LEARNED CORRECTION DYNAMICS IN RECURSIVE REASONING MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive reasoning models achieve strong performance on certain reasoning tasks by repeatedly applying a small neural network to an evolving latent state. Yet how recursion contributes to reasoning, what successive iterations accomplish, and when additional computation improves performance remain incompletely understood. We address these questions through trajectory measurements and controlled interventions on Sudoku-Extreme, studying released models alongside our compact solvers. We develop an empirical account of learned correction dynamics. Recursion can revise many cells while making little net progress before abrupt completion. Model-produced and randomly corrupted grids can differ in repairability at equal error counts. Continued communication and retained state support correction. Our controls separate solving capability from access through initialization, and finding correct answers from selecting them. Training comparisons show that useful generation and reliable selection need not be learned together. The repair and state findings also hold in our compact attention models, which achieve 99.48% with 0.62M parameters and 99.62% with 1.06M and 1.97M on all 422,786 test puzzles. Using 1,000 training puzzles and one fixed-start trajectory of 64 iterations, all exceed, to our knowledge, the best comparable published score of 98.22% with less matrix multiplication work. Extending the analysis to ARC-AGI reveals a complementary limit: a released model's fit to training demonstrations does not ensure held-out query success, and tested pools lack correct candidates for most queries. The Sudoku findings motivate a conjecture of learned relaxation with memory: retained or reconstructed context could support coordinated revision of mutually supporting errors; on ARC, revising task representations alongside answers could generate new solutions.

1 INTRODUCTION

Recursive reasoning models reuse a small neural network to update a state from which answers are read. The Hierarchical Reasoning Model (HRM) and Tiny Recursive Model (TRM) extend computation through this parameter reuse (Wang et al., 2025; Jolicoeur-Martineau, 2025). On Sudoku-Extreme, compact networks must learn from few examples to make digit assignments jointly consistent with clues and one another. Updates can correct assignments, preserve errors or undo success. We ask **what successive iterations accomplish, what makes correction possible, and when additional computation improves the returned answer**.

We study released models alongside our compact Sudoku solvers, which use the symbol-indexed representation of Symbol-Equivariant Recurrent Reasoning Models (SE-RRM) (Freinschlag et al., 2026); our attention variants also use its attention operations. Trained on 1,000 puzzles, our 0.62M-parameter attention model solves **99.48%** of all 422,786 Sudoku-Extreme test puzzles at 64 iterations; 1.06M- and 1.97M-parameter models reach **99.62%**, with one training seed per size and one fixed-start trajectory per puzzle, without external search. To our knowledge, all exceed the best published result under this evaluation protocol, SE-RRM's 98.22%. The 1.06M model uses roughly half as many parameters and half the arithmetic work in matrix multiplications.

Building on studies of refinement, state roles and inference scaling (Rodrigues & Kang, 2026; Miyanishi & Morimura, 2026; Huang et al., 2026; Sghaier et al., 2026), we develop an empirical account of **learned, state-dependent correction dynamics**. Our controls separate difficulty repairing

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.

a state from failures to access a successful trajectory or select an available answer. We make three novel, connected contributions:

Error configuration shapes repair beyond error count. We distinguish offsetting early improvements and regressions from large late corrections. Model-produced and randomly corrupted grids can differ in repairability at equal wrong-cell counts and, in our role embeddings, equal distance to the solution. Synthetic corruptions arranged to reduce conflicts among assignments reproduce much of the early repair difficulty. From identical first states, replaying earlier messages preserves correct answers but supports few new discoveries; state features beyond answer scores also influence correction.

Separating solving capability, access and selection. At a multilayer perceptron (MLP) checkpoint trained beyond the benchmark's 50,000 optimizer updates, one answer-independent Gaussian start shared across 128 validation puzzles raises success from the default start's 11 to 124 at unchanged weights and depth, without puzzle-specific search. In our TRM implementation, expanding candidate pools returns fewer correct answers despite finding more: new wrong candidates have smaller state changes and displace earlier correct choices whose scores are unchanged. Training comparisons separate useful generation from reliable ranking. These failures are model-dependent: benchmark attention models already have effective fixed starts and return a correct answer for almost every puzzle with a correct candidate available in its tested pool.

Compact solvers with directly tested correction dynamics. Our attention family combines strong full-test accuracy and explicit arithmetic counts with repair and state interventions at benchmark weights. Supplementary benchmark weights, configurations and per-puzzle records support independent verification.

ARC-AGI tests this account when transformations must be inferred from examples (Chollet, 2019). In a released TRM exposed to evaluation-task demonstrations, demonstration fit does not ensure held-out query success. Finite searches leave many queries without correct candidates. The Sudoku findings motivate a conjecture, *learned relaxation with memory*: preserving or reconstructing context while messages evolve may support revision of difficult error configurations. A separate ARC hypothesis is that revising task representations alongside answers could add missing correct candidates.

2 Models and Experimental Setting

With weights θ fixed, a recursive model starts at $s_0 = I(x;\xi)$ and updates its state while retaining input x:

$$s_{t+1} = F_\theta\big(s_t; e_\theta(x)\big), \qquad \widehat{y}_t = \mathrm{decode}\,\big(R_\theta(s_t)\big). \tag{1}$$

Here I sets the initialization, ξ specifies any random draw, e_θ embeds the input, F_θ performs one outer iteration, and R_θ produces output scores. The resulting *trajectory* is observed from the first iteration; identical decoded grids can conceal different internal states. Appendix A.1 collects notation and definitions.

Our benchmark Sudoku solvers combine TRM's shared-network recurrence (Jolicoeur-Martineau, 2025) with SE-RRM's symbol-indexed representation (Freinschlag et al., 2026). Slow and fast state arrays, $s = (h, l)$ with $h, l \in \mathbb{R}^{9\times 81\times w}$, hold w feature channels per candidate digit and cell. A *field* collects one digit's features across cells. Each field embeds clues using three shared role vectors: blank, matching its digit, or another digit. Blocks mix positions within fields, exchange information between candidate digits at each cell (**cross-field communication**), and transform channels. MLP variants use position MLPs and projected means of other fields; attention variants use attention along both axes. Sharing parameters across digits makes the update equivariant to digit relabeling: permuting input digits and state fields permutes output digit scores in real arithmetic.

The shared two-block network B_θ defines the elementary update $U_\theta(z,u) = 0.05z + 0.95B_\theta(z+u)$. Each inner cycle applies $l \leftarrow U_\theta(l, h + e_\theta(x))$ six times, then $h \leftarrow U_\theta(h,l)$ once. The puzzle embedding enters only fast updates directly. Three cycles form one outer iteration: 21 applications of B_θ. *Recursion depth* D counts outer iterations; both states persist between them as *carries*. Fast and slow name update frequencies; their arrays are activations, not parameters. Attention weights are shared across fields, cells and updates. For the narrower models, smaller feature dimensions provide the principal parameter saving relative to SE-RRM; both families reuse weights across iterations. A shared linear *readout* maps h to scores for each cell's candidate digits. Decoding selects the

highest-scoring output token, without enforcing constraints or overwriting clues. Predictions remain revisable and **never become new clues**.

2.1 Training and evaluation protocols

Our benchmark solvers train on 1,000 base Sudoku-Extreme puzzles with positional augmentations. Following TRM and Equilibrium Reasoners (EqR) (Jolicoeur-Martineau, 2025; Huang et al., 2026), each outer iteration is supervised against the complete solution. Gradients stop between outer iterations and traverse only the final inner cycle. Training starts from Gaussian states and adds update noise. An auxiliary *answer-anchor* branch trains recovery from perturbed solutions. For each example, it samples $p \sim \mathrm{Uniform}[0, 0.2)$ and independently redraws each non-clue digit with probability p. The grid is embedded into h, with fixed initial l. One iteration is supervised against the solution with the original puzzle as input. Training trajectories are capped at 16 iterations.

Names specify mixer type and hidden dimension w: MLP 192 has three training seeds; Attention 128/192/256 have one each. From each 50,000-optimizer-update (50k) run, exponential-moving-average (EMA) weights are selected by depth-16 accuracy on 512 separate validation puzzles. The same weights are evaluated at depths 16 and 64 on all 422,786 test puzzles. Each evaluation returns **one noiseless fixed-start endpoint**, without voting or early halting. Fixed starts broadcast a constant feature vector per state across cells and fields. Exactness requires all 81 cells to match; cellwise error uses initially empty cells. Released TRM, Curriculum-Guided Adaptive Recursion (CGAR) and EqR use cell tokens rather than digit fields; they mix information across cells but lack this cross-field branch. Our common evaluator for these released models has documented native-execution differences. Our separately trained TRM uses cell tokens and MLP token mixing at hidden dimension 512. This model and the earlier 30k-budget attention ablations are distinct from the released models and benchmark attention checkpoints.

Dense multiply-accumulates (MACs) distinguish parameter economy from the arithmetic expended to reach an accuracy: billions per outer iteration (GMACs) give $D\,\mathrm{GMACs}/1000$ trillions per puzzle (TMACs). Counts include projections, attention products and readouts, excluding other operations and memory traffic; they measure neither runtime nor training cost. Appendix A.2 specifies architectures, recipes, numerical settings, split-equivalence checks and reproduction procedures; A.3 provides extended Sudoku results and controls for Sections 3–6.

Table 1: Accuracy and dense inference arithmetic of compact solvers. All models use 1,000 base training puzzles. A: all 422,786 test puzzles; our solvers return one fixed-start trajectory. Attention: one seed per model; MLP: three-seed means (ranges: 94.82–95.91% at 16; 98.82–99.18% at 64). Bold marks column bests in A at the displayed precision. †SE-RRM accuracy is published; MACs use a common counting convention and measure inference arithmetic, not runtime. B: original protocols; FRM's peak uses an inference sweep on 1,000 test puzzles. Sources: SE-RRM, Table 2, TRM, Tables 1/4, FPRM, Table 1 (Movahedi et al., 2026), FRM, Table 1/Appendix A.1. SE-RRM's separate RoPE variant reports 95.4% at 16 (Table A6).

Model	Params (M)	Accuracy (%) $D=16$	Accuracy (%) $D=64$	GMACs/ iteration	TMACs/puzzle $D=16$	TMACs/puzzle $D=64$
A. Full-test depth and arithmetic comparison						
Attention 128	**0.62**	95.45	99.48	**10.24**	**0.164**	**0.655**
Attention 192	1.06	97.31	**99.62**	17.22	0.275	1.102
Attention 256	1.97	**97.93**	**99.62**	31.47	0.504	2.014
MLP 192 (3 seeds)	0.79	95.41	99.05	14.67	0.235	0.939
SE-RRM†	1.97	93.73	98.22	34.65	0.554	2.218

B. Other published results; original inference protocols			
Model	Params (M)	Reported accuracy	Inference protocol
TRM-MLP	≈5	87.4%	Original TRM protocol
FPRM	≈7	94.2%	Adaptive fixed-point solver
FRM	≈7	99.5%	Peak; 1,000-puzzle subset

3 PERFORMANCE AND CORRECTION ACROSS ITERATIONS

Continuing a fixed-start trajectory raises Attention 128's full-test accuracy from 95.45% at 16 iterations to 99.48% at 64 with 0.62M parameters: recursion remains useful **beyond the 16-iteration training horizon**. All six benchmark runs exceed SE-RRM's published 98.22% at depth 64 with fewer dense MACs (Freinschlag et al., 2026) (Table 1). At 32 iterations, Attention 128 already reaches 98.46%, exceeding SE-RRM's reported depth-64 accuracy with approximately 15% of its counted dense MACs. Flow Reasoning Models (FRM) report 99.5% on a fixed 1,000-puzzle test subset over an inference sweep (Helbling et al., 2026), rather than this full-test depth-64 protocol. Lattice Deduction Transformers report 100% on 300 test puzzles using search with an abstention option (Davis et al., 2026).

The benefit of a larger model depends on the iteration budget. At depth 64, the 1.06M-parameter Attention 192 reaches 99.62%; Attention 256 solves only 5 more puzzles at almost twice the arithmetic cost. Paired full-test comparisons through a common evaluator favor MLP 192 seed 0 over released EqR, CGAR and Sudoku TRM. In earlier attention controls, changing batch size and learning rate together produces the largest tested accuracy drop. A TRM-cell control without digit fields, matched to MLP 192's parameter count, reaches 60.3% at depth 16 under its tested recipe, versus MLP 192's three-seed mean of 95.4%. These controls narrow possible contributors without isolating the advantage over SE-RRM.

3.1 CORRECTION BEFORE EXACT COMPLETION

In 32-iteration traces on 512 uniformly sampled test puzzles, one MLP 192 trajectory has 34, 21, 3, then 0 errors among 58 empty cells at iterations 1–4. Its first transition repairs 14 cells but makes one correct cell wrong. This example, selected at median completion time, illustrates revision hidden by whole-puzzle accuracy; its small error count just before success is not typical of the completing cohort.

Among the 287 MLP puzzles first solved at iterations 3–32, most of the mean error reduction before completion occurs in the last transition between incorrect predictions. Earlier corrections and regressions largely offset: wrong-to-right and right-to-wrong revisions contribute 26.94 and 23.14 events per 100 empty cells, counting repeated changes. Substantial revision thus produces little net progress. The completing iteration then makes a large correction: the median remaining error immediately before success is still 39.66%.

EqR, released Sudoku TRM and earlier Attention 256 traces show similar concentration near completion, each within its own completion cohort. The final attention models confirm this pattern on 512 stratified puzzles through depth 16; 17–31% of their completion cohorts have more errors just before success than after the first prediction.

Early loss reduction followed by abrupt completion has been observed in HRM (Ren & Liu, 2026). Our measurements show that **earlier revision and large late corrections coexist**. Inner-cycle probes of MLP 192, Attention 128 and EqR show partial score improvement before completion and further correction over the completing cycle's fast updates.

3.2 ANSWER PRESERVATION

Reaching a correct answer improves endpoint accuracy only if it is preserved or recovered. Across the full test through depth 64, temporary losses are rare in the benchmark attention models, and every puzzle solved at least once ends correct. We next test what makes incorrect states repairable.

