptive models run different numbers of recurrent steps on different inputs, so we report the FLOPs each model actually spends rather than its iteration cap. One counter measures every reproduced model and includes every evaluation of the recurrent map and every Anderson update. Latency on a single A100 80GB at batch size 256 captures what FLOPs miss, such as kernel launches and the control flow of adaptive stopping. Reproduced results report mean and standard deviation over 5 matched seeds.

The analyses in Sections 3.3, 3.5, and 3.6 and Appendix E use a lighter study configuration (Appendix A), so their absolute accuracies sit below Table 1. Each analysis compares variants trained and evaluated the same way.

3.2 Attention Recurrence Outperforms Block Recurrence

Table 1 compares FPSA-R with published recurrent reasoners and with the two block-loop controls. FPSA-R reaches the highest accuracy among the reference methods on Maze-Hard, ARC-AGI-1, and ARC-AGI-2, and comes within 0.2 points of EqR on Sudoku-Extreme, which EqR reaches by scaling test-time iterations much further. It improves on FPRM, the strongest prior model that reports all four benchmarks, by 4.3 points on Sudoku-Extreme, 8.2 on Maze-Hard, 3.1 on ARC-AGI-1, and 9.1 on ARC-AGI-2, and it nearly doubles the best prior ARC-AGI-2 score at this scale. These rows come from different training pipelines, so they show where FPSA-R stands but not why it wins.

The controlled comparison answers that second question. The parameter-matched block loop lands within 0.4 points of published FPRM on Sudoku-Extreme and Maze-Hard, so our pipeline reproduces the prior operating point. It still trails FPSA-R by 4.7 points on Sudoku-Extreme, 8.0 on Maze-Hard, 3.5 on ARC-AGI-1, and 8.9 on ARC-AGI-2 while spending 22% more compute, and FPSA-R solves each puzzle 15% faster in wall-clock time. Matching the block loop to FPSA-R's FLOP budget widens the gaps to 6.4, 9.3, 5.0, and 9.6 points. Both models solve for a fixed point with the same solver and gradient, so the gap comes from what they solve for: looping attention beats looping the whole block whether the two models are matched in capacity or in compute.

Table 2: Accuracy (%) as reasoning depth grows, against a TRM-style block loop trained the same way (study configuration). Left: maze models trained on 7×7 and 9×9 grids; † marks unseen sizes. Right: Sudoku-Extreme grouped by number of given clues.

Maze size	Block loop	FPSA-R
9×9	94.0	**95.1**
11×11†	82.5	**88.7**
15×15†	60.8	**73.6**
30×30†	44.5	**61.8**

Clues	Block loop	FPSA-R
≥ 30	98.1	**98.8**
26–29	95.0	**97.1**
22–25	91.2	**94.9**
≤ 21	83.5	**89.8**

The gap is widest on Maze-Hard and ARC-AGI-2, the two benchmarks that demand the longest chains of dependent steps. A path through a 30×30 maze can run for hundreds of cells, and ARC-AGI-2 was built around tasks that compose several rules where ARC-AGI-1 tasks often apply one. This pattern is consistent with our hypothesis that such problems gain more from moving information often than from transforming it often.

Appendix E changes one component at a time and shows where this gain comes from. Removing the fixed-point solve costs the most, up to 22.2 points on ARC-AGI-1. Freezing the keys and letting the values evolve both hurt, so the recurrence works by updating the attention alignment while the evidence stays fixed. Moving the MLP inside the loop makes the model worse even with 27% more compute, which shows that feedforward computation helps at the answer timescale but not at the attention timescale.

3.3 The Advantage Grows with Reasoning Depth

We next raise the depth of reasoning a problem demands in three ways: more test-time iterations, larger mazes, and fewer Sudoku clues.

A model trained with $T = 32$ keeps improving when run longer at test time (Figure 4; exact values in Appendix Table 5). Sudoku accuracy rises from 70.2% at $T = 8$ to 84.6% at the training depth and 89.7% at $T = 256$, and Maze-Hard rises from 82.0% to 86.8% and then 88.6%. Accuracy saturates while the residual keeps falling: from $T = 128$ to $T = 512$ the Sudoku residual drops by more than an order of magnitude while accuracy changes by 0.4 points. Section 3.5 returns to this gap between a settled answer and a settled state.

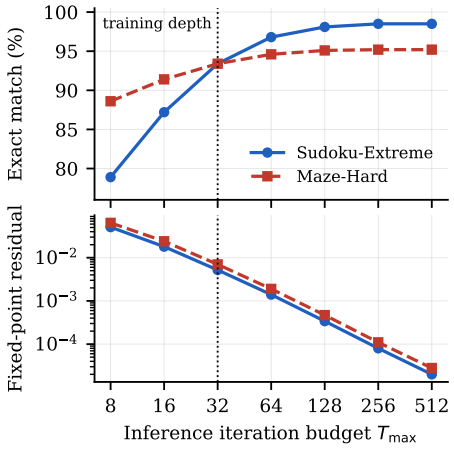

Figure 4: Test-time scaling of one FPSA-R checkpoint trained with $T = 32$ (study configuration). Top: exact-match accuracy against the inference iteration budget. Bottom: mean relative fixed-point residual.

Models trained on 7×7 and 9×9 mazes extrapolate to larger ones, and the gap to a TRM-style block loop trained the same way grows with maze size, from 1.1 points on 9×9 to 6.2 on 11×11, 12.8 on 15×15, and 17.3 on 30×30 (Table 2, left). Figure 5 shows how one trajectory gets there: early iterations recover local pieces of the route, and later iterations connect them into a consistent path. Sudoku shows the same trend when puzzles are grouped by clue count (Table 2, right). The gap is 0.7 points for heavily constrained puzzles and 6.3 points for puzzles with at most 21 clues.

