LRFEED: LATENT RESIDUAL FEEDBACK FOR EFFICIENT REASONING IN LOOPED TRANSFORMERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Looped Transformers have emerged as a promising class of models for solving complex reasoning tasks. However, maintaining effective computation over long reasoning horizons remains challenging for these models. To tackle this problem, we introduce **Latent Residual Feedback (LRFeed)**, which captures how the model's internal latent state evolves and feeds this information back to guide subsequent computation. This feedback helps provide awareness of the reasoning process and enhance reasoning efficiency. We instantiate LRFeed on a single-state variant of the Tiny Recursive Model (TRM), retaining its shared Transformer backbone. Ablation studies show that LRFeed strengthens long-horizon reasoning, even beyond the training horizon. It also promotes more diverse internal reasoning dynamics, reflected in a higher effective rank of channel activity across iterations. Across Sudoku-Extreme, Maze-Hard, and ARC-AGI-1, LRFeed surpasses TRM using less than one-eighth of its inference compute. On Sudoku, LRFeed reaches 97.07% accuracy with extended inference, exceeding TRM by 16.22 percentage points at matched compute. LRFeed also offers a favorable accuracy–compute trade-off against other compact looped Transformers.

1 INTRODUCTION

Looped Transformers have emerged as a powerful and parameter-efficient approach to reasoning (Saunshi et al., 2025; Geiping et al., 2025). By repeatedly applying shared Transformer blocks to an internal representation, or latent state, they increase computational depth without adding parameters (Dehghani et al., 2019; Giannou et al., 2023; Yang et al., 2024). This supports iterative reasoning without generating longer textual traces. Ouro and compute-matched looped mixture-of-experts models further extend this direction to large-scale language modeling (Zhu et al., 2025; Wang et al., 2026). Recent models such as the Hierarchical Reasoning Model (HRM) and the Tiny Recursive Model (TRM) achieve strong performance with remarkably few parameters on Sudoku, maze solving, and ARC—tasks that remain challenging even for frontier large language models (Wang et al., 2025; Jolicoeur-Martineau, 2025). However, making effective use of additional iterations remains a central challenge (Chen et al., 2026; Jeddi et al., 2026).

Although the latent state evolves, the weight-shared Transformer blocks apply the same transition at every step. Such rigid parameter sharing can limit flexibility across iterations (Xu & Sato, 2025; Heo et al., 2025; Bae et al., 2025; Chen et al., 2026). The transition responds to the current state, but receives no explicit description of what its previous update changed. Its adaptation to the reasoning process is therefore left implicit in that state. The model may continue updating its representation without correcting the remaining errors, adding inference cost with little improvement in accuracy (Jeddi et al., 2026; Chen et al., 2026).

Recent work makes shared computation more flexible through additional conditioning signals and selective use of shared parameters. LoopFormer conditions each iteration on time and step size (Jeddi et al., 2026), while RingFormer uses input-dependent level signals (Heo et al., 2025). Polymorphic Universal Transformer activates conditional sparse subspaces within shared parameters (Chen et al., 2026).

These approaches make shared computation more flexible, but effective adaptation also depends on the information that guides it. Their conditioning signals describe the iteration schedule or the current representation, rather than explicitly capturing how the latent state is changing. We observe