4 STATE-DEPENDENT REPAIR

Repair means correcting wrong cell assignments; we measure both error reduction and exact completion. In *matched-error* comparisons on 512 difficulty-stratified test puzzles, paired grids have equal numbers of wrong cells. EqR produces a first-iteration grid from its fixed start; we restore clues and count its k wrong empty-cell digits. The paired random grid replaces exactly k randomly chosen empty cells of the known solution with uniformly sampled different digits. We test all three benchmark attention models, selected MLP 192 and released EqR as receivers. We sample four such

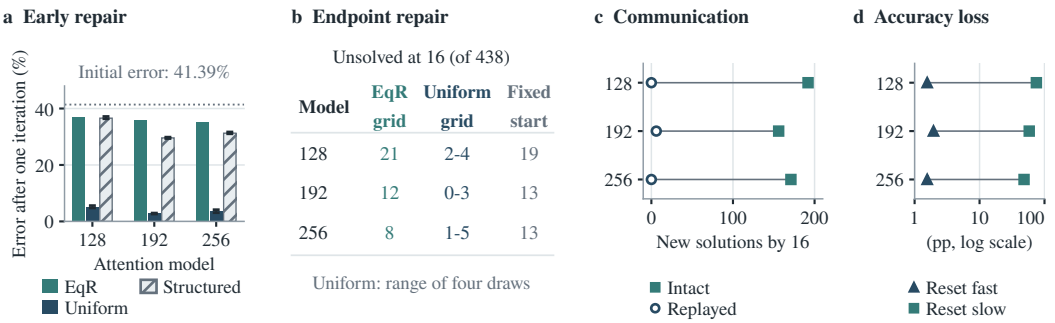


Figure 1: Configuration, communication and retained state constrain attention-model repair. Selected Attention 128/192/256 checkpoints. (a) On 438 puzzles at matched error counts, structured corruptions with reduced peer conflicts approach EqR-grid repair difficulty. Random-family bars show four-draw means and ranges, not confidence intervals; the dotted line marks initial error. (b) EqR/uniform endpoint differences narrow by iteration 16; uniform counts span four draws. Fixed starts provide context. (c,d) On 256 puzzles, continuations share the complete first state. (c) Replaying first-iteration messages in iterations 2–16 preserves all initial solutions but yields few new ones, unlike intact continuation. Removing messages instead returns no solution. (d) Resetting one carry before iterations 2–16 retains within-iteration computation. Points show net endpoint accuracy losses relative to intact continuation on a log axis.

corruptions (draws) per puzzle for attention and one for MLP and EqR. Each receiver embeds the grid into its slow state, uses its fixed initial fast state and runs 16 iterations on the original puzzle. This transfers the grid, not EqR's latent state.

Error configuration shapes repair beyond error count. In MLP and attention role embeddings, each wrong digit swaps two shared role vectors. Count-matching therefore also fixes Euclidean distance to the encoded solution and initial norm. Different repair outcomes cannot arise solely from being closer in that distance; EqR's digit-specific embedding lacks this identity.

On the 438 initially incorrect pairs, attention models show the clearest contrast in **early repair** (Figure 1a,b). After one iteration, mean empty-cell error is about 35–37% from EqR grids, versus 3–5% averaged over four random draws. Further iterations largely close this gap by iteration 16: configuration affects repair speed without requiring a large difference in eventual success. Supplying EqR's grid also gives no consistent endpoint improvement over ordinary fixed starts. In MLP and EqR, the uniform corruptions also retain larger endpoint advantages.

Only 0.17% of wrong EqR digits duplicate a given in their row, column or box, versus about 63% across uniform draws. Respecting givens, even at EqR's wrong-cell locations, reduces but does not remove the early repair gap. We therefore construct four further corruptions per puzzle that preserve clues and error count while reducing row, column and box conflicts among assignments. These structured grids leave 30–37% error after one attention iteration, reproducing much of the model-grid difficulty. MLP and EqR show the same direction, including EqR without answer-anchor training. Difficult repair does not require model-generated errors: **the arrangement of wrong assignments matters beyond their number**.

4.1 Continued communication

On 256 puzzles from the repair set, each attention model's complete first state from fixed initialization is continued through iteration 16. Intact continuation returns 250 correct answers per model. Zeroing both blocks' cross-field value projections during iterations 2–16 prevents every new discovery and loses all initial solutions. Other operations and puzzle input remain present, but message removal also withdraws digit-specific clue information.

To separate message availability from updating, we replay each puzzle's first-iteration messages at the corresponding network locations in iterations 2–16, while both states continue evolving. Every initially correct answer remains correct throughout, but new solutions fall to 0/6/0 for Attention 128/192/256 (Figure 1c). **Preservation and discovery respond differently to message replay**. Earlier messages suffice to maintain the correct answers, yet support little of the ordinary continuation's discovery. Replay retains input-derived messages but changes their relationship to evolving states; it does not identify what information those messages convey.

4.2 Dependence on retained state

Using the same first states and puzzles, we reset one state to its fixed initial value before every iteration 2–16, retaining the other. Both states still undergo all their usual within-iteration updates. Slow resets are much more damaging than fast resets (Figure 1d). Small fast-reset costs conceal paired gains and losses, not equivalence. Even under slow resets, the two wider attention models find new solutions that remain correct at the endpoint. Separate comparisons with MLP 192, an earlier MLP 384, our TRM and EqR show the same asymmetry.

To test correction beyond the displayed answer, we preserve slow-state digit scores and replace features invisible to the linear readout, retaining the fast state. Before iterations 2–16, we replace each such component with a random orthogonal vector of the same norm. Across attention models, correct endpoints fall from 250/256 to 171–230, with all initial solutions correct at the endpoint but fewer new solutions. Using the initial buffer's orthogonal components instead is more damaging. **State beyond the answer scores influences later correction**, but replacement choice matters: reset losses do not measure discarded history alone. Other retained quantities and within-iteration reconstruction can support continued repair. These controls complement state-count ablations and interventions on state and propagation (Jolicoeur-Martineau, 2025; Rodrigues & Kang, 2026; Miyanishi & Morimura, 2026). We next examine access through initialization.

5 Initialization and Access to Learned Capabilities

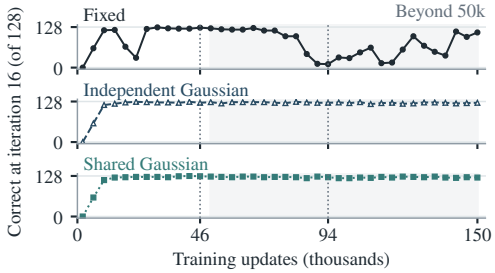
We examine 38 EMA checkpoints from 2k–150k optimizer updates in one MLP 192 run, using the same 128 puzzles from a separate validation set. Each condition runs one noiseless 16-iteration trajectory with weights, clues and subsequent operations fixed (Figure 2).

We replace fixed initial states with independent standard-normal entries in both arrays: one tensor pair per puzzle, or one pair shared across all puzzles. Draws are answer- and puzzle-independent, reused across checkpoints, without searching over starts. Training encounters Gaussian and answer-derived states, but neither branch uses the fixed state pair. At 94k, beyond the 50k benchmark training budget, the fixed start returns 11/128 correct answers; independent and shared Gaussian starts each return 124, preserving every fixed-start success. **One shared replacement suffices**, without per-puzzle search or extra iterations. This separates *conditional solving capability*, demonstrated from a tested answer-free start, from its *accessibility* under a starting rule. New feature vectors shared across cells, fields and puzzles also recover most successes; broadcasting alone does not force failure.

5.1 Initialization and answer preservation

The fixed start reaches 68 answers but loses 57 by iteration 16; neither full Gaussian condition loses a reached answer (Figure 2b). Even preserving every fixed-start discovery would leave it below the Gaussian conditions: initialization affects **both discovery and preservation**.

a Initialization across training



b 94k checkpoint: outcomes

Start	Ever correct	Correct at 16	Lost by 16
Fixed	68	**11**	57
Independent	124	**124**	0
Shared	124	**124**	0
Broadcast	125	**121**	4
Answer-injected control	128	**128**	0

Figure 2: Changing the start exposes conditional solving capability. (a) One MLP 192 run, all 38 checkpoints, the same 128 validation puzzles: one 16-iteration trajectory per puzzle and condition. Independent and shared Gaussian starts are reused across checkpoints, without searching over starts. Guides mark 46k (benchmark) and 94k (panel b); shading marks training beyond 50k updates. (b) "Lost" means once correct but wrong at iteration 16. Broadcast uses new shared feature vectors. The separate answer-injected control preserves every answer at every observed iteration.

In one paired example, both fixed and independent-Gaussian starts produce the correct grid at iteration 3, but only the Gaussian continuation remains correct at 4. Identical displayed grids can therefore have different next-step outcomes.

Supplying the correct grid once through the slow-state embedding, with fixed initial fast state and the original clues, preserves all 128 answers throughout. The answer is neither clamped nor reinserted; this tests finite preservation, not discovery or a latent fixed point.

The effect is checkpoint-dependent. At the selected 46k MLP benchmark checkpoint, fixed, independent and shared starts all solve nearly every puzzle. Across the series from 14k onward, Gaussian starts remain reliable while fixed-start success varies widely. The benchmark attention models also have effective fixed starts on the same 128 puzzles. Four independent and four shared Gaussian draws remain highly accurate, with no loss after success under any fixed or Gaussian condition; answer-injected trajectories remain correct throughout. The severe late MLP failure thus does not characterize access in the benchmark models.

The late MLP contrast is consistent with training-state exposure: its reliable starts resemble those used in training. With multiple starts, a further decision arises: which answer should be returned?

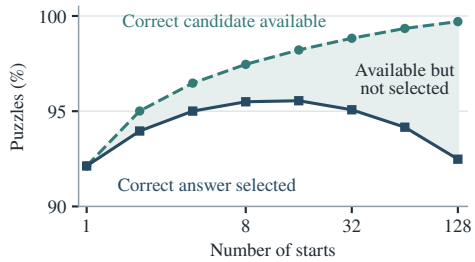
6 Exploration and Answer Selection

Each candidate is a noiseless $D = 64$ endpoint from independent standard-normal entries in both states, with EMA weights and clues fixed. Pools contain the first k candidates of an ordered bank: expansion leaves earlier trajectories and scores unchanged. Fixed starts and intermediate predictions are excluded; k trajectories cost kD outer iterations per puzzle.

Coverage is the fraction of puzzles with a reference-correct candidate; **selected accuracy** is the fraction returned correctly. We select the smallest residual: mean absolute change across both states and the final three iterations, rounded to float16, with ties favoring the earliest candidate. This tests whether learned dynamics supply a useful ranking signal; small movement does not certify correctness. EqR already links residual selection to learned alignment with correctness (Huang et al., 2026); Probabilistic TRM (PTRM) distinguishes coverage from accuracy under a learned head (Sghaier et al., 2026).

For our TRM implementation (seed 0, 50k), increasing starts from 8 to 128 on 20,000 uniformly sampled test puzzles raises coverage but lowers selected accuracy (Figure 3a): the 1,205 losses outnumber the 602 gains. With earlier candidates and scores fixed, every loss arises from a new wrong candidate winning with strictly smaller residual, while **the earlier correct candidate remains available**. At 128 starts, only 59 puzzles lack a correct candidate, but 1,445 available answers are missed. This isolates ranking failure from losing an answer during continued computation.

a Our TRM: coverage and selection



b Attention: restart performance

Model	k	Selected (%)	Coverage (%)
Our 5,000-puzzle test subset			
Attention 128	128	**99.96**	99.96
Attention 192	128	**99.98**	100.00
Attention 256	128	**99.92**	99.98
Published: different protocols			
EqR	128	99.8	-
PTRM	100	98.75	99.06

Figure 3: Candidate availability and reliable selection are different requirements. All trajectories use 64 iterations. (a) Our TRM (seed 0, 50k), 20,000 puzzles, nested Gaussian starts. The shaded gap shows covered puzzles returned incorrectly. From 8 to 128 starts, selection gains 602 successes and loses 1,205; each loss selects a new wrong candidate with strictly lower residual while correct candidates remain available. (b) Benchmark attention models: 5,000 common test puzzles, one training seed per model, Gaussian starts and residual selection (Section 6). k: candidate count; coverage: correct candidate available; dash: unreported. EqR (Table 4) and PTRM (Section 5.3) retain published populations and protocols; PTRM uses stochastic rollouts and Q-head selection. Equal iteration counts do not imply equal computational cost.

6.1 Training, candidate generation and selection

The selector has no fitted parameters, but the learned update produces both answers and the state changes used to rank them. Our TRM baseline trains without random starts or answer anchors. On the same puzzles at 128 starts, the randomized-start checkpoint covers fewer puzzles than baseline but returns more correct answers. It misses only four available solutions: better selection more than compensates for lower coverage. Conversely, the anchor-only variant generates solutions for almost every puzzle but returns a wrong candidate for most. **Useful generation need not yield reliable ranking**.

Baseline seed 1 covers almost the same number of puzzles as seed 0 but returns 7,768 fewer correctly. Its correct candidates rank slightly above incorrect ones on average, yet minimum-residual selection falls below the expected accuracy of uniform random choice. A wrong candidate can win with an extreme score despite this average advantage.

In the earlier Attention 256 ablation, jointly removing randomized starts and anchors slightly increases coverage but increases missed available answers from 1 to 69 at 32 starts on 5,000 uniformly sampled test puzzles (seed 0). By contrast, the benchmark attention models miss at most 3 covered answers each at 128 starts on this sample (Figure 3b). All three achieve **99.92–99.98% selected accuracy** on these 5,000 puzzles. Across the examined Sudoku checkpoints, Gaussian-trained models miss at most 4 available answers per bank; large selection gaps occur only without Gaussian training starts. Together with the initialization results, this links inference reliability to training-state exposure, consistent with EqR's training comparisons (Huang et al., 2026).

For unique-solution Sudoku, checking givens, rows, columns and boxes identifies correct candidates without reference answers; returning one would realize coverage. Verifier and reference counts agree per puzzle. Neither verification nor residual selection recovers answers absent from the pool; ARC tests this limit when transformations must be inferred from examples.

7 Generalization on ARC-AGI

ARC transformations vary by task. We evaluate released alphaXiv TRM on 419 queries from 400 ARC-AGI-1 tasks without augmentation, early halting or voting. It uses learned task embeddings and was trained on evaluation-task demonstrations; query answers are held out, tasks are not. At $D = 16$, only about a third of queries succeed where every demonstration is reproduced correctly: **demonstration fit does not ensure query success** (Figure 4a).

In fixed-start trajectories through $D = 64$, discovery stops at iteration 3 despite continued changes to incorrect grids, consistent with reported early saturation (Roye-Azar et al., 2025). Of 138 queries reached correctly, 10 end incorrect (Figure 4b).

All 16 fixed-start readouts plus eight Gaussian-start endpoints at $D = 16$ cover 146/419 queries: most remain **without a correct candidate to select** (Figure 4c). With selection by the smallest final slow-state change, expanding from one to eight Gaussian starts slightly lowers accuracy here but improves it without losses in a separate 40-query bank at $D = 64$, which also leaves most queries

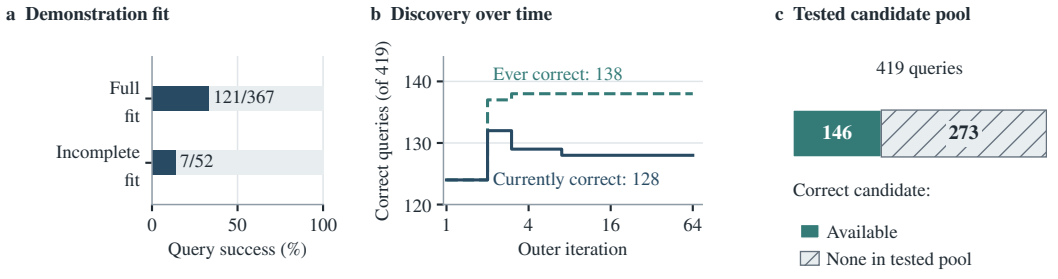


Figure 4: Demonstration fit and further computation leave an ARC coverage gap. Released TRM, 419 queries/400 tasks. (a) Query accuracy by fit to training demonstrations (1,295/1,363 correct). (b) No first success after iteration 3, despite 47 wrong grids changing after 16. (c) All 16 fixed-start readouts plus eight Gaussian-start endpoints at depth 16: 144 outer iterations per query. Reference-scored coverage concerns this finite pool, not other initializations, conditioning or budgets.

uncovered (Appendix A.4). If an inadequate task representation limits correction, revising it from demonstrations alongside the answer could add missing correct candidates.