3.4 Transfer to Decoder-Only Language Models

We also test whether fixed-point attention helps outside grid puzzles, in pretrained decoder-only language models, where chain-of-thought is the standard way to spend extra inference compute. We add fixed-point self-attention (FPSA; Mathur et al., 2025a) to Gemma 3 12B and Mistral Small 4 (Appendix F) and compare it with chain-of-thought self-consistency at a strictly matched budget

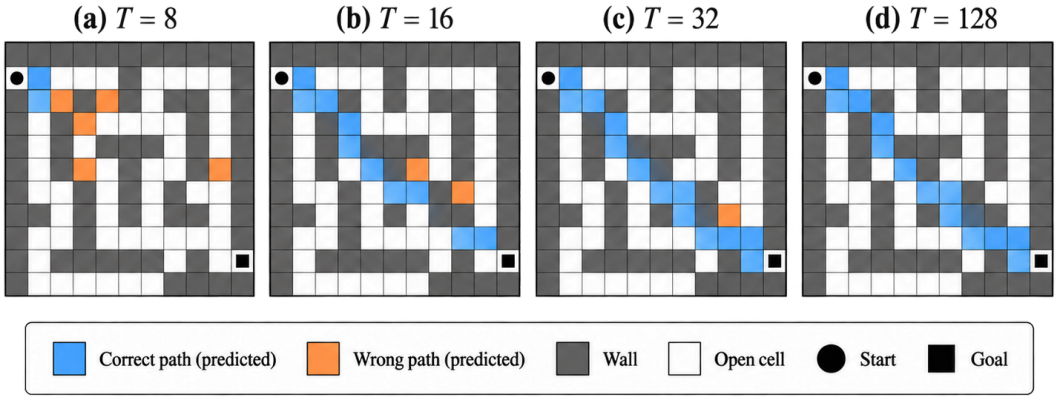

Figure 5: Progressive refinement on an unseen 11×11 maze by a model trained only on 7×7 and 9×9 grids. We decode the same recurrent trajectory after 8, 16, 32, and 128 iterations. Blue cells agree with the target path and orange cells do not.

of 1.5× direct-answer inference FLOPs (Figure 3). FPSA spends its extra compute on attention iterations during prefill and decodes no reasoning tokens, while chain-of-thought spends it on serial decoding.

On Gemma 3 12B, FPSA alone reaches 72.2 average accuracy without a single reasoning token, against 70.9 for self-consistency with 384 tokens, and it beats self-consistency on every benchmark. Mistral Small 4 shows the same order (79.5 against 78.7). Adding a 128-token chain of thought to FPSA gives the best result on both backbones (72.8 and 80.2), consistent with iterated attention resolving during prefill some of the routing that chain-of-thought would otherwise spell out token by token.

3.5 Recurrence Revises Answers, and Convergence Is Not Correctness

We decode the recurrent state after every iteration and follow each prediction until the solve stops. This shows what the extra iterations do, and whether reaching the fixed point tells us that the answer is right.

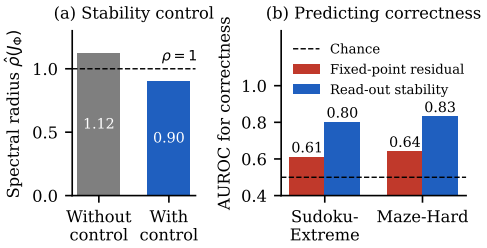

Figure 6: (a) Spectral radius of the fixed-point map at the end of training, without and with the stability control. (b) AUROC for predicting final correctness from the fixed-point residual and from read-out stability.

FPSA-R turns wrong answers into correct ones far more often than it breaks correct ones (Table 3; Appendix Figure 7). On Maze-Hard, 21.7% of examples start wrong and end correct, while only 1.2% start correct and end wrong. The block loop repairs fewer examples (14.9%) and breaks more (2.1%), and Sudoku-Extreme shows the same pattern.

Answers also settle before the recurrent state does. For trajectories that end correct, we record the first iteration with a correct answer (t_{first}), the iteration after which it stays correct (t_{stable}), and the iteration at which the residual reaches tolerance (t_ϵ). The median answer stops changing by iteration 18 on Sudoku and 16 on Maze, but the solver reaches tolerance only at 27 and 24.

Reaching the fixed point does not mean reaching the right answer. The stability control keeps the map contracting, with an estimated spectral radius of 0.90 against 1.12 without it (Figure 6a), so the solver reliably finds an equilibrium. That equilibrium is not always correct: 4.0% of Sudoku and 5.5% of Maze examples converge to a wrong answer, and the final residual predicts correctness with an AUROC of only 0.61 and 0.64, against 0.80 and 0.83 for read-out stability (Figure 6b). A small residual shows that the recurrent equation has been solved, as Section 2.2 guarantees, but

Table 3: Revision dynamics (study configuration). W→C: wrong to correct; C→W: correct to wrong; Conv.-wrong: numerically converged but incorrect. Iteration counts are medians over trajectories that end correct.

Task	Model	W→C	C→W	Conv.-wrong	t_{first}	t_{stable}	t_{ϵ}
Sudoku	**FPSA-R**	**18.4%**	**0.8%**	4.0%	14	18	27
Sudoku	Block loop	12.2%	1.6%	5.8%	16	22	31
Maze	**FPSA-R**	**21.7%**	**1.2%**	5.5%	12	16	24
Maze	Block loop	14.9%	2.1%	8.2%	15	21	29

Table 4: Gradient estimators with the forward model fixed (study configuration). Peak activation memory at $T = 128$; gradient cosine against full BPTT.

Estimator	Sudoku	Maze	Mem.	Grad. cos.	Seeds
Full BPTT	89.6 ± 0.5	88.5 ± 0.4	9.90 GB	1.0000	5
Implicit GMRES	$\mathbf{89.7 \pm 0.5}$	$\mathbf{88.6 \pm 0.4}$	1.15 GB	**0.9999**	5
Huginn-style random depth	87.9 ± 0.6	87.3 ± 0.5	3.40 GB	–	5
Phantom + penalty	89.0 ± 0.5	87.9 ± 0.5	1.40 GB	0.991	3
Truncated BPTT, $K = 8$	88.9 ± 0.5	87.8 ± 0.5	1.80 GB	0.982	3
Truncated BPTT, $K = 4$	88.2 ± 0.6	87.0 ± 0.5	1.50 GB	0.957	3
Truncated BPTT, $K = 1$	86.7 ± 0.6	85.9 ± 0.6	1.22 GB	0.893	3
$K = 0$ / one-step	85.4 ± 0.7	84.7 ± 0.7	**1.10 GB**	0.824	3

not that its solution answers the task. This matters for reasoners that halt on the residual, such as FPRM (Movahedi et al., 2026), and agrees with recent analyses showing that recurrent reasoners can converge to spurious fixed points (Rhoades, 1974; Helbling et al., 2026).

If the answer settles before the state, a model can stop on the answer (Appendix Table 6). Residual stopping at 10^{-4} gives the highest accuracy but runs 42.6 iterations on Sudoku. Stopping once the decoded answer has not changed for three iterations keeps 89.5% with 24.7 iterations, 42% fewer, and Maze shows the same trade: 88.4% at 21.6 iterations against 88.6% at 37.1.

3.6 Training Cost

We compare full BPTT, truncated BPTT, phantom gradients (Geng et al., 2021), Huginn-style random-depth training (Geiping et al., 2025a), and implicit differentiation with the forward model fixed (Table 4). For truncated BPTT, K is the number of recurrent transitions differentiated before the final map application, so $K = 0$ is the one-step estimator (Bolte et al., 2023). Implicit differentiation matches full BPTT within run-to-run variation, with gradient cosine above 0.9999, and needs 1.15 GB of peak activation memory at $T = 128$ against 9.90 GB. Full BPTT memory grows with depth and runs out at $T = 512$, while implicit differentiation stays near 1.2 GB (Appendix Figure 8 and Table 7), so it removes the tie between how deep the model recurs in training and how much memory it has.