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

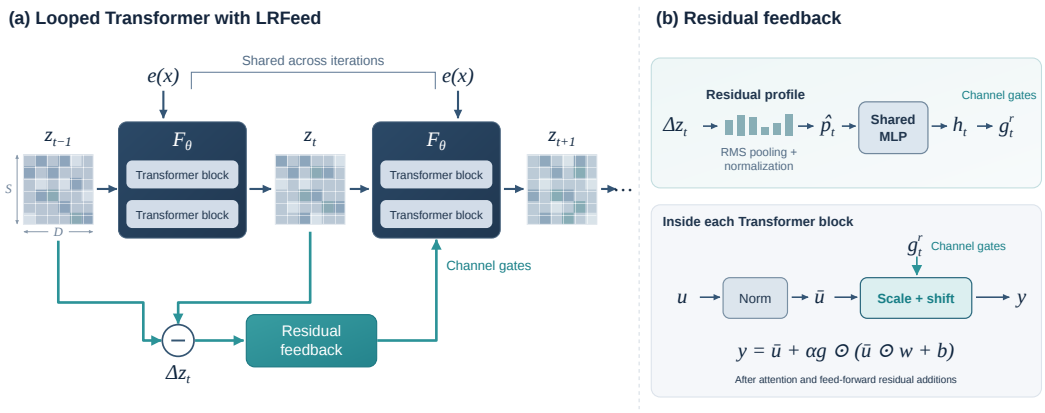


Figure 1: **Latent Residual Feedback.** (a) A shared two-block Transformer repeatedly updates the latent state. The latest change, $\Delta z_t = z_t - z_{t-1}$, provides feedback for the next transition. S and D denote the sequence length and hidden dimension, respectively. (b) RMS pooling and normalization form a residual profile $\widehat{p}_t$. A shared MLP maps it to h_t, from which each modulation site r produces channel gates g_t^r. These gates control scale and shift of the RMS-normalized activation $\bar{u}$ after each attention or feed-forward residual addition. Transformer and feedback parameters are reused across iterations.

that the difference between consecutive latent states, termed the *latent residual*, provides a direct description of the latest update. This motivates *Latent Residual Feedback* (LRFeed), which summarizes recent changes into a channel-wise profile and uses it to modulate the next shared transition (Figure 1).

We first construct a single-state backbone by replacing TRM's two-state hierarchy with one latent state while retaining its shared Transformer blocks. This simpler recurrence avoids the structural constraints imposed by TRM's hierarchical coupling (Appendix A.1). We then introduce LRFeed into this backbone. Pooling and normalization summarize the residual as relative changes across hidden feature dimensions, or *channels*. A small shared network maps this profile to gates inside the Transformer blocks.

Across Sudoku-Extreme, Maze-Hard, and ARC-AGI-1, LRFeed surpasses the corresponding TRM reference with less than one-eighth of its estimated matrix compute. At the longest matched matrix-compute cap on the full Sudoku test set, LRFeed achieves an average accuracy of 97.07% across three seeds, compared with 80.85% for our TRM reproduction. Comparisons with other compact looped Transformers, including HRM, FPRM, and EqR, show a favorable accuracy–compute trade-off for LRFeed (Figure 2).

Matched Sudoku ablations isolate the contribution of residual feedback by comparing LRFeed with its backbone alone, denoted LRFeed (w/o feedback). Adding residual feedback leads to stronger gains from additional reasoning steps beyond the training horizon. Alongside these gains, we observe higher *effective ranks* of hidden-channel activity and output probabilities in later Sudoku iterations (Section 4.4). This pattern offers a geometric perspective on the models' different long-horizon behavior.

Our contributions are threefold:

1. We introduce LRFeed, a lightweight mechanism that turns latent-state changes into explicit feedback for adapting computation in looped Transformers.
2. We identify a late-stage pattern in Sudoku reasoning: LRFeed-trained models exhibit higher effective ranks of channel activity and output probabilities. This observation offers a perspective for studying long-horizon reasoning dynamics.
3. We demonstrate stronger inference-time scaling beyond the training horizon through matched Sudoku ablations against LRFeed (w/o feedback). Evaluations across three