8 Related Work

Recursive reasoning builds on relational message passing (Palm et al., 2018), hierarchical recurrence (Wang et al., 2025) and shared-network updates (Jolicoeur-Martineau, 2025). Feature perturbations (Bansal et al., 2022) and separate state/memory resets (Drozdova et al., 2026) precede our interventions. Path independence (Anil et al., 2022), reachability (Huang et al., 2026) and basin geometry (Lai et al., 2026) motivate initialization studies. Ren & Liu (2026) interpret abrupt completion through trapping near incorrect states. Attractor search (Balwani & Efstathiou, 2026) can coexist with the active correction and state dependencies measured here. Residual alignment and coverage–selection gaps (Huang et al., 2026; Sghaier et al., 2026), and sampling limits under imperfect selection (Brown et al., 2024; Stroebl et al., 2024; Gao et al., 2023), motivate our selection analysis. Our comparisons hold error count and encoded distance, the first recurrent state, or earlier candidates and scores fixed to distinguish repair, continued state use and selection failure (Appendix A.5).

9 Discussion and Conclusion

We asked what successive recursive iterations accomplish and when additional computation improves the returned answer. The Sudoku results reveal learned correction dynamics: substantial, offsetting revisions can precede abrupt completion, and error configuration, communication and recurrent state shape repair. Initialization can expose solving capability at fixed weights; training comparisons distinguish useful generation from reliable selection. Our compact attention solvers combine strong full-test accuracy with direct tests of repair and state dependence.

The repair and state interventions, together with the initialization contrasts, motivate a **learned relaxation with memory** hypothesis: retained or reconstructed context may support repeated revision of coupled representations, while evolving messages may propagate useful changes. Errors that reinforce one another internally may resist isolated correction. Tests could vary error relationships at matched error counts across controlled training exposures, or compare score-preserving state edits at matched perturbation sizes (Appendix A.6 gives a renormalization-group reference).

On ARC, the tested searches leave most queries without correct candidates, motivating a separate test of revising task representations from demonstrations alongside answers. On tasks excluded from parameter training, this could be compared with further search under unchanged conditioning, matching evidence and computation while measuring coverage, preservation and selection.

More broadly, our results sharpen the question of whether more recursion helps: which states can the learned update repair, which initializations expose that capability, when are correct answers preserved or recovered, and does selection return them when available? These experimentally grounded distinctions provide measurable targets for improving the learned update, training, initialization, recurrent state and return rule of recursive reasoning systems.

Limitations

The interventions identify dependencies, not message or memory contents, an energy function, latent fixed points or coordinated repair. The source of the measured accuracy gains over SE-RRM remains unresolved because representation, update design and training differ together; no common-budget ranking against FRM is available. One training seed per attention size, small or stratified diagnostics, near-ceiling outcomes and rare selection losses limit effect-size and prevalence estimates. Some analyses followed result inspection, and test outcomes informed experimental choices. The late MLP initialization contrast is numerically sensitive.

REPRODUCIBILITY STATEMENT

Appendices A.1 and A.2 specify the definitions, model configurations, training and evaluation procedures, randomization and compute conventions; the remaining appendices detail the experiments and their scope. The anonymized code supplement includes six selected checkpoints, pinned software requirements, diagnostic inputs, per-puzzle observations and scripts for regenerating the reported summaries. Its `README.md` distinguishes saved-record recounting, fresh inference and retraining. To meet the upload limit, the supplement retains decoded attention trajectories and residuals rather than the full logits, whose decoding was checked before export. A manifest identifies retained files and the hashes of omitted source arrays; the full archive is retained for release after review. Intermediate MLP checkpoints and the dependencies for fresh ConceptARC task-vector fitting are not bundled. Recounts use the saved precision and tie rules; fresh runs may differ with backend, precision, batching and random draws.

AI USE STATEMENT

We used generative AI assistants, including OpenAI Codex and Anthropic Claude, to aid in writing and polishing, locating relevant literature, refining author-developed hypotheses and providing feedback on experiments and interpretations, writing and debugging analysis code, and executing computational workflows. The authors led the research ideation, hypothesis development and experimental design. AI-assisted outputs were checked against source literature, implementations and recorded observations; quantitative results were verified through saved-record recounts and automated tests. The authors reviewed the manuscript and take responsibility for its content.

REFERENCES

Ekin Akyürek, Mehul Damani, Adam Zweiger, Linlu Qiu, Han Guo, Jyothish Pari, Yoon Kim, and Jacob Andreas. The surprising effectiveness of test-time training for few-shot learning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 942–963. PMLR, 2025.

Cem Anil, Ashwini Pokle, Kaiqu Liang, Johannes Treutlein, Yuhuai Wu, Shaojie Bai, J. Zico Kolter, and Roger B. Grosse. Path independent equilibrium models can better exploit test-time computation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi: 10.52202/068431-0566.

Arip Asadulaev, Rayan Banerjee, Fakhri Karray, and Martin Takac. Latent reasoning in TRMs is secretly a policy improvement operator. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2511.16886v5.

Darpan Aswal, Thomas Palmeira Ferraz, Yongxin Zhou, and Maxime Peyrard. Observable patterns are not explanations: A causal-geometric analysis of latent reasoning models. *arXiv preprint arXiv:2606.12689v1*, 2026.

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

Aishwarya Balwani and Andreas Efstathiou. Recursive reasoning as attractor landscape search: Mechanistic dynamics of the tiny recursive model. In *ICLR Workshop on Latent & Implicit Thinking*, 2026. OpenReview.

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, 2022. doi: 10.52202/068431-1471.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787v3*, 2024.

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

Liam Davis, Leopold Haller, Alberto Alfarano, and Mark Santolucito. Lattice deduction transformers. *arXiv preprint arXiv:2605.08605*, 2026.

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

Joshua V. Dillon. Speed is confidence. *arXiv preprint arXiv:2601.19085*, 2026.

Mariia Drozdova, Aidan Sirbu, Pietro Miotti, Robert Obryk, Mayalen Etcheverry, Eyvind Niklasson, and Blake Richards. Diffusion as a training curriculum for timestep-free iterative reasoning. *arXiv preprint arXiv:2609.01449*, 2026.

Richard Freinschlag, Timo Bertram, Erich Kobler, Andreas Mayr, and Günter Klambauer. Symbol-equivariant recurrent reasoning models. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2603.02193v1.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10835–10866. PMLR, 2023.

Alec Helbling, Andrey Bryutkin, Mauro Martino, Duen Horng Chau, Nima Dehmamy, and Hendrik Strobelt. Flow reasoning models: Turning flows into efficient recurrent reasoners. *arXiv preprint arXiv:2606.29150*, 2026.

Benhao Huang, Zhengyang Geng, and J. Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2605.21488.

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

Jeffrey Lai, Anthony Bao, John Quinn, and William Gilpin. Fractal basins trap latent reasoning. *arXiv preprint arXiv:2609.04963*, 2026.

Chao Ma, Jianchun Wang, and Weinan E. Model reduction with memory and the machine learning of dynamical systems. *Communications in Computational Physics*, 25(4):947–962, 2019. doi: 10.4208/cicp.OA-2018-0269.

Yosuke Miyanishi and Tetsuro Morimura. Interaction locality in hierarchical recursive reasoning. *arXiv preprint arXiv:2605.20784v1*, 2026.

Arseny Moskvichev, Victor Vikram Odouard, and Melanie Mitchell. The ConceptARC benchmark: Evaluating understanding and generalization in the ARC domain. *Transactions on Machine Learning Research*, 2023.

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.

Rasmus Berg Palm, Ulrich Paquet, and Ole Winther. Recurrent relational networks. In *Advances in Neural Information Processing Systems*, volume 31, pp. 3368–3378, 2018.

Lucas Prieto, Melih Barsbey, Pedro A. M. Mediano, and Tolga Birdal. Grokking at the edge of numerical stability. In *International Conference on Learning Representations*, 2025. arXiv:2501.04697.

Kaleem Ullah Qasim and Jiashu Zhang. Accelerating training speed of tiny recursive models with curriculum guided adaptive recursion. *arXiv preprint arXiv:2511.08653*, 2025.

Zirui Ren and Ziming Liu. Are your reasoning models reasoning or guessing? A mechanistic analysis of hierarchical reasoning models. *arXiv preprint arXiv:2601.10679*, 2026.

Leo Raphael Rodrigues and Jian Kang. Dissecting hierarchical reasoning models: A mechanistic study. *arXiv preprint arXiv:2609.22197v1*, 2026.

Antonio Roye-Azar, Santiago Vargas-Naranjo, Dhruv Ghai, Nithin Balamurugan, and Rayan Amir. Tiny recursive models on ARC-AGI-1: Inductive biases, identity conditioning, and test-time compute. *arXiv preprint arXiv:2512.11847*, 2025.

Amin Sghaier, Ali Parviz, and Alexia Jolicoeur-Martineau. Probabilistic tiny recursive model. *arXiv preprint arXiv:2605.19943*, 2026.

Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. Inference scaling fLaws: The limits of LLM resampling with imperfect verifiers. *arXiv preprint arXiv:2411.17501v2*, 2024.

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.

Kenneth G. Wilson. The renormalization group: Critical phenomena and the Kondo problem. *Reviews of Modern Physics*, 47(4):773–840, 1975. doi: 10.1103/RevModPhys.47.773.

A Appendix

A.1 Notation, Terminology and Definitions

We distinguish recurrent state, decoded prediction and returned answer. Notation follows Equation 1; state shapes and update schedules accompany each experiment.

Notation	Meaning
$x,\ y^\star,\ \theta$	Input, reference answer used for scoring, and fixed model parameters.
$I(x;\xi),\ e_\theta(x)$	Initialization, with random draw ξ when applicable, and the retained input embedding.
$F_\theta,\ R_\theta$	One outer update and the readout producing output scores.
$s_t,\ \widehat{y}_t$	Internal state and decoded prediction after outer update t. In our Sudoku models, $s_t = (h_t, l_t)$ contains the slow and fast states.
$w,\ D$	Hidden dimension (feature channels) and recursion depth (number of outer updates).

A *trajectory* follows one initialization through successive updates; reported predictions begin at $t = 1$. Slow and fast name update frequencies, not semantic roles. A *carry* is state retained between outer iterations. An *outer update* is one such iteration, distinct from the elementary fast- or slow-state updates within it. Depth counts outer iterations; training steps count optimizer updates. Outer updates contain model-specific inner computation, so equal depths need not imply equal arithmetic cost.

Representation and communication. A *digit field* holds one candidate digit's features across the Sudoku grid. Spatial mixing exchanges information across cells within a field; *cross-field communication* exchanges information between candidate digits at a cell. Message removal targets this latter branch in our models, not all information exchange. Message replay reuses first-iteration messages at the same network locations in later iterations, without freezing recurrent states. Released cell-token models also mix information across cells but lack this branch. The *readout* maps state to output scores; *decoding* converts scores to a predicted grid. The return rule chooses which prediction to output.

Correctness and progress. For puzzle or query i, let $c_i(t) = \mathbf{1}[\widehat{y}_{i,t} = y_i^\star]$ indicate exact correctness. Sudoku exactness requires all 81 cells, including clues, to match; ARC exactness requires both output dimensions and cell values to match. For Sudoku, let U_i contain the originally empty cells. The cellwise error fraction is

$$e_i(t) = \frac{1}{|U_i|} \sum_{j \in U_i} \mathbf{1}[\widehat{y}_{i,t,j} \neq y^\star_{i,j}].$$

Reported mean error fractions average these per-puzzle quantities. *Repair* means correction of wrong assignments: error reduction measures partial repair, while exact repair requires the complete solution at the stated observation. *Matched-error* grids share a puzzle, clues and wrong-cell count, but can differ in error locations and digit values. A *corruption draw* is one grid sampled by a specified perturbation procedure; training anchors and matched-error tests use different procedures. First completion is $\tau_i = \min\{t \in \{1, \ldots, D\} : c_i(t) = 1\}$; if no such update is observed, completion time is right-censored at D. First completion records an event; it does not impose an early-stopping rule.

Discovery, loss and preservation. A trajectory is *ever correct* if any observed prediction is exact, and *endpoint correct* if $c_i(D) = 1$. An observed correctness loss is a transition from $c_i(t) = 1$ to $c_i(t+1) = 0$. A loss is temporary when correctness is later recovered; a *terminal loss* means the trajectory was ever correct but ends incorrect. An any-loss count includes each trajectory only once. Natural answer persistence means remaining correct at every observed update after first completion. In a *supplied-answer retention* assay, the known answer is encoded once into the specified state; retention requires every subsequent recorded prediction to remain exact. The answer is neither clamped nor reinserted. This tests preservation under that encoding and continuation, without testing discovery.

Paired comparisons and access. Paired gains and losses compare outcomes for the same puzzles under two conditions or budgets. A gain changes incorrect to correct; a loss reverses it. These exclude intermediate correctness transitions; equal accuracies can conceal paired gains and losses. Success from a tested answer-free start demonstrates *conditional solving capability* under the stated budget and return rule. *Accessibility* concerns whether a specified initialization exposes that capability with the same weights and subsequent computation.

Candidates and selection. A candidate pool contains the endpoints, intermediate readouts or both specified by the collection rule. *Coverage* is the fraction of puzzles with at least one reference-correct candidate; *selected accuracy* is the fraction whose returned candidate is correct. An uncovered puzzle has no correct answer in that finite pool. A *mixed pool* contains both correct and incorrect candidates. A *residual* measures a specified state discrepancy, either between successive states or between a state and its proposed update. Its components, norm and averaging window depend on the protocol. A small residual does not certify correctness or the latent fixed-point equality $F_\theta(s; e_\theta(x)) = s$. Unchanged decoded answers over a finite horizon do not establish this equality either.

A.2 Model, Training and Experimental Details

We specify the shared architecture and training procedure of our benchmark Sudoku solvers, then distinguish their evaluation protocols from those of released comparison models. The Sudoku implementation uses JAX random keys for training and fixed buffers, and NumPy's `default_rng` seed sequences for diagnostic starts and corruptions. The pinned reproduction environment uses JAX 0.10.2 and NumPy 2.5.1; these versions do not certify every historical execution environment.

A.2.1 Architecture and Digit Symmetry

Our models combine TRM's shared-network recurrence with SE-RRM's symbol-indexed representation and attention, and EqR-inspired training choices (Jolicoeur-Martineau, 2025; Freinschlag et al., 2026; Huang et al., 2026). For digit d and cell i, the input is $e_\theta(x)[d, i, :] = \sqrt{w}\, r_{\rho(d,x_i)}$, where the three learned vectors r encode blank, matching-digit and other-digit clues. Both recurrent arrays have shape $9 \times 81 \times w$. The fixed initial state broadcasts two nonlearned vectors across fields and cells, sampled once from standard normal distributions truncated to $[-2, 2]$, using seed 20260902. Random-start evaluations instead specify full state tensors. Table 2 identifies the benchmark weights.