4 Conclusion

We argue that recurrent reasoners should repeat attention more often than feedforward transformation. FPSA-R implements this by solving an answer-conditioned attention equilibrium before each feedforward update. With 7.0M parameters, it reaches 98.5% on Sudoku-Extreme, 95.2% on Maze-Hard, and 50.6%/15.3% on ARC-AGI-1/2, while outperforming matched block-loop controls. The gains grow with reasoning difficulty, and decoded-answer stability provides a better stopping signal than fixed-point convergence.

Our study focuses on short, structured reasoning tasks. For long contexts, attention cost may dominate and reduce the benefit of attention-only recurrence, motivating sparse or linear variants. More broadly, our results suggest that recurrent architectures should treat loop placement as a design choice: information may need to move more often than it needs to be transformed.

AI USE STATEMENT

In this work, generative AI tools were used for language editing, improving the clarity and organization of the manuscript, and assisting with the presentation and visualization of experimental results. Generative AI was not used to generate experimental measurements, fabricate results, determine benchmark outcomes, or make decisions about the scientific conclusions of the work.

All mathematical formulations, model designs, experimental protocols, reported results, and scientific claims were reviewed and verified by the authors. AI-assisted text and figures were checked against the underlying implementation, experimental logs, and source literature, and were revised or discarded when they did not accurately reflect the work. Any AI-assisted code or analysis used during development was inspected and validated by the authors before use.

The authors take full responsibility for the final content of this work, including all text, claims, code, figures, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 2 specifies FPSA-R completely, and Section 3.2 in Appendix B gives one refinement step together with the Anderson update, the residual used for stopping, and the implicit backward pass. The theoretical results in Section 2.2 state all of their assumptions, and Appendix C proves every claim, including the explicit Lipschitz constant behind Lemma 1. Appendix A describes the benchmarks, the data augmentation, the ARC evaluation protocol, every hyperparameter, and how we count FLOPs and measure latency and memory, and Appendix F does the same for the language-model experiments. All four reasoning benchmarks are public, and we use their published splits. The block-loop controls share every setting with FPSA-R except the recurrent operator, and reproduced results report the mean and standard deviation over 5 seeds. We provide anonymized code, training configurations, and the scripts that generate every table and figure as supplementary material.

REFERENCES

Donald G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12 (4):547–560, 1965.

Haoran Bai, Di Kang, Haoxian Zhang, Jinshan Pan, and Linchao Bao. FFHQ-UV: Normalized facial UV-texture dataset for 3D face reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 362–371, 2023.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. Deep equilibrium models. *Advances in Neural Information Processing Systems*, 32, 2019a.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *Advances in Neural Information Processing Systems*, volume 32, 2019b.

Andrea Banino, Jan Balaguer, and Charles Blundell. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Jérôme Bolte, Edouard Pauwels, and Samuel Vaiter. One-step differentiation of iterative algorithms. *Advances in Neural Information Processing Systems*, 36:77089–77103, 2023.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

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.

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

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. *arXiv preprint arXiv:2502.05171*, 2025a.

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. *arXiv preprint arXiv:2502.05171*, 2025b.

Zhengyang Geng, Xin-Yu Zhang, Shaojie Bai, Yisen Wang, and Zhouchen Lin. On training implicit models. *Advances in Neural Information Processing Systems*, 34:24247–24260, 2021.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

Alec Helbling, Andrey Bryutkin, Mauro Martino, Nima Dehmamy, and Hendrik Strobelt. Flow reasoning models: Scaling reasoning through iterative self-refinement. *arXiv preprint arXiv:2606.29150*, 2026.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 8340–8349, 2021.

Benhao Huang, Zhengyang Geng, and J. Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. In *International Conference on Machine Learning*, 2026.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pp. 4651–4664. PMLR, 2021.

Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention. In *International Conference on Machine Learning*, pp. 5562–5571. PMLR, 2021.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Tianle Lu, Ke Chen, and Yuping Duan. Total normal curvature regularization and its minimization for surface and image smoothing. *SIAM Journal on Imaging Sciences*, 19(2):879–912, 2026.

Mrinal Mathur and Sergey Plis. DynaLay: An introspective approach to dynamic layer selection for deep networks. *arXiv preprint arXiv:2312.12781*, 2023.

Mrinal Mathur, Mike Doan, Barak A. Pearlmutter, and Sergey Plis. Change of thought: Adaptive test-time computation. *arXiv preprint arXiv:2507.13569*, 2025a.

Mrinal Mathur, Barak A. Pearlmutter, and Sergey Plis. MIND over body: Adaptive thinking using dynamic computation. In *International Conference on Learning Representations*, 2025b.

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.

Peng Qiao, Sidun Liu, Tao Sun, Ke Yang, and Yong Dou. Towards vision transformer unrolling fixed-point algorithm: a case study on image restoration. *arXiv preprint arXiv:2301.12332*, 2023.

BE Rhoades. Fixed point iterations using infinite matrices. *Transactions of the American Mathematical Society*, pp. 161–176, 1974.

Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

Mark Schöne, Babak Rahmani, Heiner Kremer, Fabian Falck, Hitesh Ballani, and Jannes Gladrow. Implicit language models are rnns: Balancing parallelization and expressivity. *arXiv preprint arXiv:2502.07827*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. arxiv e-prints, art. *arXiv preprint arXiv:2104.09864*, 10, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

A Experimental Details

A.1 Benchmarks and Evaluation Protocol

Sudoku-Extreme. We use the Sudoku-Extreme benchmark of Wang et al. (2025): 1,000 training puzzles, expanded with validity-preserving augmentations (digit relabeling, band and stack permutations, row and column permutations within bands and stacks, and transposition), and the full held-out test split. A prediction is correct only if all 81 cells match the unique solution.

Maze-Hard. We use the 1,000 training and 1,000 test 30×30 mazes of Wang et al. (2025). The model labels every cell as on or off the shortest path, and a prediction is correct only if the full path matches the target. For the size-extrapolation study (Table 2), separate models train on 7×7 and 9×9 mazes generated with the same procedure and are evaluated without fine-tuning on 9×9 through 30×30 grids.