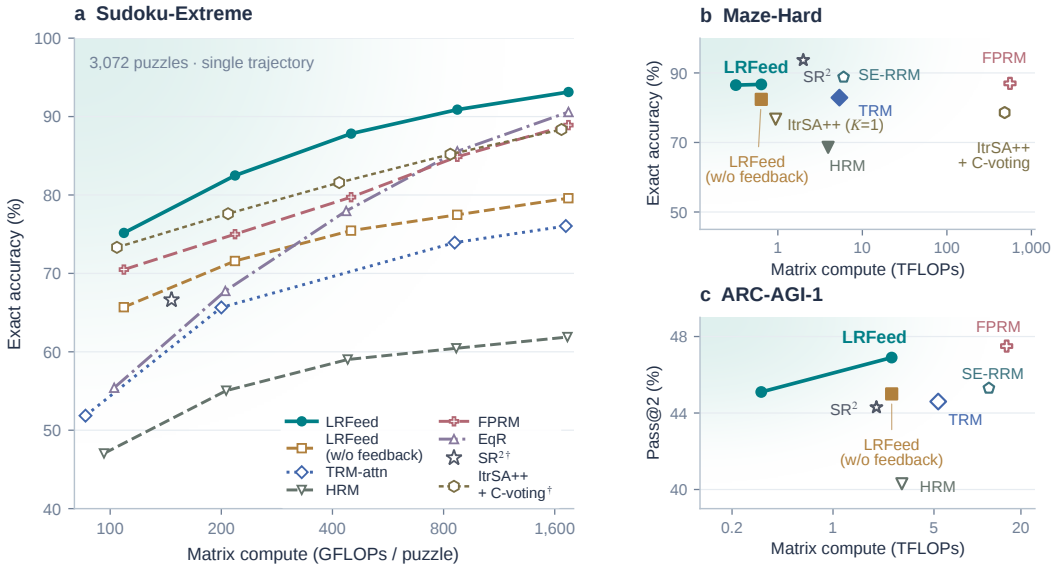


Figure 2: **Accuracy–compute trade-offs across three benchmarks.** (a) Sudoku curves use one trajectory per run on 3,072 common puzzles; † marks published references (Deng et al., 2026; Kubo et al., 2026). (b,c) Maze and ARC show selected-budget results from individual runs (filled markers) and published references (open markers). ARC compute is per augmented input grid. Reference points retain their source protocols; details are in Appendices C.1 and C.3.

benchmarks further establish favorable accuracy–compute trade-offs against other compact looped Transformers.

2 Related Work

Looped Transformers and latent reasoning. Looped Transformers share parameters across depth, combining compact networks with repeated latent computation (Dehghani et al., 2019). Studies of their computational and learning capabilities motivate this design for reasoning (Giannou et al., 2023; Yang et al., 2024; Saunshi et al., 2025). Huginn and Ouro bring latent recurrence to pretrained language models (Geiping et al., 2025; Zhu et al., 2025), while SMELT studies looped mixture-of-experts models under matched compute and parameter budgets (Wang et al., 2026). HRM and TRM apply compact latent recurrence to structured reasoning (Wang et al., 2025; Jolicoeur-Martineau, 2025). We compare the same single-state backbone with and without LRFeed, separating feedback gains from changes to the recurrent structure.

Conditioning shared computation. Relaxed Recursive Transformers specialize shared layers through depth-specific low-rank adapters (Bae et al., 2025). Channel recalibration and feature-wise modulation provide general tools for conditioning activations (Hu et al., 2018; Perez et al., 2018; Peebles & Xie, 2023). In looped Transformers, timestep encodings modulate each iteration (Xu & Sato, 2025); LoopFormer adds step-size conditioning and shortcut-consistency training (Jeddi et al., 2026). RingFormer generates input-dependent level signals (Heo et al., 2025). SR^2 feeds latent estimates back into iterative refinement and aligns intermediate predictions (Deng et al., 2026). LRFeed instead derives its condition from a channel-wise profile of the change between consecutive latent states. Both the backbone and feedback parameters remain shared across iterations.