Table 2: Selected EMA checkpoints. Parameter counts include auxiliary parameters. All use the same outer schedule; changing mixer family also changes cross-field mixing and parameterization, so the comparison does not isolate attention alone.

Model	Parameters	Training seeds	Selected update
MLP 192	789,125	0, 1, 2	46,000 each
Attention 128	623,365	0	42,000
Attention 192	1,057,925	0	40,000
Attention 256	1,967,621	0	46,000

Each of the two blocks in B_θ applies spatial mixing, cross-field mixing and a channel MLP, in that order. Spatial mixing connects all 81 positions within a field, without a prescribed Sudoku constraint graph. The MLP variant uses a shared SwiGLU network over positions. The attention variant uses $w/32$ heads of dimension 32, with two-dimensional rotary positions for rows and columns. Cross-field mixing acts at each cell. Writing $r_{d,i}$ for features after spatial mixing, the MLP message is $m_{d,i} = W_C\big(\frac{1}{8}\sum_{e\neq d} r_{e,i}\big)$. For one attention head,

$$m_{d,i} = \sum_{e\neq d} \alpha_{d,e,i} W_V r_{e,i}, \qquad \alpha_{d,e,i} = \frac{\exp(q_{d,i}^\top k_{e,i}/\sqrt{32})}{\sum_{f\neq d}\exp(q_{d,i}^\top k_{f,i}/\sqrt{32})}.$$

Queries q and keys k are learned linear projections. The four cross-field heads use query/key dimension 32 and value dimension $w/4$; their outputs are concatenated without an additional output projection. All projections are shared across digit labels; self-field attention is masked. The mean message also depends on the current features; attention additionally makes its mixing coefficients state-dependent.

Each operation has a residual addition followed by root-mean-square normalization, with numerical constant 10^{-5}. The position MLP normalizes over positions; the other branches normalize over features. SwiGLU intermediate dimensions use $8/3$ times the input dimension, rounded up to a multiple of 256 after integer rounding: 256 for the position MLP, and 512/512/768 for channel dimensions 128/192/256. Each block has its own parameters, shared across states and updates. The elementary update and inner cycle are

$$U_\theta(z, u) = 0.05z + 0.95B_\theta(z + u),$$
$$l \leftarrow U_\theta(l, h + e_\theta(x)) \quad \text{six times}, \qquad h \leftarrow U_\theta(h, l) \quad \text{once}.$$

The slow state stays fixed during the six fast updates; the puzzle embedding enters these fast updates directly. Three cycles give 21 applications of B_θ per outer iteration, with no additional damping between iterations.

Attention 128's 623,365 parameters comprise channel MLPs (393,216), spatial-attention projections (131,072), cross-field projections (98,304), and embedding, readouts and auxiliary scalars (773), counted from the stored network tensors without the unused task table. Its two carries contain $2 \times 9 \times 81 \times 128 = 186{,}624$ activations per puzzle; these are not learned parameters and account for only part of runtime memory. At $D = 64$, the same two-block stack executes $21 \times 64 = 1{,}344$ times. Its hidden dimension is half SE-RRM RoPE2D's 256; at dimension 256, our 1,967,621 parameters are close to its 1,970,176. The smaller dimension explains the principal parameter reduction; symbol sharing and recurrent weight reuse are inherited principles, not isolated explanations of the accuracy difference.

For a digit permutation π and the corresponding field permutation P_π,

$$e_\theta(\pi x) = P_\pi e_\theta(x), \quad F_\theta(P_\pi s; P_\pi e) = P_\pi F_\theta(s; e), \quad R_\theta(P_\pi s) = P_\pi R_\theta(s).$$

The last identity concerns digit scores. Shared spatial/channel operations, symmetric cross-field aggregation and the shared readout preserve these identities in real arithmetic. Fixed starts are invariant across fields; for samplewise equivariance, random states and any update noise must be permuted with the input. Exchangeable draws instead give symmetry in distribution. This is digit symmetry, not arbitrary spatial symmetry; floating-point reductions and tied-score argmax need not preserve decoded trajectories exactly.

The digit readout is $a_{t,i,d} = v_\theta^\top h_t[d, i, :]$. Blank and padding scores are fixed at $-10{,}000$, and benchmark argmax includes all 11 tokens. No constraint solver, clue overwriting or predicted-clue feedback is applied. The earlier Attention 256 recipe study uses a 30k budget and a 28k reference; MLP 384 is selected at 16k. The 38-checkpoint MLP 192 initialization series extends the seed-0 run to 150k; its highlighted 94k checkpoint is separate from the selected benchmark weights.

A.2.2 Training states, supervision and optimization

The 1,000 base training puzzles each contribute their original view and 1,000 precomputed positional augmentations. Transformations apply equally to puzzle and solution: transpose, row-band and column-stack permutations, and permutations within bands and stacks. These views need not all be distinct. Digit labels are not augmented in the benchmark models.

Each persistent batch slot retains a puzzle and two state arrays. A fresh puzzle starts from independent standard-normal entries in both arrays. One outer iteration produces a supervised prediction and is followed by an optimizer update. A continuing slot reuses its state with the updated parameters; a halted slot receives a new sampled puzzle at the next step. Training thus carries states produced by earlier parameter values, whereas inference holds parameters fixed. Following TRM and EqR, gradients stop between outer iterations and after each of the first two inner cycles. All 21 stack applications execute, but the recurrent gradient passes only through the final seven, together with their input embedding and readout.

Training adds independent Gaussian noise after each elementary update:

$$U_\theta^{\text{train}}(z, u; \epsilon) = 0.05z + 0.95B_\theta(z + u) + 0.01\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I).$$

Evaluation omits this noise. Output probabilities use StableMax (Prieto et al., 2025), independently of softmax inside attention:

$$p_c = \frac{\phi(a_c)}{\sum_{v \in \mathcal{V}} \phi(a_v)}, \qquad \phi(a) = \begin{cases} 1 + a, & a \geq 0, \\ (1 - a)^{-1}, & a < 0. \end{cases}$$

Here $\mathcal{V}$ contains all 11 tokens. Even the two fixed negative scores receive small positive mass. For batch size B, the main prediction loss is

$$\mathcal{L}_{\text{main}} = -\frac{1}{81B} \sum_{b=1}^{B} \sum_{i=1}^{81} \log p_{b,i,y^\star_{b,i}}.$$

Every segment targets the complete solution, including givens, without intermediate deduction traces. Probability analyses must retain their stated normalization: digit-conditional StableMax and full-vocabulary StableMax differ, as does replacing either with softmax.

The halting head reads the mean slow-state feature across fields and cells. Its first output q_b is trained by binary cross-entropy with logits against $\mathbf{1}[\widehat{y}_b = y_b^\star]$, without differentiating through this label; the second output is unused by the loss. A positive q_b normally halts the training trajectory. At each decision, the minimum allowed iteration is redrawn: with probability 0.1 it is uniform from 2 through 16; otherwise it is zero. The 16-iteration cap always applies. Reference correctness supplies the training label, not the halt decision.

The answer-anchor branch uses the first quarter of each device's batch. For each example, draw $p \sim \mathrm{Uniform}[0, 0.2)$ and independently attempt to replace each empty cell's solution digit with probability p. Replacement digits are uniform in $\{1, \ldots, 9\}$, so they may be unchanged: each empty cell is wrong with probability $8p/9$. Givens are preserved. The corrupted grid initializes the slow state through the role embedding; the fast state uses its fixed vector, and the original puzzle remains the input. One noisy outer iteration receives the same all-cell prediction loss. This branch shares the main gradient truncation, supplies no halting loss and does not feed its state into the main carry. With each loss averaged over its own examples,

$$\mathcal{L} = \mathcal{L}_{\mathrm{main}} + 0.5\,\mathcal{L}_{\mathrm{halt}} + \mathcal{L}_{\mathrm{anchor}}.$$

The quarter-batch allocation does not multiply the anchor loss by 0.25. There is no latent-stationarity penalty, residual-ranking objective or requirement that every update reduce decoded errors.

Table 3: Shared settings for the six benchmark runs. Training uses current parameters; evaluation selects EMA weights. The implementation uses AdamW rather than released TRM's AdamATan2.

Setting	Benchmark training recipe
Optimizer	AdamW; global gradient-norm clipping at 1.0 after averaging across devices
Batch and budget	768 persistent slots; 50,000 optimizer updates
Learning rate	Linear warmup for 2,000 updates to 10^{-4}, then constant
Weight decay; moments	Weight decay 1.0; $(\beta_1, \beta_2) = (0.9, 0.95)$; $\epsilon = 10^{-8}$ outside the square root
Parameter averaging	EMA decay 0.999, updated after each optimizer step
Trajectory and anchors	At most 16 outer updates; one anchor update on one quarter of slots

Attention training was extended from 30k to 50k updates. Resumption restores parameters, optimizer, EMA and random-generator state but resets the active example carries; uninterrupted retraining is therefore a replication of the recipe, not a bitwise replay. Earlier recipe comparisons change starts/anchors, damping/noise, or batch size/learning rate jointly. Their selected ages and individual settings accompany the comparisons in Appendix A.3.6 and Appendix A.3.5.

A.2.3 Data, evaluation and reproducibility

Splits and selection. We use the released Sudoku-Extreme corpus.[1] Preparation samples 1,064 source training-file rows without replacement at seed zero, sorts their indices, and assigns the first 1,000 to training and the remaining 64 to validation. An extension adds 448 disjoint rows using seed 20260908, preserving training and test arrays. All six benchmark runs share this split and the full 422,786-puzzle test file. Training seeds vary initialization and augmentation while preserving the base split. An exact search finds no equivalent puzzle pair across the training, validation and test splits under digit relabeling, within-band row permutations, band permutations, corresponding column/stack permutations and transpose. Invariant signatures filter pairs before exact group-equivalence checks.

Checkpoints are ranked by depth-16 EMA accuracy on the 512-puzzle validation set, then non-EMA accuracy at the same depth, then earliest step. The selected weights are used at both benchmark depths. This within-run rule does not remove the influence of test outcomes on broader experimental choices. Table 4 separates the principal diagnostic samples. For stratified sampling, we rank the corpus difficulty ratings into eight nearly equal-sized groups and draw equally from each.

The initialization pool samples source training rows at seed 20260914, excluding training and validation-monitor rows. It sorts the indices and checks exact-string nonoverlap with training, validation and test grids. The MLP study uses these same 128 cases at 38 checkpoints from 2k to 150k, with training resumed after 50k. The uniform trajectory sample uses seed 20260919; uniform restart and extended-depth subsets use seed 20260822. Diagnostic conditions pair puzzles and weights; answer-derived starts are repair/preservation assays, not answer-free benchmark runs.

[1] https://huggingface.co/datasets/sapientinc/sudoku-extreme

Table 4: Distinct diagnostic populations. Saved puzzle identities define pairing; equal sample sizes or sampling seeds do not imply identical or nested sets. The full-test benchmark is reported separately.

Analysis	Population and budget
Outer-update trajectories	512 uniformly sampled test puzzles, $D = 32$; earlier attention checkpoint at 28k and released Sudoku ports included
Extended attention depth	50,000 uniformly sampled test puzzles at $D = 128, 256$
Matched-error repair	512 difficulty-stratified test puzzles; 438 source-error pairs; $D = 16$
Communication and resets	256 alternating rows of the repair set for communication and final-attention resets; a separate, disjoint stratified 256-puzzle set for the original four-model reset study
Initialization	First 128 stored rows of a separate 10,000-puzzle validation pool; $D = 16$
Recipe and restart studies	20,000 uniform test puzzles for TRM recipes; 5,000 for attention recipes and final models; restart depth $D = 64$

Evaluation and numerical conventions. Benchmark inference uses the original input orientation and returns one noiseless fixed-start endpoint, without voting or early halting. Exactness and empty-cell error follow Appendix A.1. Public TRM, Curriculum-Guided Adaptive Recursion (CGAR; Qasim & Zhang, 2025) and EqR Sudoku weights use common JAX implementations. On 64-puzzle, depth-16 conversion checks, endpoint correctness differs from native execution on 3, 2 and 2 puzzles, respectively. EqR's common-evaluator traces omit update noise. Our separately trained TRM uses two MLP-mixer blocks and two 97×512 state arrays: 81 cell tokens plus 16 learned prefix positions. It follows the same three-cycle, six-fast/one-slow schedule as our benchmark solvers, with digit-specific token embeddings rather than digit fields. It is distinct from released TRM and the smaller TRM-cell control in Appendix A.3.6.

The original reset and MLP initialization studies use CPU execution; the original resets decode digits 1–9 only, with their own intact controls. For 37 of 38 MLP checkpoints with paired CPU/accelerator records, endpoint correctness agrees on 98–126 of 128 puzzles, including 122 at 94k. Alternative starts have no corresponding cross-device validation. The benchmark attention diagnostics use float32 CPU arithmetic, batch 128, full-vocabulary decoding and $D = 16$, without noise, clue clamping, halting or parameter adaptation. Settings were specified before inference. Each model reproduces earlier 128-puzzle predictions and digit logits bitwise; Attention 128 also reproduces its earlier communication control. These execution checks are neither training replicates nor full-test cross-device comparisons.

The four original attention corruption draws were saved before inference and are shared across receivers. Draw zero defines the declared primary endpoint contrast; draws 1–3 assess sampling sensitivity. Four-draw means summarize that sensitivity, without replacing the primary contrast. Communication and reset continuations share each checkpoint's complete first state. Message removal zeros both blocks' cross-field value projections; resets restore the chosen state before each iteration 2–16, retaining all within-iteration computation. Shared Gaussian draw j broadcasts the first puzzle's independent draw j across puzzles; these conditions are paired, not independent replicate sets. All four draws are reported without answer-based selection. Answer injection embeds the solution in the slow state and uses the fixed fast state, without replacing the original clues.

The clue-compatible and structured corruption controls use four further draws per family, shared across receivers, with four-draw summaries specified before inference. Message replay uses batch 64 rather than 128. Validation checks all 256 complete first carries and prediction feedback, including first-iteration replay, bitwise against the unmodified block. On the first 64 puzzles, recording without replay reproduces every iteration's logits through 16 bitwise against unmodified batch-64 execution and the archived batch-128 control.

ARC protocols remain separate. Public TRM uses its native recurrence in float32 on 419 queries from 400 tasks, with learned task identifiers and prior exposure to evaluation-task demonstrations. Query outputs are held out. FPRM is evaluated under a reconstructed task-ID mapping; its agreement with the historical training assignment remains unverified. The auxiliary ConceptARC models A and B are time-conditioned answer-canvas checkpoints, each trained for 53,333 steps at depth six, with a 32-dimensional task vector. They use a different state encoding and support-only task-vector fitting (Appendix A.4). ARC solver steps are not interchangeable with Sudoku outer iterations.