ARC-AGI-1 and ARC-AGI-2. We evaluate on the ARC-AGI-1 evaluation set (400 tasks) and the ARC-AGI-2 public evaluation set (120 tasks) (Chollet, 2019; Chollet et al., 2025). We follow the data construction and adaptation protocol of Jolicoeur-Martineau (2025) exactly and add no external training data. For ARC-AGI-1, training uses the ARC-AGI-1 training and concept sets together with the provided demonstration pairs from the evaluation tasks; ARC-AGI-2 is trained analogously from the ARC-AGI-2 training and concept sets and its evaluation-task demonstrations. We train the two benchmarks separately, since the ARC-AGI-2 training set overlaps with ARC-AGI-1 evaluation tasks. Held-out query outputs are never exposed as training labels.

At evaluation time, each task is presented under the canonical view and up to 1,000 distinct augmentations produced by dihedral transformations and color permutations, matching the TRM protocol. Random canvas translations are used only for training augmentation and are disabled on the evaluation split. Predictions from all views are mapped back through the inverse transformation and grouped by exact output grid. Candidates are ranked first by vote count and then by mean halt-head confidence to break ties. We report the standard pass@2 score: for each test query, the prediction is correct if either of the two highest-ranked candidate grids exactly matches the target; query accuracies are averaged within each task and then macro-averaged across tasks.

As in TRM, the model is therefore adapted using the provided demonstration pairs of each evaluation task; this is not a frozen-model unseen-task evaluation. We perform no additional per-query gradient updates, test-time fine-tuning, or optimization after the trained checkpoint is fixed. Because FPSA-R reaches 15.3% on ARC-AGI-2, compared with 7.8% for the published TRM-Att reference at the same model scale, we use the identical data builder, augmentation pool, candidate-voting procedure, and pass@2 evaluator for every reproduced model.

A.2 Study Configuration

The analyses isolate one variable at a time, and many of them require training every variant several times or unrolling the recurrence under full BPTT, which is expensive at the scale of the main model. We therefore run them in a lighter study configuration that differs from the main model in nothing. Absolute accuracies in this configuration sit below Table 1. Every analysis compares variants trained and evaluated in the same configuration, so the comparisons within a table or figure are controlled even though their absolute numbers differ from the headline results.

A.3 Compute Accounting

FLOPs. We count two FLOPs per multiply–accumulate (Geiping et al., 2025b; Snell et al., 2024) using a custom execution-aware FLOP counter applied identically to every reproduced model. The counter records the operations actually executed by the forward pass, including all recurrent-map evaluations, attention projections and matrix products, feedforward layers, halting and prediction heads, and the small linear-algebra operations used by the fixed-point solver. For Anderson acceleration, we additionally count construction of the residual Gram matrix and solution of the regularized least-squares system.

Unless stated otherwise, FLOPs are measured with batch size one and averaged over the full evaluation set. This choice is important for adaptive models: at batch size one, stopping after T_i iterations for example i corresponds directly to T_i executed recurrent-map evaluations. In a dense multi-example batch, converged examples remain in the tensor until the slowest example finishes, so their logical halting does not imply proportional hardware savings. We therefore use single-example realized FLOPs for compute matching and report batched wall-clock measurements separately. Section 2.3 derives the dominant per-iteration arithmetic in closed form.

Latency and memory. Latency is measured on one NVIDIA A100 80GB GPU at batch size 256 using CUDA events, with an explicit synchronization before and after each measurement. We report the median over 1,000 timed batches following 100 warm-up batches. All compared models use the same PyTorch 2.x/CUDA environment, eager execution (`torch.compile` disabled), and identical kernels where applicable. Transformer matrix multiplications use BF16, while fixed-point states and the small Anderson/GMRES linear-algebra operations are evaluated in FP32 for numerical stability.

For Tables 4 and 7, activation memory is measured using `torch.autograd.graph.saved_tensors_hooks`, summing the storage of tensors retained for backward rather than allocator-reserved memory. This isolates the quantity relevant to the $O(T)$ versus $O(1)$ differentiation comparison. The memory-depth experiment is run on an A100 40GB. Full BPTT stores 18.9 GB of activations at $T = 256$; doubling the unrolled depth pushes the complete training footprint beyond device capacity at $T = 512$, whereas the implicit variant remains approximately constant. The OOM entry therefore refers to the complete training execution on this device, not to an extrapolated activation-only number.

B Solving and Differentiating the Equilibrium

Anderson acceleration. Write $G(R) = \Phi_\ell(R; A, V) - R$ for the fixed-point residual map. Given the last $m_k = \min(m, k)$ iterates, Anderson acceleration (Anderson, 1965; Lu et al., 2026) solves the small regularized least-squares problem

$$\alpha^{(k)} = \underset{\alpha \in \mathbb{R}^{m_k+1},\ \mathbf{1}^\top \alpha = 1}{\arg\min} \left\| \sum_{j=0}^{m_k} \alpha_j \, G\big(R^{(k-j)}\big) \right\|^2 + \lambda_{\mathrm{AA}} \|\alpha\|^2 \tag{15}$$

and sets $R^{(k+1)} = \sum_j \alpha_j^{(k)} \Phi_\ell(R^{(k-j)})$. We use

$$\lambda_{\mathrm{AA}} = 10^{-4}, \qquad m = 6,$$

with damping 0.8 on the accelerated candidate. Each iteration requires one new evaluation of Φ_ℓ plus $O(m^2 N d + m^3)$ work for the Gram matrix and the small dense solve, which is negligible relative to attention when $m \ll d$. The solve is warm-started from the recurrent state carried from the preceding refinement step; the first refinement is initialized from the zero carry.

We evaluate the normalized residual $r(R)$ of Eq. 7 on the exact state returned by the solver, not on the displacement between successive accelerated iterates. Anderson mixing can make two successive iterates close before the fixed-point equation itself is accurately solved, so a displacement-based criterion would overstate convergence. We therefore re-evaluate Φ_ℓ on the returned state and report its true residual. Taking the maximum over positions requires every position in an example, rather than only the mean position, to satisfy the tolerance.

Propositions 1 and 2 require local stability of the recurrent map on the region visited by the solver. We estimate the dominant local mode of J_Φ using eight steps of power iteration implemented with Jacobian–vector products, so the Jacobian is never materialized. Without stability control, the measured local spectral radius rises to approximately 1.12 during training.

We regularize this quantity with the one-sided penalty

$$\mathcal{L}_{\text{stab}} = \lambda_\rho \left[\max(0, \hat{\rho}(J_\Phi) - \rho_0)\right]^2, \qquad \rho_0 = 0.90, \quad \lambda_\rho = 50, \tag{16}$$

where $\hat{\rho}(J_\Phi)$ is the power-iteration estimate. The penalty leaves the map unconstrained while it remains below the target and acts only when the estimated dominant mode approaches or exceeds the stability boundary. In our audit this reduces the measured local spectral radius from approximately 1.12 to 0.90 (Figure 6a), with no measurable loss in task accuracy. We emphasize that this is a local empirical stability control, not a certificate of global contraction. Related work regularizes the Jacobian of equilibrium models directly (Bai et al., 2023; 2019a).