Scaling inference-time computation. Longer recurrences have been studied through progressive training (Bansal et al., 2022), stability constraints (Bear et al., 2024), and energy or equilibrium formulations (Du et al., 2022; Bai et al., 2019; Anil et al., 2022; Fein-Ashley & Rashidinejad, 2026). FPRM uses convergence-based halting, while EqR learns attractors and scales both depth and candidate count (Movahedi et al., 2026; Huang et al., 2026). Voting provides another route to exploiting

multiple candidates (Kubo et al., 2026). Lattice Deduction Transformers combine candidate-set refinement with stochastic search and conflict detection (Davis et al., 2026). LRFeed targets latent updates within a trajectory, without an explicit candidate-set search loop. Our fixed-budget rollouts test whether feedback improves the benefit of additional iterations, including beyond the training horizon.

Representation dynamics. Prior work analyzes token-rank degeneration in pure attention (Dong et al., 2021) and cross-iteration representation similarity in looped models (Jeddi et al., 2026). We examine temporal variation in channel-activity profiles and output probabilities. Effective rank describes how broadly this variation is distributed across directions, while continuation tests measure whether additional iterations solve previously unresolved puzzles.

3 Latent Residual Feedback

Our goal is to use recent latent changes to adapt shared computation, without introducing iteration-specific Transformer blocks. LRFeed extracts a compact profile of these changes and uses it to generate channel gates for the next transition. We first define the backbone, then explain the choice of feedback signal and how it enters the shared blocks.

3.1 A looped Transformer backbone

We build a single-state backbone for LRFeed. We denote the backbone alone as *LRFeed (w/o feedback)* and use it as our ablation baseline. It retains TRM's Transformer blocks (Jolicoeur-Martineau, 2025) but replaces its two interacting latent states with a single recurrent state (Appendix A.1). The resulting looped Transformer embeds the input, repeatedly updates this state with a shared module, and reads out an answer. Its input embedding and state update are

$$\begin{aligned} e(x) &= E(x), \\ z_{t+1} &= F_\theta(z_t + e(x)), \qquad z_0 = z_{\text{init}}. \end{aligned} \tag{1}$$

Here E combines token embeddings with auxiliary puzzle embeddings. Both $e(x)$ and z_t lie in $\mathbb{R}^{S\times D}$, where S is the total sequence length and D is the hidden dimension. The initial state z_{init} broadcasts a fixed vector across sequence positions. At each transition, we add the same input embedding and update the state using the same Transformer parameters. We call this one *atomic transition*, indexed by t.

A linear output head maps the state to token logits and predictions:

$$\ell_t = z_t W_{\text{out}}^\top, \qquad \widehat{y}_{t,s} = \arg\max_v \ell_{t,s,v}. \tag{2}$$

Here $W_{\text{out}} \in \mathbb{R}^{V\times D}$ is shared across iterations, V is the vocabulary size, and s ranges over answer positions, excluding the auxiliary puzzle-embedding positions. The readout does not replace the latent state; further iterations continue from that state.

The shared module F_θ consists of two post-norm Transformer blocks. Each block uses bidirectional self-attention and a SwiGLU feed-forward network, with RMS normalization after each residual addition (Zhang & Sennrich, 2019). The two blocks have distinct parameters, which are reused across iterations. We group ten atomic transitions into one *outer step* and read out an answer at its end. The final state carries into the next outer step.

3.2 Residual feedback

The current state represents the model's latest estimate, but does not explicitly separate what the preceding update changed. LRFeed makes this change available to the next transition. We retain the preceding state and take their difference:

$$\Delta z_t = z_t - z_{t-1}. \tag{3}$$

Any additive component common to the two states cancels. The residual therefore describes the latest change rather than the current representation or a position in a fixed iteration schedule.

A compact profile of change. The full residual has $S \times D$ entries. For channel-wise feedback, we summarize how strongly each channel changed across sequence positions:

$$p_{t,c} = \left(\frac{1}{S} \sum_{s=1}^{S} (\Delta z_{t,s,c})^2 \right)^{1/2}, \qquad \widehat{p}_t = \frac{p_t}{\sqrt{D^{-1} \sum_c p_{t,c}^2 + \epsilon}} \frac{1}{\sqrt{2}}, \tag{4}$$

where ϵ stabilizes normalization near zero. Squaring avoids cancellation between positive and negative changes. Normalization suppresses overall magnitude while retaining relative change strengths across channels. This yields a D-dimensional feedback signal; the full latent state remains available to the backbone.