Uncertainty and example selection. Full-test fractions describe the fixed corpus. Puzzle-bootstrap intervals condition on trained weights and, for repair, saved corruption draws; they do not estimate training-seed variation. Attention repair uses 10,000 paired resamples within eight difficulty strata;

four-draw averages keep each puzzle's outcomes together. Intervention intervals use unstratified paired puzzle resampling. ARC query resampling must preserve task membership. Timing decompositions, revision counts, ranking summaries and clue-conflict analyses were specified after examining results.

The Section 3.1 example uses MLP 192 at 46k: among puzzles first solved at updates 3–32, select the median completion time, then the smallest puzzle ID. For the paired example in Section 5.1, select cases where both starts are correct at $t < 16$ but only the independent-Gaussian start is correct at $t + 1$; choose the smallest stored index among the 60 qualifying cases, then the earliest transition. This gives index 1 and updates 3 to 4. Both examples and the highlighted 94k checkpoint were selected after inspecting trajectories.

Arithmetic and reproduction. One scalar multiply-accumulate is one MAC. Counts include dense projections, attention score/value products (including computed masked entries), and each outer readout including the auxiliary head. They exclude nonlinearities, normalization, rotary arithmetic, softmax, other elementwise operations and memory traffic. For k trajectories of depth D and update cost c_θ GMACs, recurrent cost is $kDc_\theta/1000$ TMACs; selection and verification costs are separate. SE-RRM's count comes from its official Sudoku configuration and shape-checked implementation; its accuracy remains the published value. Our outer iteration executes 21 two-block stacks, versus 18 for SE-RRM; the MAC counts include these schedules and their different internal dimensions. Parameter sharing allows additional iterations to increase computation without adding weights, while equal outer depths can conceal different amounts of work per update. Reporting accuracy together with MACs accounts for model size, recurrence depth and restart count, allowing parameter economy and inference arithmetic efficiency to be assessed separately. Runtime additionally depends on hardware, precision, batch size, kernels, memory access and sequential dependencies. Parallel starts may reduce latency without reducing summed arithmetic. These operating-point comparisons neither measure elapsed runtime nor interpolate accuracy at equal arithmetic cost. Training cost also includes backward passes and the number of training updates and examples processed; inference MACs do not establish training efficiency.

Code and records. The supplement supplies six selected weights, configurations, split hashes and per-puzzle records, with separate recounting, inference and retraining procedures. Fresh inference depends on backend, precision, batching and random draws; recounts preserve stored precision and selection ties. Decoded records cover the original 138 attention chunks and added repair and state controls, with inputs and source hashes. Export checks independently decoded the original 282,624 attention predictions, 444,422 TRM/FPRM follow-up outputs and 1,640 readout-control outputs against the recorded answers, plus the added attention controls. Full logits are omitted to meet the upload limit; their hashes and decoded observations are retained. Intermediate MLP checkpoints remain unbundled. ConceptARC supports recounting saved flags; fresh task-vector fitting requires archived weights and task data.

A.3 Extended Sudoku Results and Controls

These results supplement Sections 3–6, using the model identities and protocols in Appendix A.2. Performance and timing are in Subsection A.3.1; repair and state interventions, Subsections A.3.2–A.3.3; initialization, Subsection A.3.4; and selection, Subsection A.3.5. Recipe comparisons in Subsection A.3.6 support Sections 3 and 6.

A.3.1 Performance and timing of correction

Depth and model comparisons. Table 5 separates the full-test depth sweep from its 50,000-puzzle extension. Attention 128 trails SE-RRM RoPE2D only at depths 2 and 4 among the listed full-test depths; the two wider models lead at every listed full-test depth. SE-RRM reports 98.84% at depth 128 with RoPE2D, and 95.4% at depth 16 with its separate RoPE variant (Freinschlag et al., 2026). On the common extended-depth subset, Attention 128 adds 164 successes from depth 64 to 128 and 66 from 128 to 256, despite twice as many additional updates. Attention 128 has the highest observed accuracy at depth 256 on this subset.

The three MLP 192 seeds each cost 14.67 GMACs per update, or 0.235/0.939 TMACs at depths 16/64. Full-test correct counts for seeds 0/1/2 are 405,477/400,900/403,707 at depth 16 (95.91/94.82/95.49%) and 419,215/417,784/419,306 at depth 64 (99.16/98.82/99.18%), out of 422,786. The depth-64 mean is 99.05%; attention has one training run per size.

Table 5: Endpoint performance at fixed selected weights. The upper block uses all test puzzles; SE-RRM values are published results (Freinschlag et al., 2026). The lower block follows one shared subset through 256 updates, including its own depth-64 prefix. Its depth-256 accuracies are 99.886/99.872/99.828%. Dashes indicate unavailable subset results. Our training trajectories are capped at 16 updates.

Depth	Attention 128	Attention 192	Attention 256	SE-RRM
Full test: 422,786 puzzles; endpoint accuracy (%)				
1	18.15	32.68	26.72	16.05
2	45.69	65.85	62.97	62.06
4	71.45	81.90	85.04	77.31
8	87.77	92.40	94.29	87.38
16	95.45	97.31	97.93	93.73
32	98.46	99.05	99.21	96.82
64	99.48	99.62	99.62	98.22
Common subset: 50,000 puzzles; correct endpoints (counts)				
64	49,713	49,789	49,805	–
128	49,877	49,894	49,882	–
256	49,943	49,936	49,914	–

On the full test at depth 64, EqR, CGAR and public alphaXiv TRM evaluated through the common implementation solve 393,835, 387,718 and 352,809 puzzles (93.15/91.71/83.45%). Against MLP seed 0's 419,215 successes, the paired MLP-only/public-only counts are 26,727/1,347, 32,745/1,248 and 67,338/932, respectively. These advantages of 6.00/7.45/15.71 percentage points carry the implementation and numerical qualifications in Appendix A.2.3. They neither replace native published scores nor establish comparisons at equal arithmetic cost. The TRM-cell and recipe controls below examine possible contributors.

Decoded progress and probability. The MLP example in Section 3.1 has negative log-likelihoods (NLLs) 1.2124, 0.9785, 0.4677 and 0.0034 nats at updates 1–4. Besides the reported repairs and regression, its first transition exchanges one wrong digit for another; the next two repair 18 and three cells without regressions. The grid remains exact through update 32. For digit logits a, the probability diagnostic uses

$$\widetilde{p}_d = \frac{\phi(a_d)}{\sum_{r=1}^{9} \phi(a_r)}, \qquad \mathrm{NLL}_i(t) = -\frac{1}{|U_i|} \sum_{j \in U_i} \log \widetilde{p}_{i,j,y^\star_{i,j}}(t),$$

with StableMax weights ϕ from Appendix A.2.2. This conditions on digits and averages empty cells, unlike the all-cell, full-vocabulary training objective. Stored float16 logits are promoted for this calculation; decoded errors use the original predictions. Released per-puzzle summaries permit recounting these NLLs; full logit tensors are not bundled.

Across all 512 MLP trajectories, exact counts increase from 82 to 504 between updates 1 and 32, mean empty-cell error falls from 39.26% to 0.79%, and mean NLL from 0.9103 to 0.0221. Continued change can also fail: among the eight unsolved cases, the smallest test ID, 22140, has 54 empty cells and 27/28/30/25/29/27 errors at updates 1/2/4/8/16/32. Its revisions do not produce sustained error reduction.

Completion-conditioned timing. First completions at updates 1, 2, 3–32 and no completion by 32 number 82/135/287/8 for MLP, 122/199/184/7 for the earlier Attention 256, 45/162/249/56 for EqR and 136/160/127/89 for public TRM. Each model partitions the same 512 inputs into completion cohorts. The MLP cohort's mean error falls from 54.31% initially to 37.50% immediately before success; its precompletion error quartiles are 29.82% and 46.49%. For each qualifying puzzle, decompose the reduction before completion as

$$e_i(1) - e_i(\tau_i - 1) = \big[e_i(\tau_i - 2) - e_i(\tau_i - 1)\big] + \big[e_i(1) - e_i(\tau_i - 2)\big].$$

This excludes the completing transition; the earlier term is zero for $\tau_i = 3$ (counts in Table 6). Concentration near completion also holds among the 49 MLP cases with $\tau_i \geq 9$: the last precompletion transition contributes 10.84 of the 16.50 points corrected before success. Longer completion-aligned offsets admit fewer histories, so their means cannot be added across changing denominators.

Earlier digit changes count every empty-cell revision through $\tau_i - 2$, normalized by each puzzle's empty-cell count and averaged over the original completion cohort, including zeros for $\tau_i = 3$.

Repeated changes count separately, without duration normalization. Wrong-to-right minus right-to-wrong events recovers the earlier net reduction; remaining changes exchange wrong digits. Table 6 distinguishes the uniform depth-32 corpus from final-attention traces on the 512 stratified repair puzzles at depth 16.

Table 6: Timing and revisions within each checkpoint's completion cohort. Upper block: mean net correction before completion, its last-precompletion and earlier components (percentage points), and median remaining error (%). Original rows use 512 uniform puzzles, $3 \leq \tau \leq 32$; benchmark rows use 512 stratified puzzles, $3 \leq \tau \leq 16$. Lower block: revision events per 100 empty cells through $\tau - 2$, allowing repeated changes; better/same/worse compares the last incorrect and first predictions. Each row conditions on its own completions; the two blocks use identical row-wise cohorts.

Checkpoint	n; $n_{\tau=3}$	Before	Last	Earlier	Remaining
MLP 192 (46k)	287; 86	16.82	13.02	3.80	39.66
EqR	249; 82	16.51	12.44	4.06	36.84
Public TRM	127; 43	18.99	16.00	2.99	31.03
Attention 256 (28k)	184; 57	7.52	7.48	0.04	45.61
Attention 128 (benchmark)	260; 78	9.08	8.89	0.19	43.64
Attention 192 (benchmark)	151; 39	5.23	5.06	0.16	47.27
Attention 256 (benchmark)	181; 83	7.88	6.33	1.55	44.83

Checkpoint	Digit changes	Wrong → right	Right → wrong	Better/same/worse
MLP 192 (46k)	73.11	26.94	23.14	245/11/31
EqR	72.11	27.43	23.37	220/10/19
Public TRM	102.54	38.66	35.67	107/0/20
Attention 256 (28k)	69.57	23.94	23.91	114/7/63
Attention 128 (benchmark)	59.32	20.41	20.22	196/21/43
Attention 192 (benchmark)	56.76	19.67	19.51	93/11/47
Attention 256 (benchmark)	45.24	16.30	14.75	121/8/52

Finite preservation. None of the four displayed uniform-sample checkpoints loses correctness after first success through depth 32. This does not contradict the rare full-test attention losses in Section 3.2: full-test any-loss counts for Attention 128/192/256 are 0/4/7. All these cases recover by depth 16, and all three attention models have zero terminal losses at depth 64. MLP seed-1 test ID 218508 is correct at update 2, incorrect at 3–24, and correct again at 25–64; its depth-16 and depth-64 correctness prefixes agree. MLP seed 0's full-test depth-64 archive contains first-hit and endpoint fields rather than every correctness observation, so it cannot exclude temporary losses. Later first completions also provide less follow-up.

Inside the completing cycle. We instrument separate fixed-start, noiseless executions of MLP 192 (46k), Attention 128 (42k) and ported EqR on 256 stratified test puzzles, with float32 arithmetic, float16 stored margins, batch 64 and digit-only decoding. Each of 16 iterations has three inner cycles. The respective cohorts contain 134/122/108 puzzles first exact at slow readout $\kappa \geq 7$ and exact at readout 48. For empty cells wrong throughout $\kappa - 4, \ldots, \kappa - 1$, let m_j be the within-puzzle median of the correct-digit score minus the best alternative's score. Median gap closure $[m_{\kappa-1} - m_{\kappa-4}]/[-m_{\kappa-4}]$ is 0.408/0.388/0.462: scores partly improve before completion.

After fast update k, a *shadow readout* applies the slow update early, $h' = U_\theta(h, l_k)$, without feeding it back. At $k = 6$ it equals the actual slow readout exactly. In the completing cycle, the median first exact shadow occurs at fast update 3.5/3/3.5; no shadow in the preceding cycle is exact. These are probes of answer availability through a slow update, not identified fast-state contents. Correct endpoint counts in instrumented/reference executions are 247/241, 241/242 and 229/225; early agreement does not certify identical later trajectories. The measurements condition on the instrumented completions.

An oracle retaining only correct predictions and givens permits repeated naked- and hidden-single deductions (one candidate per cell or one place per digit in a row, column or box) to solve the puzzle a median 11/11/12 slow readouts before completion. Seeding deductions from the correct cells at $\kappa - 1$, we compare deduction depth with the first shadow update remaining correct through $k = 6$ for cells wrong at $\kappa - 1$. Per-puzzle Spearman correlations have medians 0.056/0.063/0.028 (68/83/56 eligible puzzles with at least five cells and nonconstant ranks). Most cells have depth one. The oracle supplies knowledge of correctness, and weak association with this deduction order establishes neither simultaneous revision nor the absence of other cascades.

A.3.2 MATCHED-ERROR REPAIR

The matched corruption in Section 4 uses the seed sequence $[20260917, \text{puzzle ID}, 77]$ for NumPy's generator. Empty cells are selected without replacement; each receives a uniformly sampled different digit. All 16 updates execute. The remaining 74 source grids are correct and succeed under both supplied conditions in both original receivers.

Role embedding gives

$$\|E(g) - E(y^\star)\|_F^2 = 2wk\|r_{\text{match}} - r_{\text{other}}\|_2^2.$$

Each of the k wrong digits swaps two roles. Complete digit-grid embeddings have equal norm, and the fast buffer is shared. Count-matching therefore controls distance to the encoded answer and initial norm in MLP and attention receivers. EqR instead has squared distance $w\sum_j \|v_{g_j} - v_{y_j^\star}\|_2^2$ for its digit-specific token vectors, which need not be equal at matched counts. Neither encoding identifies a latent fixed point.

At update 16, MLP repairs 414 EqR grids versus 432 random grids, while EqR repairs 373 versus 430; ordinary fixed starts solve 418 and 382, respectively. For this original MLP/EqR comparison, random-only/model-only successes are 21/3 and 58/1, giving random advantages of 4.11 points [2.05, 6.39] and 13.01 [10.05, 16.21] with 95% intervals from the original 20,000-resample stratified paired bootstrap. The intervals condition on one corruption draw. Supplying EqR's grid instead of the ordinary start gains/loses 11/15 MLP successes and 9/18 EqR successes. After one update, model/random mean error is 41.20/5.91% for MLP and 37.55/2.44% for EqR. Better initial denoising therefore does not determine which receiver eventually repairs more model-produced grids.

Constraint structure and training exposure. Of 10,185 wrong digits in the saved EqR grids, 17 (0.17%) directly contradict a given; reconstructing the documented random draw gives 6,409 (62.93%). General row/column/box duplicates involve 4,635/10,185 wrong EqR digits (45.51%) versus 9,870/10,185 random wrong digits (96.91%), but also implicate 3,425/14,333 correct EqR empty-cell digits (23.90%) and 6,412/14,333 correct random digits (44.74%). Conflict proportions pool cells, counting each once per test; error means weight puzzles equally. The original random grids were reconstructed rather than saved; matching counts does not independently certify their exact positions and values.