At the returned state $\hat{R}$, we solve the adjoint system

$$(I - J_\Phi^\top)u = \frac{\partial \mathcal{L}}{\partial R^\star}$$

(Eq. 2) using restarted GMRES (Saad & Schultz, 1986). We use restart length 16, a maximum of 64 Krylov iterations, and relative residual tolerance 10^{-5}. GMRES requires only products $v \mapsto J_\Phi^\top v$, which reverse-mode automatic differentiation evaluates as a vector–Jacobian product through one application of Φ_ℓ; J_Φ is never explicitly formed. The parameter gradient then follows from one additional vector–Jacobian product (Eq. 2).

Forward activations from the recurrent trajectory are discarded, so the activation-memory cost is independent of the number of forward iterations, apart from the fixed Krylov workspace. We use no silent numerical fallback. If the forward solve fails to satisfy ϵ_{fp}, or GMRES does not reach relative residual 10^{-5} within 64 iterations, the optimization step is rejected and a diagnostic record is written. We therefore never attach an implicit gradient to a state that has not passed the forward convergence test, nor replace a failed adjoint solve with its last iterate.

Training objective. We supervise the prediction after every refinement step (Eq. 10) using StableMax cross-entropy as $\mathcal{L}_{\text{task}}$. Refinement steps receive equal task-loss weight, $w_s = 1$. Following Jolicoeur-Martineau (2025), the halting target c_s indicates whether the entire decoded answer $\hat{y}_s$ is exactly correct, and $\mathcal{L}_{\text{halt}}$ is binary cross-entropy on the halting logit q_s, with

$$\lambda_h = 0.5.$$

The detached carry of Eq. 11 bounds each backward graph to one refinement step while preserving the current answer and recurrent state as the initialization of the next step.

C Proofs

Throughout, $\|\cdot\|$ is the Frobenius norm on $\mathbb{R}^{N\times d}$ and $\|\cdot\|_2$ the spectral norm. For a row-wise map, a row-wise Lipschitz bound $\|f(z) - f(z')\|_2 \le c\,\|z - z'\|_2$ lifts to the same bound in Frobenius norm by summing squares over rows. We drop the stage index ℓ.

C.1 Proof of Lemma 1

We bound the Lipschitz constant of each map in the composition

$$R \;\mapsto\; H \;\mapsto\; (Q_h, K_h) \;\mapsto\; (\widehat{Q}_h, \widehat{K}_h) \;\mapsto\; S_h \;\mapsto\; P_h \;\mapsto\; \Phi(R), \qquad S_h = \widehat{Q}_h \widehat{K}_h^\top / \tau_h,$$

on $\mathcal{U}$, and multiply the constants.

Row normalization. For $z, z' \in \mathbb{R}^d$ with $\|z\|_2, \|z'\|_2 \geq \nu$,

$$\left\| \frac{z}{\|z\|_2} - \frac{z'}{\|z'\|_2} \right\|_2 \leq \frac{\|z-z'\|_2}{\|z\|_2} + \|z'\|_2 \left| \frac{1}{\|z\|_2} - \frac{1}{\|z'\|_2} \right| = \frac{\|z-z'\|_2 + \big|\|z'\|_2 - \|z\|_2\big|}{\|z\|_2} \leq \frac{2}{\nu}\|z - z'\|_2.$$

RMSNorm rescales the unit row by $\sqrt{d}$ and a learned gain g, so it is $c_H = 2\sqrt{d}\,\|g\|_\infty/\nu$-Lipschitz on rows of norm at least ν. Because the anchor A is fixed during a solve, $\|H(R) - H(R')\| \leq c_H\|R - R'\|$.

Projections and RoPE. RoPE applies a position-dependent orthogonal rotation to each row, so $\|Q_h(R) - Q_h(R')\| \leq \|W_Q^h\|_2\,\|H(R) - H(R')\|$, and likewise for K_h.

Query-key normalization. By the same argument as above, $\|\widehat{Q}_h(R) - \widehat{Q}_h(R')\| \leq (2/\nu)\|Q_h(R) - Q_h(R')\|$, and likewise for $\widehat{K}_h$.

Logits. Every row of $\widehat{K}_h$ has unit norm, so $\|\widehat{K}_h\|_2 \leq \|\widehat{K}_h\| = \sqrt{N}$, and the same holds for $\widehat{Q}_h$. Writing $S_h - S'_h = \big[(\widehat{Q}_h - \widehat{Q}'_h)\widehat{K}_h^\top + \widehat{Q}'_h(\widehat{K}_h - \widehat{K}'_h)^\top\big]/\tau_h$ gives

$$\|S_h - S'_h\| \leq \frac{\sqrt{N}}{\tau_h}\Big(\|\widehat{Q}_h - \widehat{Q}'_h\| + \|\widehat{K}_h - \widehat{K}'_h\|\Big).$$

Softmax. The Jacobian of a row-wise softmax at probabilities p is $\mathrm{diag}(p) - pp^\top$. It is positive semidefinite and bounded above by $\mathrm{diag}(p)$, so its spectral norm is at most $\max_i p_i \leq 1$. Softmax is therefore 1-Lipschitz row-wise, and $\|P_h - P'_h\| \leq \|S_h - S'_h\|$.

Values and output. Since V_h is a column block of V, $\|(P_h - P'_h)V_h\| \leq \|P_h - P'_h\|\,\|V\|_2$. Concatenation over heads and the output projection give

$$\|\Phi(R) - \Phi(R')\| \leq \beta\|W_O\|_2\Big(\sum_h \|(P_h - P'_h)V_h\|^2\Big)^{1/2} \leq \beta\|W_O\|_2\|V\|_2\sqrt{H}\,\max_h\|P_h - P'_h\|.$$

Chaining the bounds,

$$\|\Phi(R)-\Phi(R')\| \leq \frac{\beta\,\|W_O\|_2\,\|V\|_2}{\min_h \tau_h}\,\kappa\,\|R-R'\|, \qquad \kappa = \frac{4\sqrt{HNd}\,\|g\|_\infty}{\nu^2}\,\max_h\big(\|W_Q^h\|_2+\|W_K^h\|_2\big), \tag{17}$$

which is linear in $\beta\|W_O\|_2\|V\|_2$ and in $1/\min_h \tau_h$, as claimed. □

Remarks. Without query-key normalization, the logits scale with $\|Q_h\|_2\|K_h\|_2$, which grows without bound as the state grows, and dot-product self-attention has no finite global Lipschitz constant (Kim et al., 2021). Normalization caps the logits at $1/\tau_h$ and removes this growth. The constant in Eq. 17 is a worst-case bound and is loose in practice, since it ignores the saturation of softmax and the alignment of the weight matrices. The quantity that governs convergence is the local spectral radius of J_Φ at the fixed point, which we estimate directly (Figure 6a).