Predictive information in recent changes. A held-out prediction test supports this choice. On LRFeed (w/o feedback), we predict the next atomic update's relative channel-energy profile using current state energy and the update's position within the outer step. Adding residual energy reduces test MSE by 24.86%. Shuffling residuals across puzzles at the same update position does not reproduce the gain (Section 4.2). This predictive signal exists in a backbone without feedback, motivating its use to condition subsequent computation.

We map the normalized profile to a feedback vector using a shared MLP:

$$h_t = f_\phi(\mathrm{stopgrad}(\widehat{p}_t)) \,. \tag{5}$$

The stop-gradient prevents this pathway from backpropagating through the residual history. The next transition then becomes

$$z_{t+1} = F_\theta(z_t + e(x); h_t) \,. \tag{6}$$

The preceding state serves only as a history buffer, not a second reasoning stream.

3.3 Channel Gates

The feedback should adapt the next transition while retaining its shared Transformer parameters. We therefore use h_t to generate channel gates inside each block. These gates adjust feature scale and shift, allowing the shared module to respond to recent changes without adding another Transformer block.

We apply this modulation at two sites in each block: after RMS normalization of the attention and MLP residual additions. Each site r has independent gate parameters. Let u^r be the feature at that site after its attention or MLP residual, and let $\bar{u}_c^r = \mathrm{rms}(u^r)_c = u_c^r / \sqrt{D^{-1} \sum_j (u_j^r)^2 + \epsilon}$. At transition t, the gate and affine modulation are

$$\begin{aligned} g_{t,c}^r &= \sigma(h_{t,c} a_c^r + q^r) \,, \\ y_c^r &= \bar{u}_c^r + \alpha g_{t,c}^r \, (\bar{u}_c^r w_c^r + b_c^r) \,, \qquad \alpha = 0.15. \end{aligned} \tag{7}$$

We omit the position and transition indices of u^r and y^r for clarity; the same gate is applied at every sequence position. Here a^r is the learned diagonal gate coefficient. The vectors w^r and b^r are learned per-channel modulation parameters. The complete pathway is

$$\Delta z_t \longrightarrow \widehat{p}_t \longrightarrow h_t \longrightarrow \{g_t^r\}_r \longrightarrow z_{t+1}. \tag{8}$$

Both Transformer and feedback parameters are shared across iterations. The channel gates can vary with recent latent changes, while the same backbone is reused throughout reasoning. Modulation acts on normalized features inside the blocks, not directly on the latent update $z_{t+1} - z_t$.

In our Sudoku implementation, feedback adds 0.53M parameters, increasing the total from 6.83M to 7.36M. This overhead is shared across all iterations and does not increase with inference depth.

Implementation details are provided in Appendix A.

4 Experiments

We compare accuracy and inference compute across Sudoku-Extreme, Maze-Hard, and ARC-AGI-1. Sudoku-Extreme supports sustained gains from deeper inference, even far beyond the training horizon (Appendix C.5). Maze-Hard and ARC-AGI-1 benefit less from additional depth. This makes Sudoku particularly useful for studying long-horizon reasoning. We therefore focus our residual-information analyses, matched-backbone ablations, and studies of reasoning dynamics on Sudoku.