The four saved attention corruptions give 62.93–63.58% clue refutability. Saved draw zero follows the earlier reconstruction's seed sequence and procedure and reproduces its 6,409 clue-refuted errors. Table 7 reports their endpoints, early repair and all paired contrasts. Three Attention 256 draw-specific intervals include zero. Relative to ordinary fixed starts, EqR grids change endpoint totals by -2, $+1$ and $+5$ puzzles for Attention 128/192/256, respectively. All three attention models preserve injected correct answers throughout all 16 updates on all 512 repair puzzles. These checkpoints do not establish a causal effect of feature dimension on repair.

On these same 438 puzzles, the ordinary fixed starts leave 41.13/29.84/35.23% mean empty-cell error after one update for Attention 128/192/256, versus 36.96/35.94/35.15% when supplied with EqR's grid. The effect relative to ordinary inference varies across models. Fixed starts are not matched in error count or encoded distance to the supplied grids; the re-encoded EqR/random pair remains the controlled repair comparison.

Random diagnostic grids begin with 41.39% mean error, compared with the anchor branch's expected 8.89% under its sampled redraw probabilities. Among the 155 pairs with at least half the empty cells wrong, MLP solves 136 model grids versus 150 random grids; EqR solves 103 versus 147. The contrast extends beyond typical anchor corruption rates, though not necessarily beyond the training distribution's support. EqR shows the same direction without the anchor objective. Training familiarity and relational structure remain plausible, interacting explanations. A conflict-free wrong digit implies that its reference-solution peer in each relevant unit has also changed; mutual reinforcement in the hidden state and coordinated repair remain hypotheses, not consequences of that combinatorial fact.

Clue-compatible corruption controls. On the same 512 puzzles, four draws per family preserve givens and EqR's error count; analysis retains the 438 source-error cases. One family samples feasible empty cells, then uniformly chooses wrong digits that do not duplicate givens. Every puzzle has enough feasible cells. The second keeps EqR's wrong-cell locations and samples compatible alternatives to EqR's digits, retaining its choice only where no alternative exists (274 of 10,185 wrong cells per draw). Seeds are $[20260926, \text{puzzle ID}, 11, j]$ and $[20260926, \text{puzzle ID}, 12, j]$,

Table 7: Final-attention repair on 438 initially incorrect source grids, $D = 16$. Top: endpoint counts; mean advantage compares four random draws with EqR starts. Middle: puzzle-averaged empty-cell error (random: four-draw mean) and exact counts (random: range); both supplied families start at 41.39% error. Bottom: paired random-only/EqR-only successes and accuracy differences. Brackets are conditional paired 95% bootstrap intervals. Each model uses its selected benchmark weights.

Endpoint repair

Attention	Fixed	EqR	Random draws 0 / 1 / 2 / 3	Mean advantage (pp)
128	419	417	434 / 434 / 435 / 436	4.05 [2.17, 6.11]
192	425	426	437 / 435 / 437 / 438	2.45 [1.03, 4.05]
256	425	430	436 / 437 / 433 / 434	1.14 [0.00, 2.40]

Repair after one update

Attention	EqR error (%)	Random error (%)	EqR exact	Random exact
128	36.96	5.04	104	388–397
192	35.94	2.84	116	404–410
256	35.15	3.80	114	393–408

Draw-specific endpoint differences

Attention	Draw	Random-only	EqR-only	Advantage (pp; 95% interval)
128	0	20	3	3.88 [1.83, 5.94]
128	1	20	3	3.88 [1.83, 5.94]
128	2	20	2	4.11 [2.05, 6.16]
128	3	20	1	4.34 [2.51, 6.39]
192	0	12	1	2.51 [0.91, 4.11]
192	1	11	2	2.05 [0.46, 3.65]
192	2	12	1	2.51 [0.91, 4.11]
192	3	12	0	2.74 [1.37, 4.34]
256	0	8	2	1.37 [0.00, 2.74]
256	1	8	1	1.60 [0.46, 2.97]
256	2	7	4	0.68 [-0.68, 2.28]
256	3	5	1	0.91 [0.00, 2.05]

respectively, for $j = 0, \ldots, 3$. Both families have zero direct clue conflicts. Receivers and encodings are unchanged; MLP/EqR references are rerun alongside the controls.

Table 8: Corruption controls on the same 438 source-error puzzles. Top: error after one iteration (%); random conditions average four draws, except the one-draw MLP/EqR uniform reference. Legal conditions exclude clue conflicts; EqR-cell replacements retain wrong-cell locations. Structured corruptions also reduce peer conflicts. Bottom: structured-draw endpoint counts and mean accuracy difference from EqR grids at iteration 16. Brackets are paired 95% intervals conditional on the four draws.

Receiver	EqR grid	Uniform	Legal, random cells	Legal, EqR cells	Structured
Attention 128	36.96	5.04	16.82	16.21	36.59
Attention 192	35.94	2.84	12.72	15.74	29.59
Attention 256	35.15	3.80	12.86	15.58	31.26
MLP 192	41.20	5.91	18.61	23.39	36.58
EqR	37.55	2.44	13.43	16.72	29.75

Receiver	EqR	Structured draws 0 / 1 / 2 / 3	Difference (pp)
Attention 128	417	417 / 406 / 418 / 418	-0.51 [-2.40, 1.48]
Attention 192	426	429 / 426 / 425 / 427	+0.17 [-1.20, 1.66]
Attention 256	430	427 / 428 / 427 / 427	-0.63 [-1.77, 0.51]
MLP 192	414	413 / 409 / 410 / 403	-1.20
EqR	373	386 / 390 / 391 / 388	+3.60

Every receiver retains an early gap under these two controls (Table 8). For Attention 128/192/256, legal-random endpoint advantages over EqR grids average 1.03/0.86/0.68 points, with paired 95% intervals $[-0.91, 3.08]$, $[-0.46, 2.34]$ and $[-0.51, 1.94]$; location-matched advantages are

1.48/1.08/-0.11, with intervals $[-0.51, 3.60]$, $[-0.17, 2.51]$ and $[-1.20, 0.97]$. Intervals use 10,000 stratified puzzle resamples and condition on the four draws. MLP's legal-random/location-matched endpoint accuracies are 97.37/97.15%, versus 94.52% for EqR grids; EqR's are 92.52/90.24%, versus 85.16%. Feasible-cell sampling changes location distributions; location-matched sampling deliberately changes EqR's digit choices. Neither comparison partitions the gap causally or isolates training familiarity or hidden-state mutual reinforcement.

Structured corruption controls. Four draws reduce conflicts among assignments, using seeds $[20260926, \text{puzzle ID}, 31, j]$. Starting from the solution, a random feasible empty cell receives a clue-compatible wrong digit. Its solution peers holding that digit in the row, column or box are queued for corruption. Queued cells receive compatible wrong digits minimizing additional solution holders outside the corrupted set. Chains restart when the queue empties, until exactly k cells are wrong. New chains prefer digits whose solution holders are already corrupted; ties are random. Up to 12 sweeps reduce duplicates at these fixed locations. Of eight seeded constructions, we select the highest fraction of wrong digits duplicating no peer, without consulting receiver outcomes. These answer-informed starts are repair assays, not inference strategies.

All draws preserve givens and EqR's error count. Of the 10,185 wrong digits, 56.91–57.56% duplicate no peer, versus 54.49% in EqR grids, 6.38% in the first clue-compatible random draw and 3.09% in the first uniform draw. These grids remain partially conflicting. Early repair slows in every receiver (Table 8). Attention endpoint differences have conditional paired intervals including zero, without establishing equivalence; MLP endpoints are slightly lower and EqR endpoints higher. Reproducing the early difficulty without model-generated digits changes locations, values and relationships together, without isolating a conflict measure or training familiarity.

A.3.3 Communication and retained-state controls

Table 9 reports the shared-first-state attention comparisons on 256 puzzles, separating endpoint changes against intact continuation from new discoveries. Under slow reset, Attention 192/256 each lose seven initially correct puzzles at the endpoint; subtracting initial from final totals therefore misses some new discoveries. Fast-reset intervals include zero without establishing equivalence. Message removal yields no new solution and zero endpoint successes under both full-grid and empty-cell scoring. These conventions differ only at one Attention 256 observation: update 2 has correct empty cells and a wrong clue on a puzzle already correct at update 1.

Replaying earlier messages. Each puzzle's 42 first-iteration cross-field message tensors (21 stack applications, two blocks each) replace their counterparts before residual addition and normalization in iterations 2–16. This sequence repeats, rather than remaining constant within an iteration; both carries, spatial mixing and direct input continue normally. Population-mean replay averages corresponding tensors over all 256 puzzles, including the recipient, with one bank across batches.

Puzzle-specific replay preserves all 58/94/79 initial solutions throughout, but adds 0/6/0, versus 192/156/171 intact (Table 9). The six Attention 192 discoveries start with 1, 1, 1, 3, 4 and 5 errors; the first three complete at iteration 2, the others at 3 (a post-run subdivision). Wrong grids can still change; repeated predictions do not establish latent fixed points. Mean replay returns no solution, although three Attention 192 puzzles become correct transiently. Full-grid and empty-cell endpoint scoring agree. Preservation and discovery thus respond differently to message updating, without identifying semantics: stale messages can conflict with later states, and averaging changes their magnitude and structure.

In MLP 192, intact continuation preserves all 42 initial solutions and adds 207; later-message removal leaves 24 and adds none. Of the 42, 22 lose correctness at least once and four recover. First-iteration-only removal followed by restored communication returns 245 versus 249 intact, with nine losses and five gains; removal throughout yields no solution. MLP 384 initially solves 67, returns 242 intact and zero with later messages removed, with no new discovery under message removal. Attention 128's repeated comparison is a reproduction at the same weights, not another training replicate. The first-iteration-only restoration control was performed on MLP 192. Recovery after restoring messages could include recomputation of digit-specific clue information; it does not isolate preserved intermediate deductions from renewed access to the input.

Scaling both MLP message projections to 0.5 throughout separates decoded progress from probability improvement. Mean empty-cell error falls from 49.38% to 42.58% between updates 1 and 16, while NLL rises from 1.252 to 1.498 nats; 42 individual puzzles have both fewer errors and worse NLL. Exact counts rise from four to 63. Here NLL uses full-vocabulary StableMax on empty cells and

saved float32 logits, unlike the digit-conditioned trajectory diagnostic above. The scale acts before residual normalization and is not a fraction of useful information.

The original four-model reset study uses a separate stratified 256-puzzle set and digit-only decoding (Table 9). Its 20,000-resample paired 95% intervals for fast-reset accuracy changes, in table order, are $[-6.64, -0.39]$, $[-4.69, 0.00]$, $[-6.25, 0.78]$ and $[-7.42, 0.00]$ points; slow-reset intervals are $[-40.63, -28.91]$, $[-65.23, -53.52]$, $[-72.27, -60.55]$ and $[-85.55, -75.78]$. These equal-sized populations have distinct cases and intact references. In both studies, resets preserve the fast state's within-iteration computation. The asymmetry does not establish a unique fast-state equilibrium conditioned on the slow state.

Table 9: Communication and carry interventions, $D = 16$, 256 puzzles per population. Attention initial correct/incorrect counts are 58/198, 94/162 and 79/177. Lost/gained and accuracy changes (paired 95% intervals) compare endpoints with intact; new ever/final counts concern initial failures. Original resets use different puzzle IDs and their own intact control. Labels name the reset state; all within-iteration computation remains present. Replay also reports initially correct puzzles lost at the endpoint.

Final attention: shared first states and common population

Attention	Condition	Final	Lost/gained	New: ever/final	Change (pp; 95% interval)
128	Intact	250	0/0	192/192	0.00 [0.00, 0.00]
128	Messages removed	0	250/0	0/0	-97.66 [-99.22, -95.70]
128	Fast reset	246	6/2	188/188	-1.56 [-3.91, 0.39]
128	Slow reset	58	192/0	0/0	-75.00 [-80.08, -69.53]
192	Intact	250	0/0	156/156	0.00 [0.00, 0.00]
192	Messages removed	0	250/0	0/0	-97.66 [-99.22, -95.70]
192	Fast reset	245	7/2	151/151	-1.95 [-4.30, 0.39]
192	Slow reset	101	149/0	15/14	-58.20 [-64.06, -51.95]
256	Intact	250	0/0	171/171	0.00 [0.00, 0.00]
256	Messages removed	0	250/0	0/0	-97.66 [-99.22, -95.70]
256	Fast reset	246	6/2	167/167	-1.56 [-3.91, 0.39]
256	Slow reset	126	124/0	63/54	-48.44 [-54.69, -42.19]

Original resets: separate population and digit-only decoding

Checkpoint	Intact	Fast reset	Lost/gained	Slow reset	Lost/gained
MLP 192 (46k)	247	238	13/4	158	90/1
MLP 384 (16k)	241	235	8/2	89	152/0
Our TRM (50k)	222	215	15/8	52	170/0
EqR	229	220	16/7	22	207/0

Message replay: same attention population

	Puzzle-specific replay			Population-mean replay		
Attention	Final	New: ever/final	Initial lost	Final	New: ever/final	Initial lost
128	58	0/0	0	0	0/0	58
192	100	6/6	0	0	3/0	94
256	79	0/0	0	0	0/0	79

Slow state beyond its readout. On the same 256 shared first states, we retain the fast state and replace slow state before each iteration 2–16. Re-encoding embeds the current decoded grid with clues restored, preserving its non-clue argmax but changing score magnitudes. Two further edits preserve the linear digit scores algebraically. For each cell/digit vector, decompose $h = h_{\parallel} + h_{\perp}$, where $h_{\parallel} = v_\theta(v_\theta^\top h)/(v_\theta^\top v_\theta)$. Replace $h_{\perp}$ with either the initial buffer's orthogonal component or a fresh projected Gaussian vector rescaled to $\|h_{\perp}\|_2$. The random seed is $[20260926, \text{puzzle ID}, 21, t]$, with zero-based $t = 1, \ldots, 15$. The latter edit preserves each vector's norm as well as its readout score; perturbation magnitudes are not matched across the two replacement families.

Table 10 separates the substantial cost of replacing features beyond the scores from sensitivity to the replacement chosen. With random replacements, all 58/94/79 initially correct puzzles remain correct at the endpoint; 113/118/151 initially incorrect puzzles become correct, versus 192/156/171 intact. Against intact endpoints, losses/gains are 82/3, 41/3 and 21/1. These subdivisions are retrospective. The fast state, digit scores, per-vector norms and puzzle input retain information, and each iteration

Table 10: Slow-state replacements before iterations 2–16, retaining the fast state. Correct endpoints out of 256 shared puzzles. Score-preserving replacements keep the readout-parallel component; the random replacement also keeps each orthogonal component's norm.