C.2 Proof of Proposition 1

Uniqueness. If $R_1, R_2 \in \mathcal{U}$ are both fixed points, then $\|R_1 - R_2\| = \|\Phi(R_1) - \Phi(R_2)\| \leq \gamma\|R_1 - R_2\|$, and $\gamma < 1$ forces $R_1 = R_2$.

State error. For $R \in \mathcal{U}$,

$$\|R - R^\star\| \leq \|R - \Phi(R)\| + \|\Phi(R) - \Phi(R^\star)\| \leq \|\Phi(R) - R\| + \gamma\|R - R^\star\|,$$

and rearranging gives $\|R - R^\star\| \leq \|\Phi(R) - R\|/(1 - \gamma)$.

From the normalized residual. By the definition of $r(R)$ as a maximum over positions (Eq. 6), every row satisfies $\|\Phi(R)_i - R_i\|_2 \leq r(R)\,(\|\Phi(R)_i\|_2 + \|R_i\|_2 + \epsilon)$. Summing squares over rows gives $\|\Phi(R) - R\| \leq r(R)\,c(R)$. □

C.3 Proof of Proposition 2

We make the smoothness assumptions of the proposition explicit. On $\mathcal{U}$: (a) Φ is differentiable in R with $\|J(R)\|_2 \le \gamma$, which the γ-Lipschitz property implies at interior points; (b) $\|b(R)\| \le B_L$ and $\|\partial_\theta \Phi(R)\|_2 \le B_\Phi$, where $b(R) = \partial\mathcal{L}/\partial R$; and (c) the maps $R \mapsto J(R)$, $b(R)$, $\partial_\theta \mathcal{L}(R)$, and $\partial_\theta \Phi(R)$ are Lipschitz with constants L_J, L_b, L_θ, and $L_{\Phi\theta}$.

Let $M(R) = I - J(R)^\top$. Since $\|J(R)\|_2 \le \gamma < 1$, the Neumann series gives $\|M(R)^{-1}\|_2 \le \sum_{k\ge 0} \gamma^k = 1/(1-\gamma)$. The exact gradient is $\nabla_\theta \mathcal{L} = \partial_\theta \mathcal{L}(R^\star) + u^{\star\top} \partial_\theta \Phi(R^\star)$ with $u^\star = M(R^\star)^{-1} b(R^\star)$, by the implicit function theorem, which applies because $M(R^\star)$ is invertible. The computed gradient is $\hat{g} = \partial_\theta \mathcal{L}(\hat{R}) + \hat{u}^\top \partial_\theta \Phi(\hat{R})$, where GMRES returns $\hat{u}$ with $\|M(\hat{R})\hat{u} - b(\hat{R})\| \le \eta \|b(\hat{R})\|$.

Solver error. Let $\bar{u} = M(\hat{R})^{-1} b(\hat{R})$ be the exact adjoint at $\hat{R}$. Then

$$\|\hat{u} - \bar{u}\| = \|M(\hat{R})^{-1}\big(M(\hat{R})\hat{u} - b(\hat{R})\big)\| \le \frac{\eta B_L}{1-\gamma}.$$

Linearization error. Using $A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1}$,

$$\|\bar{u} - u^\star\| \le \|M(\hat{R})^{-1}\big(b(\hat{R}) - b(R^\star)\big)\| + \|\big(M(\hat{R})^{-1} - M(R^\star)^{-1}\big) b(R^\star)\| \le \frac{L_b\,\delta}{1-\gamma} + \frac{L_J B_L\,\delta}{(1-\gamma)^2}.$$

Gradient error. With $\|u^\star\| \le B_L/(1-\gamma)$,

$$\|\hat{g} - \nabla_\theta \mathcal{L}\| \le L_\theta \delta + B_\Phi \|\hat{u} - u^\star\| + \|u^\star\| L_{\Phi\theta}\delta \le L_\theta \delta + B_\Phi \Big(\frac{\eta B_L}{1-\gamma} + \frac{L_b \delta}{1-\gamma} + \frac{L_J B_L \delta}{(1-\gamma)^2}\Big) + \frac{B_L L_{\Phi\theta}\delta}{1-\gamma}.$$

Since $1 - \gamma \le 1$, every term in δ is at most a constant times $\delta/(1-\gamma)^2$, which gives the claim with $C = \max\{L_\theta + B_\Phi L_b + B_\Phi B_L L_J + B_L L_{\Phi\theta},\ B_\Phi B_L\}$. □

Corollary (gradient error in terms of the residual). Combining Propositions 1 and 2, the gradient computed at a state with normalized residual $r(\hat{R})$ satisfies

$$\|\hat{g} - \nabla_\theta \mathcal{L}\| \le C\Big(\frac{r(\hat{R})\, c(\hat{R})}{(1-\gamma)^3} + \frac{\eta}{1-\gamma}\Big).$$

The cubic dependence on $1/(1-\gamma)$ explains why the gradient degrades sharply as the map approaches the contraction boundary, and why keeping the spectral radius near 0.90 rather than just below 1 matters for training.

D Additional Results

Test-time scaling. Table 5 gives the values plotted in Figure 4. Accuracy at the training depth ($T = 32$) is 5.1 points below its saturated value on Sudoku and 1.8 points below on Maze, so a model trained with a short budget leaves accuracy that more test-time iterations recover. The residual falls roughly geometrically with the budget, by about a factor of four per doubling, which matches linear convergence of the solver on a contraction.

Stopping rules. Table 6 compares stopping rules on one checkpoint. Tightening the residual tolerance from 10^{-3} to 10^{-4} buys 0.8 points on Sudoku at the cost of 11.6 more iterations, while read-out stability recovers most of that accuracy with fewer iterations than either residual rule. The ACT head is slightly cheaper still, but it needs a trained halting head and halts on a learned estimate rather than on the observed trajectory.

Revision dynamics. Figure 7 traces how predictions change over iterations. Repairs accumulate quickly in the first 12 iterations and then level off, while breaks stay rare throughout, so the net gain of the recurrence is nearly its repair rate. The Maze-Hard example shows the typical mechanism: early iterations commit to locally plausible segments, and later iterations revise the segments that do not connect.

Table 5: Test-time scaling of one FPSA-R checkpoint trained with $T = 32$ (study configuration). Accuracy is exact match (%); residual is the mean normalized fixed-point residual.