Slow-state treatment	Attention 128	Attention 192	Attention 256
Intact	250	250	250
Full reset	58	101	126
Re-encode decoded grid	148	152	151
Keep scores; initial-buffer remainder	109	138	122
Keep scores; matched-norm random remainder	171	212	230

can reconstruct features. Random replacement is therefore not a neutral erasure of memory; neither its loss nor the larger buffer-reset loss identifies the content or isolated contribution of stored history. The execution reproduces the original intact first batch bitwise at all 16 iterations. A separate numerical replay on the first 16 puzzles per model checks both orthogonal replacements before all iterations 2–16 after float32 conversion: maximum digit-score change is below 9×10^{-5}, with no immediate argmax changes. This bounded check supplements algebraic score preservation; the original full-run precision check preceded the float32 cast.

A.3.4 Initialization and preservation

Table 11 reports all 38 checkpoints, including fixed-start recoveries and the strong 46k benchmark result. From 14k onward, fixed-start endpoints range from 11 to 127; independent and shared Gaussian counts range from 120 to 127 and 121 to 128. The 50k continuation restores weights, optimizer and EMA but resets training carries and changes from four to eight devices. Global batch size is preserved; random streams and reduction order change. The series does not isolate the cause of later instability.

Table 11: Complete MLP 192 series on 128 validation puzzles at depth 16. Read the left block, then the right. F/I/S/B give endpoint counts for fixed, independent, shared and broadcast Gaussian starts; L counts fixed-start terminal losses. At all 11 steps with a broadcast measurement, the answer-injection control holds all 128 answers at every update; both extra controls are unmeasured at dashes. The benchmark selection is 46k. Checkpoints share a training lineage, puzzles and draws.

Update (k)	F	I	S	B	L	Update (k)	F	I	S	B	L
2	0	0	0	–	0	78	99	123	125	103	11
6	61	59	60	–	0	82	99	127	124	–	12
10	118	117	115	–	0	86	43	125	121	–	60
14	119	121	123	122	0	90	12	124	126	–	75
18	66	125	124	–	1	94	11	124	124	121	57
22	31	127	125	123	17	98	32	125	121	–	24
26	122	126	125	–	0	102	30	123	121	–	53
30	127	125	125	125	0	106	48	126	123	–	35
34	124	125	124	–	0	110	63	120	122	125	20
38	123	124	127	–	0	114	14	123	125	–	58
42	123	127	128	–	0	118	16	124	123	–	53
46	126	124	127	123	0	122	57	121	126	–	48
50	124	126	126	–	0	126	99	122	125	112	9
54	121	122	125	–	4	130	69	125	124	–	28
58	120	126	124	–	4	134	50	125	126	–	59
62	120	126	126	123	4	138	38	124	123	–	65
66	125	126	126	–	0	142	114	122	122	94	3
70	117	126	124	–	4	146	97	123	124	–	1
74	117	124	125	–	4	150	111	124	123	115	0

For the MLP's full Gaussian pair of shape $2 \times 9 \times 81 \times 192$, the independent condition uses seed sequence $[4242, i, 0, 7]$ for local validation index i; the shared pair is the draw at $i = 0$. The broadcast control uses $[4242, 0, 0, 11]$ to draw shape 2×192 and repeats it across cells, fields and puzzles. All draws are float32 and reused across checkpoints. A full shared tensor must be permuted with digit labels for samplewise equivariance; the broadcast control is field-invariant. Its changed scale and feature pattern are not separately controlled.

At 94k, the fixed, independent, shared, broadcast and answer-injected starts solve 10, 39, 37, 6 and 128 puzzles after update 1. Their any-loss counts are 62, 0, 0, 7 and 0; five fixed-start and three

broadcast-start cases recover by the endpoint. The two full Gaussian conditions share 123 endpoint successes and have one unique success each. Independent starts solve all 57 fixed-terminal-loss cases and 56 of the 60 never-correct cases; shared starts solve 56 and 57, respectively. At local index 1, the fixed trajectory is exact only at updates 3 and 15, while both full Gaussian trajectories are exact throughout.

Even perfect injected-answer preservation does not imply stationarity: all 128 late-state residuals are nonzero at 94k, with median 0.0179490 and range 0.00212425–0.0460018 for the final-three-update mean absolute state change. These are coordinate-dependent movements, not stability constants. Table 12 reports every final-attention draw on the same 128 validation puzzles at depth 16. Numerical sensitivity and fresh-inference dependencies remain as specified in Appendix A.2.3.

Table 12: Initialization on the same 128 validation puzzles at depth 16. Counts are endpoints except the answer column, which counts preservation at every step. No fixed or Gaussian trajectory loses correctness after reaching it under these conditions.

Attention	Fixed	Independent draws 0 / 1 / 2 / 3	Shared draws 0 / 1 / 2 / 3	Answer
128	126	125 / 122 / 128 / 123	127 / 125 / 124 / 128	128
192	126	128 / 128 / 127 / 125	127 / 128 / 125 / 126	128
256	128	127 / 125 / 127 / 127	128 / 128 / 127 / 127	128

A.3.5 Candidate generation, ranking and nested pools

Let c_{ij} indicate endpoint correctness of candidate j for puzzle i, and $j_i(k)$ the earliest minimum-residual index among its first k candidates. Then

$$C_k = \sum_i \mathbf{1}[\max_{j \le k} c_{ij} = 1], \qquad S_k = \sum_i c_{i,j_i(k)}, \qquad N - S_k = (N - C_k) + (C_k - S_k).$$

Dividing by N gives fractions, separating returned errors into uncovered puzzles and missed available answers. For m scalar coordinates in both states, the unrounded score is

$$r_{ij} = \frac{1}{3m} \sum_{t=D-2}^{D} \|s_t^{(i,j)} - s_{t-1}^{(i,j)}\|_1.$$

At $D = 64$ this uses transitions $61 \to 62$, $62 \to 63$ and $63 \to 64$, including TRM's 16 prefix positions. Scores are rounded to float16 before storage; higher-precision storage does not recover the unrounded ordering. Ties retain the earliest index. Gaussian starts use seed sequence $[4242, \text{test ID}, j - 1, 7]$, with float32 draws; all other weights, inputs and operations are unchanged.

Table 13: Nested depth-64 pools for our TRM implementation, seed 0 at 50k, on 20,000 test puzzles. Each larger pool preserves the endpoints and stored scores of every earlier candidate. Fixed-start trajectories are excluded.

Starts k	Covered C_k	Selected S_k	Uncovered $N - C_k$	Missed $C_k - S_k$
1	18,425	18,425	1,575	0
2	19,002	18,792	998	210
4	19,295	19,001	705	294
8	19,492	19,099	508	393
16	19,643	19,110	357	533
32	19,766	19,015	234	751
64	19,869	18,832	131	1,037
128	19,941	18,496	59	1,445

Table 13 shows the entire baseline prefix curve. From eight to 128 starts, 449 more puzzles become covered while the number of missed available answers grows by 1,052, producing 603 fewer correct returns. Every one of the 1,205 paired losses selects a new wrong candidate with a strictly smaller stored residual. For the smallest test ID among these losses, 151, all eight early candidates and 121 of 128 are correct; the selector switches from correct draw 4 (0.006511688) to wrong draw 33 (0.003000259). Among all losses, the median bank has 124 correct candidates; 1,146 have a strict correct majority, one has exactly half and 58 have fewer. Grid majority voting would solve the strict-majority cases; unsaved wrong grids prevent determining the complete majority/plurality result.

Training and ranking. Table 14 reports fixed-start and first-Gaussian outcomes alongside coverage and selection. In our TRM, fixed-start counts span only 18,414–18,893, while the two baseline seeds' selected counts differ by 7,768 despite coverage differing by ten. Adding randomized starts covers 278 fewer puzzles than seed-0 baseline but returns 1,163 more correct answers. All four use a 50k budget, batch 768, learning rate 10^{-4}, damping 0.05 and training noise 0.01; only the named training start/anchor conditions and seed differ in the recipes, while selected ages and training histories also differ.

Table 14: Training, generation and selection. Fixed is a separate fixed-start rollout; first is the first Gaussian candidate. All trajectories have depth 64. Missed and uncovered counts partition returned errors. Recipe arms retain their selected ages; these are checkpoint comparisons, not replicated ingredient effects. The attention fixed-start counts here and in Table 17 come from separate executions on the same IDs.

Recipe	Step	Fixed	First	C_k	S_k	$C_k - S_k$	$N - C_k$
Our TRM: $N = 20{,}000$, $k = 128$, training budget 50k							
Baseline, seed 0	50k	18,573	18,425	19,941	18,496	1,445	59
Baseline, seed 1	40k	18,893	11,698	19,951	10,728	9,223	49
Answer anchors only	42k	18,805	10,663	19,972	7,645	12,327	28
Random starts only	48k	18,414	18,398	19,663	19,659	4	337
Earlier Attention 256: $N = 5{,}000$, $k = 32$, training budget 30k							
Reference	28k	4,980	4,979	4,997	4,996	1	3
No starts or anchors	30k	4,987	4,956	4,998	4,929	69	2
No damping or noise	20k	4,981	4,977	4,998	4,998	0	2
Changed batch and rate	22k	4,576	4,587	4,868	4,867	1	132

Across these 11 banks, the seven checkpoints trained with Gaussian starts miss at most four available answers; the four trained without them miss 69–12,327. This correspondence links training-state exposure to inference reliability, but does not isolate its cause or establish a universally sufficient training ingredient. The banks differ in architecture, selected checkpoint age, population size and candidate count. Likewise, the MLP initialization series associates reliability with training-familiar start families without isolating why the fixed start deteriorates.

Uniform choice from a complete bank has expected accuracy $N^{-1} \sum_i k^{-1} \sum_j c_{ij}$. Within a mixed pool containing both correct and incorrect candidates, the area under the receiver operating characteristic curve (AUC) is the fraction of correct–incorrect pairs in which the correct candidate has the smaller residual, with half credit for ties. Table 15 averages AUC equally across mixed pools; 0.5 means no average pairwise advantage. Mixed-pool cohorts differ across recipes. The anchor-only mean is near chance, not uniformly reversed; seed 1 has weak positive ranking while selecting below uniform expectation. Selecting the minimum depends on the extreme scores, not average ordering.

Table 15: Uniform-choice expectation and within-puzzle ranking in 128-candidate, depth-64 banks on 20,000 common puzzles. AUC averages only pools containing both correct and incorrect candidates, with equal weight per puzzle and half credit for residual ties.

Recipe	Uniform (%)	Selected (%)	Mixed pools	Mean AUC
Baseline, seed 0	91.946	92.480	12,496	0.611
Baseline, seed 1	58.425	53.640	19,951	0.585
Answer anchors	53.637	38.225	19,972	0.488
Random starts	91.859	98.295	3,492	0.984

Table 16: Nested restart banks on the same 5,000 puzzles. Every selection loss displaces an earlier correct candidate with a new wrong candidate of strictly smaller stored residual.

Attention	Covered, $8 \to 128$	Selected, $8 \to 128$	Gained	Lost
128	4,997 → 4,998	4,996 → 4,998	2	0
192	4,994 → 5,000	4,994 → 4,999	6	1
256	4,997 → 4,999	4,997 → 4,996	1	2

For final Attention 128/192/256, the first Gaussian candidate solves 4,971/4,980/4,981 of the same 5,000 puzzles. Table 16 shows the eight-to-128-start comparison and its rare strict displacements. These rare events establish a possible selection failure; this 5,000-puzzle sample does not establish a large practical bottleneck or its prevalence. One Attention 256 depth-64 trajectory costs about 2.014

TMACs; 128 cost 257.839, before selection or verification. Parallelism can reduce latency without reducing arithmetic. These comparisons do not determine the optimal allocation between depth and restarts at matched cost. Verifier and reference counts agree per puzzle in all 11 banks; this does not measure deployment time.

A.3.6 Recipe sensitivity and performance attribution

In the earlier Attention 256 study, the batch-size/learning-rate bundle changes batch 768 to 272 and rate 10^{-4} to 5×10^{-4}. Its selected checkpoint is at 22k rather than 28k; examples processed and optimizer/weight-decay timescales also differ. Other arms jointly remove randomized starts and anchors, or damping and update noise. Table 17 gives each arm's fixed-start depth sweep under the common 30k budget.

Table 17: Fixed-start depth sweeps for the earlier Attention 256 recipe study, seed 0, a common 30k-update budget and 5,000 test puzzles. Each arm uses its own selected checkpoint at both depths. These are the depth-sweep executions, distinct from the candidate-bank executions.

Recipe	Selected step	Correct at 16	Correct at 64
Reference	28k	4,912 (98.24%)	4,980 (99.60%)
No random starts or anchors	30k	4,921 (98.42%)	4,988 (99.76%)
No damping or update noise	20k	4,902 (98.04%)	4,977 (99.54%)
Changed batch size and learning rate	22k	4,271 (85.42%)	4,575 (91.50%)

Candidate-bank fixed-start counts in Table 14 differ by one to four from three depth-sweep counts on the same IDs; both executions are reported. Jointly removing starts and anchors gives four selected gains and 71 losses relative to the reference at 32 starts. Within that ablated bank, expanding from one to 32 starts yields 39 gains and 66 losses, while covering 42 additional puzzles. Removing damping and noise still returns all 4,998 covered answers; the changed batch/rate arm leaves 132 puzzles uncovered and misses only one available answer. The largest tested fixed-start change is the batch/rate bundle's 12.82/8.10-point decrease at depths 16/64; the other contrasts are at most 0.20 points, not evidence of zero effects. Removing damping changes the inference map as well as training.

A separate TRM-cell control has 795,909 parameters versus MLP 192's 789,125, with the same general optimizer, randomized-start and anchor regime. It changes the cell and input/state representation, adds digit augmentation and trains for 960k updates versus 50k. Post-warmup timestamps suggest approximately equal allocated training time, without matching training arithmetic or time to the independently selected checkpoints. Its selected 660k weights solve 254,779/422,786 puzzles at depth 16 (60.26%) and 62,402/100,000 at depth 64 (62.40%). The three MLPs average 95.41% on the full depth-16 test and 99.01% on those same 100,000 depth-64 IDs. This tests one trained cell configuration, without searching for the best TRM at that size.

These comparisons do not isolate an ingredient or reproduce SE-RRM training. They make optimization and training exposure plausible contributors alongside representation, update design and their interactions, but neither control isolates the source of Attention 128's accuracy. FRM's separate 99.5% operating point uses difficulty-sampled training sets per seed and validation-selected checkpoints and inference settings (Helbling et al., 2026), with the population and budget differences described in Section 3.

A.4 Extended ARC-AGI and ConceptARC Analyses

A.4.1 Public TRM: conditioning, trajectories and candidate pools

We use alphaXiv's ARC-AGI-1 h3l4 checkpoint at training step 259,320. Each outer iteration has three cycles of four fast updates and one slow update. Query input and a learned task-ID embedding condition the recurrence, under the exposure regime in Section 7. Evaluation uses the identity input view, float32 arithmetic on MPS and batches of 16, bypassing native halting and augmentation voting. Success requires exact output dimensions and cell values; counts refer to queries, not tasks solved on every query. Demonstrations are deduplicated within each task. At depth 16, 1,295/1,363 are reproduced correctly. The 352 tasks with complete fit give 121/367 correct queries; the remaining 48 give 7/52. This association does not isolate a cause of failed transfer.

For demonstration-state transfer, both terminal states from the last example in recorded order, initialized independently, replace the query's initial states. Query input, task ID and weights are unchanged. At depth 16, three gains and three losses leave 128 correct queries, of which 125 also

succeed from the fixed start. This intervention does not isolate reusable task information from information specific to the demonstration input.

Prior ARC work reports early TRM accuracy saturation (Roye-Azar et al., 2025). Our fixed-start correct counts are 124, 132 and 128 at updates 1, 2 and 16, with 12 gains and eight losses from 1 to 16. First discoveries occur on 124, 13 and one query at updates 1, 2 and 3, respectively. Among the 138 ever-correct queries, 12 lose correctness at least once and ten end incorrect. The depth-64 extension reproduces all 6,704 depth-1–16 correctness flags and has no later correctness changes, although 47 incorrect cropped grids continue changing. This does not imply stationary latent states.

The eight-start, depth-16 bank samples both complete state arrays from independent standard Gaussians and excludes the fixed start. At prefixes $k = 1, 2, 4, 8$, endpoint coverage is $134, 134, 135, 136$, whereas minimum-residual selection returns $134, 133, 133, 131$ correct answers: one gain and four losses from $k = 1$ to 8. The residual is final-update mean absolute slow-state change, stored without float16 rounding; ties select the earliest draw. Fixed-trajectory time coverage and restart-endpoint coverage overlap on 128 queries, with ten and eight unique to each. Their 146-query union in Section 7 consults reference answers. Replacing the fixed depth-16 trace with depth 64 leaves this union unchanged and raises cost from 144 to 192 outer updates per query.

A separate uniform sample without replacement (seed 20260921) covers 40 queries from 40 distinct tasks. Eight fresh Gaussian starts run to depth 64. At depths 16, 32 and 64, prefixes $k = 1, 2, 4, 8$ give endpoint coverage, minimum-residual selection and maximum-halt-logit selection of $12, 12, 13, 13$; coverage over all intermediate readouts is $12, 12, 14, 14$. Of 320 trajectories, 101 ever succeed and 98 finish correct, with no correctness changes after depth 16. Adding the fixed trajectory leaves coverage at 14/40, at $64 + 8 \times 64 = 576$ updates per query.

A.4.2 FPRM: CONDITIONING AND SOLVER CONTROLS

This diagnostic uses the released final EMA weights (step 518,071) of the Fixed-Point Reasoner (FPRM; Movahedi et al., 2026) on the same 40 queries and their 150 demonstrations (float32, MPS, batch eight), with four native truncated-normal starts per query and one per demonstration. The reconstructed task-ID table matches the builder code and expected 876,406 rows; without the original assignment, its historical correspondence remains unverified. These results do not reproduce the published augmented benchmark score.

FPRM blends a proposed state $\widetilde{z}$ with the current state z using an adaptive step size; its solver steps are not equal-cost units to TRM outer updates. Its residual precedes damping:

$$r = \max_j \frac{\|z_j - \widetilde{z}_j\|_\infty}{\|\widetilde{z}_j\|_\infty + 10^{-8}},$$

where j indexes tokens. The solver retains step size, patience and best residual in addition to neural state. After five updates without a qualifying improvement, step size is multiplied by 0.9 if $r \geq 0.1$. The solver stops a whole batch when all residuals are below 0.1, all step sizes are below 10^{-3}, or 1,000 steps have elapsed. Batch membership therefore affects continuation. We compare improvement margins 0.01 (release-era code) and zero (strict improvement), keeping seeds and batches paired.

Both margins yield 140/150 correct demonstrations. Tasks with complete fit give 15/34 correct queries on the first start, versus 1/6 otherwise. At margin 0.01, cyclically reassigning task IDs, with inputs, weights, seeds and batches unchanged, reduces demonstration success to 13/150: 127 paired losses and no gains; no task retains complete fit. At common steps 16/32/64, original versus reassigned counts are 65/2, 141/14 and 143/15. Conditioning matters beyond differences in stopping time; this control neither certifies the historical ID mapping nor identifies what the embeddings represent.

For the first start, (current, ever-correct) counts are $(1, 18)$ at step 8, $(12, 20)$ at 16, $(18, 20)$ at 32 and $(16, 20)$ at both 64 and native stopping. All 20 first succeed by step 12: the six gains from 16 to 32 recover earlier answers, and two are subsequently lost. Nineteen queries lose correctness at least once; four end incorrect. Across all four starts, all-step coverage remains 20, while endpoint coverage and both residual and halt-logit selection equal 16. Thus four discovered answers are lost before endpoint collection. Step 32 was examined after observing trajectories and is not a validated stopping rule.

For each margin, 12/20 query batch trajectories (five batches, four starts) stop at the step-size threshold, eight reach the cap and none satisfies whole-batch residual tolerance. Only 12/160 query endpoint residuals are below 0.1; all are incorrect. Neither small damped movement nor solver termination establishes a fixed point. The margins give identical paired endpoint and ever-correctness flags, despite different lengths for 72/160 query and 134/150 demonstration trajectories. Three query and two demonstration endpoint grids differ, all remaining wrong.

A.4.3 SUPPLIED-ANSWER RETENTION AND DISCOVERY

We separately examine the auxiliary answer-canvas models A and B on ConceptARC (Moskvichev et al., 2023): 48 tasks (three from each of 16 concept groups), with 144 queries per checkpoint. The tasks are absent from both pretraining lists; concept groups are not held out. Network weights remain frozen while a task vector is fitted to support examples. The last support pair supplies validation; the earliest exact validation prediction is selected, otherwise selection uses validation exactness, pixel accuracy and fitting loss in that order. Query answers do not select the task vector.

For retention, the correct one-hot 32×32 answer canvas is supplied once, the companion state is reset, and all eight readouts under the final time-conditioned update must remain exact. Discovery instead uses endpoints from 32 uniform random full-color canvases, continued for 16 updates along the trained time schedule, repeating its final map after trained depth six. Both assays require exact cropped dimensions and cell values.

Table 18: Supplied-answer retention versus discovery in 32 random-start endpoints, on the same 144 ConceptARC queries per checkpoint. Retention and discovery use the distinct continuation schedules specified in the text.

Checkpoint	Answer retained		Answer not retained	
	Found	Not found	Found	Not found
A	21	19	1	103
B	7	20	0	117

Table 18 shows retention without discovery on 19 and 20 queries; A also discovers one answer that fails retention. The assays therefore separate preservation from discovery over their finite horizons. Because models, starts and schedules differ from the Sudoku assays, this is not a controlled comparison between domains. Starting from the model's empty canvas gives 11 correct endpoints for A and seven for B. Selection from the random starts also returns 11 and seven, versus coverage of 22 and seven. It minimizes the mean absolute difference between proposed probabilities and the current canvas state, averaged over the final three updates, with earliest-draw ties.

An exploratory TRM control uses the same 40-query sample. At depth 16, we swap predicted and target logits at incorrect tokens, including padding and end markers. A readout pseudoinverse gives the minimum-norm slow-state change realizing those specified logits, preserving its readout nullspace and task prefix; grid-correct cases are unedited. All 40 immediate readouts are exact. With either retained or reset fast state, none of the 28 corrected failures remains exact throughout eight more updates or at their endpoint. Retaining the fast state gives one transient first-update success; untouched continuation preserves the original 12 successes throughout. This tests one readout edit, not a validated encoding of a solution into the internal state.

A.5 RELATION TO PRIOR WORK AND EXPLANATORY INTERPRETATION

Refinement and state use. Universal Transformers reuse shared transformations across depth (Dehghani et al., 2019); deep equilibrium models seek equilibrium states through root-finding (Bai et al., 2019). Ren & Liu (2026) observe early loss reduction, per-puzzle plateaus and abrupt completion in HRM, interpreting delayed success as escape from neighborhoods of incorrect states. Our measurements distinguish substantial offsetting revisions from small net progress before completion: a plateau in answer quality need not mean inactivity. The models and cohorts differ; decoded revisions do not certify valid deductions.

State interventions answer complementary questions. Training with different numbers of carries (Jolicoeur-Martineau, 2025) tests whether another architecture can learn a solution; editing a carry at fixed weights tests sensitivity to its state. The recurrent-feature, memory and state-transplant interventions discussed in Sections 4.2 and 8 precede this approach. HRM ablations, freezing and state transplants support iterative refinement with different state roles on Sudoku and Maze (Rodrigues & Kang, 2026). Finite-noise patching in HRM and TRM measures spatial propagation (Miyanishi & Morimura, 2026); our communication intervention tests continued dependence after an identical first state. Neither comparison identifies message content.

Access, training and selection. EqR compares Gaussian and learned input-dependent initializers (Huang et al., 2026); path-independence studies connect initialization to steady-state behavior (Anil et al., 2022), and basin analyses examine convergence-time geometry and transient trapping (Lai et al., 2026). At the examined MLP checkpoint, a shared alternative start exposes solving capability

hidden by the default start, without puzzle-specific search. This finite comparison neither estimates basin volumes nor establishes asymptotic path independence.

Revealing extra clues (Ren & Liu, 2026), rollout-derived self-conditioning (Helbling et al., 2026) and diffusion curricula (Drozdova et al., 2026) already make training-state exposure part of the explanation. Deep Improvement Supervision uses intermediate targets from a corruption schedule ending at the answer (Asadulaev et al., 2026); our main and answer-anchor branches instead target the complete solution. Training-state exposure and the prescribed response are distinct interventions. Our recipe comparisons separate generation from residual ranking without isolating a universally sufficient ingredient. HRM input relabeling and checkpoint ensembles diversify inference before voting among halted outputs (Ren & Liu, 2026). PTRM and *Speed is Confidence* separate generation and selection (Sghaier et al., 2026; Dillon, 2026); best-of-n proxy overoptimization is also established (Gao et al., 2023). Repeated sampling can grow coverage while majority-vote and reward-model selection plateau (Brown et al., 2024). Imperfect pass/fail verifiers limit attainable accuracy; with a cost for false positives, further sampling can lower expected utility (Stroebl et al., 2024). That utility result differs from our exact-accuracy decline under minimum-residual ranking. Our contribution concerns the learned state-motion proxy: tracing wrong-candidate displacements with earlier scores unchanged, relating selection to training, and testing transfer to benchmark models where selection is largely effective. Ranking endpoints is distinct from FPRM's use of convergence for adaptive stopping (Movahedi et al., 2026).

Beyond prior ARC saturation and conditioning analyses (Roye-Azar et al., 2025), our measurements separate missing candidates, losses and selection errors. They partially address task-dependent inference scaling (Sghaier et al., 2026); the different exposure regimes and assays do not isolate a cause of the Sudoku–ARC difference. Test-time training provides a precedent for adapting to ARC demonstrations by updating model parameters (Akyürek et al., 2025).

Distinguishing the proposed mechanism. Computational dependencies leave open which represented information guides individual corrections. Matched controls in latent-thought models examine this distinction (Aswal et al., 2026), without establishing it in our models. The measured dependencies motivate the learned-relaxation conjecture in Section 9, without identifying mutually supporting errors in the hidden representation. A triggering correction followed by a deduction cascade within one outer iteration is also compatible with the observed jump. Such a cascade could implement coordinated revision without establishing that mutually supporting errors sustained the preceding difficulty. The within-cycle probes in Appendix A.3.1 show partial score improvement followed by further correction during the completing cycle; targeted interventions are still needed to explain why correction was delayed.

The repair controls make model origin unnecessary for difficult repair, without isolating one structural cause or demonstrating internal reinforcement. Varying training exposure could separate relational structure from familiarity. Message and state interventions were not performed at 94k and do not explain its initialization failure.

Replay separates preservation from discovery. Evolving messages could guide later revisions, or stale inputs could constrain predictions near earlier states; population averaging also changes magnitude and structure. State replacement shows dependence beyond answer scores, while retained quantities or reconstruction can support continued repair. These controls do not identify semantic contents. State transfers, sham continuations and perturbations matched in size could separate useful information from generic disruption. Hybrid states can leave the usual distribution; a correct readout does not validate their meaning (the ARC pilot in Appendix A.4).

Extending and perturbing incorrect trajectories with small state changes, recording both carries and readouts, could distinguish slow transients, persistent errors and changes missed by the selected residual. Finite persistence would still not prove convergence. ARC's separate task-representation test (Section 9) must add previously missing correct candidates, with preservation and selection measured alongside coverage. Success would establish a useful procedure, not by itself recover represented rules.

A.6 RG as a Reference Mechanism

Renormalization-group (RG) methods describe how interactions change when degrees of freedom are eliminated or combined (Wilson, 1975). They provide a reference for asking whether the retained slow state can support a faithful description of subsequent computation. We have not identified an RG flow of effective interactions in the learned models.

Conditioning and elimination. In a constraint problem, fixing selected variables to chosen values conditions the remaining problem. Statistical-mechanical decimation instead sums over eliminated variables. For a hypothetical discrete energy $E(z_R, z_E)$, retained variables z_R, eliminated variables z_E and inverse temperature $\beta > 0$, the effective energy

$$E_{\text{eff}}(z_R) = -\beta^{-1} \log \sum_{z_E} \exp[-\beta E(z_R, z_E)]$$

preserves the Gibbs marginal over z_R. Reading a digit prediction from our models performs neither operation: predictions remain revisable and are not added to the clues. No such energy has been identified for the learned update; this expression defines the reference operation, not a model of the measured trajectories.

A reduced dynamical description. For fixed input and weights, write the outer update as $F_x(h, l) = F_\theta((h, l); e_\theta(x))$, and project onto the slow state with $P(h, l) = h$. An exact deterministic description in this state alone requires a map G_x satisfying

$$PF_x(s) = G_x(Ps)$$

on a declared set of states traversed by the full dynamics. Thus states with the same retained component must have the same next retained component. The readout already depends only on h. An approximation must specify observable or norm, state population, horizon and tolerated error; similar endpoint accuracy does not establish closure.

The fast-reset experiment chooses one particular reconstruction:

$$L_0(h) = (h, l_0), \qquad G_x^{\text{reset}} = PF_x L_0,$$

where l_0 is the fixed initial fast state. Before each outer update after the first, this rule discards fast history and recomputes both states through the original inner loops. It retains all within-iteration computation and does not reduce its arithmetic cost or reconstruct a measured fast-state equilibrium.

What the interventions establish. The reset asymmetry in Appendix A.3.3 shows that the trained computation tolerates the fixed fast-state reset better than its slow-state counterpart. The readout-preserving controls show why this difference cannot be attributed to discarded history alone: the replacement state matters. Paired gains and losses nevertheless rule out exact preservation of endpoint answers by this reconstruction on the tested puzzles. Different correctness statuses imply different grids; agreement establishes neither equal wrong grids nor equal latent states. These measurements do not estimate closure error. Reset states can also differ from naturally reached states, so failure of this reconstruction does not exclude another effective map. Different update frequencies alone establish neither fast-state equilibration nor dynamical timescale separation. Eliminating variables from dynamics can introduce memory effects (Ma et al., 2019), requiring more than a map of the current slow state alone. This reference clarifies a remaining question about learned correction: which history must be retained for faithful continuation?