$T_{\max}$	Sudoku	Maze	Sudoku residual	Maze residual
8	70.2	82.0	5.1×10^{-2}	6.4×10^{-2}
16	78.4	84.8	1.8×10^{-2}	2.4×10^{-2}
32	84.6	86.8	5.2×10^{-3}	7.0×10^{-3}
64	88.0	88.0	1.4×10^{-3}	1.9×10^{-3}
128	89.3	88.5	3.4×10^{-4}	4.7×10^{-4}
256	**89.7**	**88.6**	8.0×10^{-5}	1.1×10^{-4}
512	89.7	88.6	2.0×10^{-5}	2.8×10^{-5}

Table 6: Accuracy (%) and mean iterations for inference-time stopping rules on a fixed checkpoint (study configuration).

Stopping rule	Sudoku	Iters.	Maze	Iters.
Fixed $T = 16$	78.4	16.0	84.8	16.0
Fixed $T = 32$	84.6	32.0	86.8	32.0
Fixed $T = 64$	88.0	64.0	88.0	64.0
Residual 10^{-3}	88.9	31.0	88.3	27.2
Residual 10^{-4}	**89.7**	42.6	**88.6**	37.1
Read-out stable, 3 steps	89.5	24.7	88.4	21.6
ACT head	89.4	**22.9**	88.3	**20.4**

Gradient propagation and memory. Figure 8 complements Table 4. In the block loop, the gradient reaching early iterations decays by several orders of magnitude, so early iterations receive almost no training signal. In FPSA-R it stays nearly constant, because the implicit gradient does not propagate through the unrolled trajectory at all. Table 7 gives the memory values.

Table 7: Peak training activation memory as recurrent depth increases.

T	Full BPTT	Implicit GMRES
8	1.46 GB	1.10 GB
16	2.02 GB	1.10 GB
32	3.15 GB	1.11 GB
64	5.40 GB	1.12 GB
128	9.90 GB	1.15 GB
256	18.90 GB	1.18 GB
512	OOM	1.24 GB

E Component Ablations

We change one component of FPSA-R at a time (Figure 9). Each variant is compared with the full model trained in the same study configuration, so absolute accuracies sit below Table 1.

Recurrence is the largest single source of accuracy. Limiting each stage to one attention iteration ($T = 1$) keeps the stages and the outer refinement steps but removes the fixed-point solve. It costs 13.3 points on Sudoku-Extreme, 15.9 on Maze-Hard, and 22.2 on ARC-AGI-1. The drop is largest on ARC, where the model must keep revising which parts of the demonstrations to relate.

The recurrence updates routing, not evidence. Freezing the keys, so that only the queries change across iterations, costs 5.4, 5.6, and 6.9 points: the model needs both sides of the attention alignment to move. Letting the values evolve as well, the opposite change, also hurts, by 2.5, 1.8, and 1.4 points, while adding compute at every iteration. Holding the evidence fixed while the alignment changes is better than freezing the alignment or letting the evidence drift.

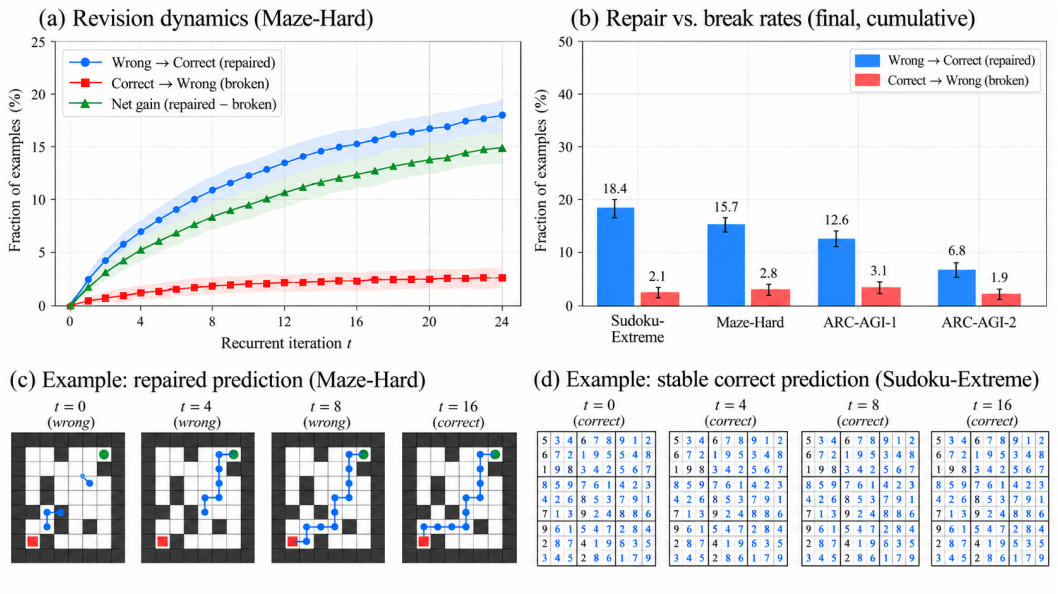

Figure 7: Recurrence repairs far more answers than it breaks. (a) Cumulative fraction of Maze-Hard examples repaired (wrong to correct) and broken (correct to wrong) by iteration t, and their difference. (b) Final repair and break rates on all four benchmarks. (c) A Maze-Hard prediction that is wrong at $t = 0$, 4, and 8 and correct by $t = 16$. (d) A Sudoku-Extreme prediction that is correct from $t = 0$ and stays correct. Shaded regions and error bars show one standard deviation over 5 seeds.

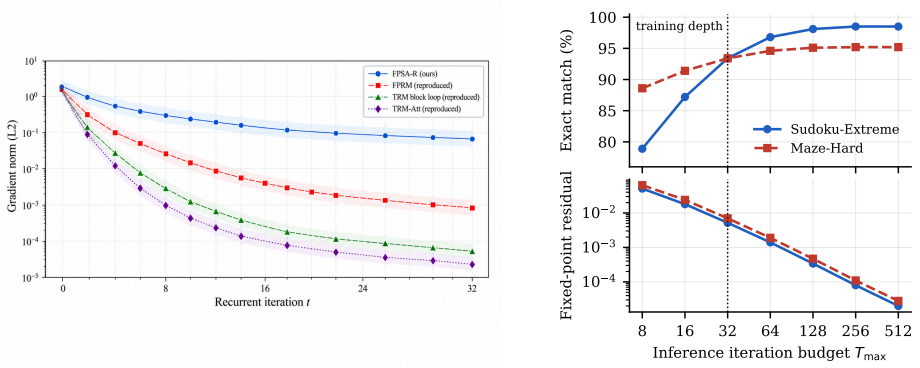

Figure 8: (a) Gradient norm with respect to the recurrent state across iterations. (b) Peak training memory against recurrent depth: full BPTT grows with depth and runs out of memory at $T = 512$, while implicit differentiation stays near 1.2 GB.

The MLP belongs at the answer timescale. The width-matched attention+MLP loop spends 27% more compute than FPSA-R and still loses 7.5 points on Sudoku-Extreme, 8.4 on Maze-Hard, and 14.1 on ARC-AGI-1, and the FLOP-matched version loses 9.2, 11.0, and 16.8. At matched width, a feedforward update at every attention step makes the model worse, so the result reflects more than compute efficiency. An MLP inside the map rewrites the representation that attention reads at every iteration, which is exactly what fixed values prevent, and the evolving-V result points the same way.

The LSE feature gives a smaller but consistent gain. Removing the LSE feature (Section 2.1) costs 2.2 to 2.5 points across the three tasks. Replacing it with a plain LSE term costs more, 2.8 to 3.3 points, than removing it entirely, so the gain comes from the specific form of the feature rather than from adding an LSE term at all.

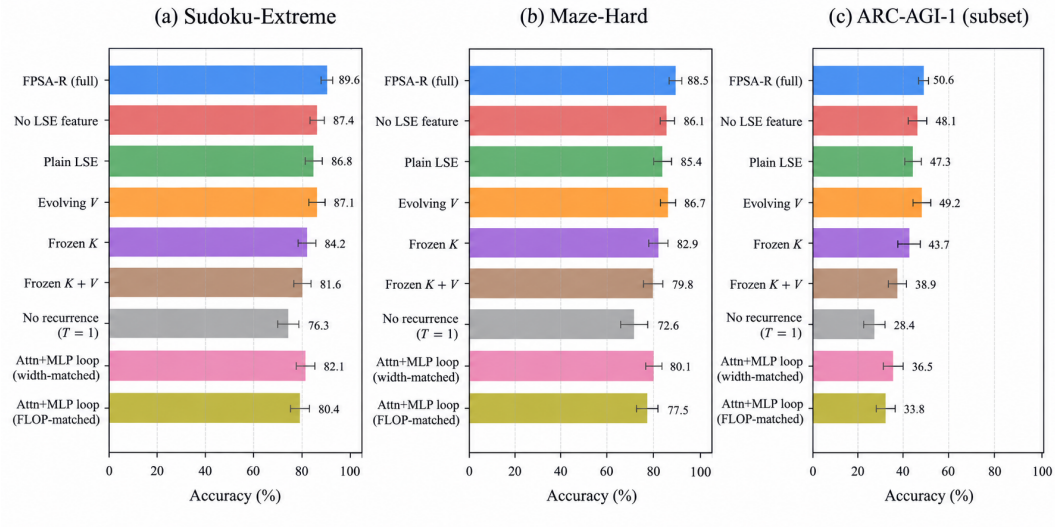

Figure 9: Component ablations. Each bar changes one part of the full model. All variants use the study configuration and train with full BPTT, so the gradient estimator is held fixed. ARC-AGI-1 uses a fixed subset of 10 evaluation tasks. Error bars show one standard deviation over 3 seeds.

Across the three tasks the ranking of the interventions is stable, and most interventions cost the most on ARC-AGI-1. This agrees with the main results, where ARC and Maze-Hard gain the most from attention recurrence.

F Transfer to Decoder-Only Language Models

Integration. We insert fixed-point self-attention into the global-attention layers of Gemma 3 12B and into all causal self-attention layers of Mistral Small 4. For a pre-normalized layer input H, we compute the value bank once,

$$V = \mathrm{RMSNorm}(H)W_V,$$

and hold it fixed throughout the inner solve. We initialize the recurrent residual state with $R^{(0)} = 0$, so the first query/key state is exactly the pretrained layer representation, and recompute only Q and K as the state evolves. The standard causal mask is applied at every inner iteration.

FPSA is applied only during prompt prefill. After convergence, we retain the refined prompt representation and its resulting KV cache; autoregressive generation then proceeds token by token with the original pretrained attention operator, without additional FPSA iterations on generated tokens. This isolates latent refinement to prefill and avoids multiplying the serial cost of autoregressive decoding.

Fine-tuning. Every method, including the chain-of-thought baselines, starts from the same instruction-tuned checkpoint and is fine-tuned with rank-16 LoRA ($\alpha = 32$, dropout 0.05) on $\{q_{\mathrm{proj}}, k_{\mathrm{proj}}, v_{\mathrm{proj}}, o_{\mathrm{proj}}\}$. We train on a fixed 200k-example decontaminated reasoning mixture for 5,000 optimizer steps with global batch size 128, AdamW at 2×10^{-5}, 3% linear warm-up, followed by cosine decay. All benchmark evaluation examples and near-duplicates are removed from the fine-tuning mixture. Thus, within each backbone, all methods receive the same training exposure and the same trainable-parameter budget.

FLOP matching. Let F_{direct} denote the measured inference FLOPs of direct answering, including prompt prefill and answer decoding. Every matched method is calibrated to $1.5F_{\mathrm{direct}}$ per example on a held-out split. Chain-of-thought self-consistency uses three independently sampled reasoning traces followed by majority voting. The per-trace token cap is chosen separately for each backbone to satisfy the same profiled-FLOP budget: 105 generated tokens per trace for Gemma 3 12B and

151 for Mistral Small 4. Prompt prefill is counted once and its KV cache is shared across the three trajectories.

FPSA spends the additional budget entirely on attention refinement during prefill and generates no reasoning tokens. Its iteration cap is selected on the same calibration split so that mean realized inference FLOPs fall within $\pm 2\%$ of $1.5F_{\text{direct}}$.

FPSA with short chain-of-thought divides the same budget between latent refinement and explicit generation. The calibrated generation budgets are 96 tokens for Gemma 3 12B and 129 tokens for Mistral Small 4, with the remaining budget allocated to FPSA prefill iterations. We count prefill, FPSA iterations, KV-cache operations, and every generated token with the same profiler used in Appendix A.

G LIMITATIONS

Guarantees are local and conditional. Propositions 1 and 2 hold on a region where the map contracts, and they weaken as γ approaches 1. The Lipschitz constant of Lemma 1 is a worst-case bound, so in practice we rely on the measured spectral radius rather than on the bound to confirm contraction. None of these results says anything about whether the equilibrium is the correct answer, and Section 3.5 shows that it sometimes is not.

Stopping on read-out stability is a heuristic. Read-out stability predicts correctness far better than the residual, but it cannot detect an answer that is stable and wrong, which is the case for the 4.0% of Sudoku and 5.5% of Maze examples that converge to incorrect solutions.

Language-model transfer. The language-model experiments test the fixed-point attention layer, not the full answer-conditioned reasoner, on two backbones and four benchmarks at a single budget. Whether answer conditioning and the two-timescale design transfer to autoregressive models remains open.

Latency. We measure latency on one GPU type at one batch size. Batched adaptive stopping can leave examples that have already converged waiting for the slowest example in the batch, so realized speedups depend on the serving implementation.