Table 1: **Performance and inference compute across three benchmarks.** Accuracies are percentages; budgets are maximum matrix TFLOPs per puzzle or ARC augmented input. Our Sudoku results use the full test set. Scores for LRFeed with and without feedback on Sudoku and Maze are means ± sample standard deviations over three seeds. † marks reported results under their original protocols; ∼ marks digitized accuracy. ‡ marks SR^2's ARC metric ambiguity. N/R denotes an unavailable score-specific budget; a dash denotes no result. C-voting uses $K = 256$ for Sudoku and $K = 512$ for Maze. LLM rows are reported references, not compute-matched comparisons; ∗ marks ARC scores on the semi-private split. Reference protocols are in Appendix C.3.

	Sudoku-Extreme		Maze-Hard		ARC-AGI-1	
Method	Acc.	≈ TFLOPs	Acc.	≈ TFLOPs	Acc.	≈ TFLOPs
General-purpose LLMs (Wang et al., 2025; Davis et al., 2026)						
o3-mini-high	$0.0^{\dagger}$	N/R	$0.0^{\dagger}$	N/R	$34.5^{\dagger *}$	N/R
Claude 3.7 (8K)	$0.0^{\dagger}$	N/R	$0.0^{\dagger}$	N/R	$21.2^{\dagger *}$	N/R
DeepSeek-R1	$0.0^{\dagger}$	N/R	$0.0^{\dagger}$	N/R	$15.8^{\dagger *}$	N/R
Claude Opus 4.6	$0.0^{\dagger}$	N/R	$0.0^{\dagger}$	N/R	—	—
DeepSeek V4-Pro	$0.0^{\dagger}$	N/R	$0.0^{\dagger}$	N/R	—	—
GPT-5.4	$0.0^{\dagger}$	N/R	$0.0^{\dagger}$	N/R	—	—
Compact Looped Transformers						
HRM (Wang et al., 2025)	55.90	N/R	68.6	3.933	$40.3^{\dagger}$	2.997
TRM (Jolicoeur-Martineau, 2025)	75.63	1.829	82.9	5.350	$44.6^{\dagger}$	5.350
SR^2 (Deng et al., 2026)	$66.63^{\dagger}$	0.147	$93.7^{\dagger}$	1.998	$44.3^{\dagger\ddagger}$	1.998
ItrSA++ ($K = 1$) (Kubo et al., 2026)	$\sim 63.23^{\dagger}$	0.026	$\sim 76.7^{\dagger}$	0.942	—	—
+ C-voting	$\sim 93.33^{\dagger}$	6.675	$78.6^{\dagger}$	482.2	—	—
FPRM (Movahedi et al., 2026)	$94.2^{\dagger}$	47.654	$87.0^{\dagger}$	557.4	$47.5^{\dagger}$	15.927
SE-RRM (Freinschlag et al., 2026)	$93.73^{\dagger}$	N/R	$88.8^{\dagger}$	5.994	$45.3^{\dagger}$	12.001
Attractor (Fein-Ashley & Rashidinejad, 2026)	$91.4^{\dagger}$	N/R	$93.1^{\dagger}$	N/R	—	—
LRFeed (w/o feedback)	78.51 ± 1.22	1.742	82.77 ± 0.81	0.637	45.0	2.548
LRFeed (low budget)	91.36 ± 0.39	0.872	86.03 ± 0.57	0.318	45.1	0.318
LRFeed (extended)	97.07 ± 0.16	27.891	86.27 ± 0.51	0.637	46.9	2.548

4.1 Accuracy–compute trade-offs

We compare complete models to assess the accuracy they achieve for a given inference budget. We measure compute in matrix FLOPs; benchmark protocols and compute accounting are provided in Appendix B. TRM is our direct architectural reference because LRFeed retains its shared Transformer blocks. On all three tasks, LRFeed surpasses the corresponding TRM reference with less than one-eighth of its matrix compute (Appendix C.2).

Compared with the compact looped Transformers in Table 1, LRFeed offers a strong accuracy–compute trade-off across all three benchmarks. It achieves competitive accuracy with inference budgets below one TFLOP.

Figure 2a compares Sudoku accuracy as the inference-compute budget increases. Across the plotted budgets, LRFeed consistently achieves higher accuracy at comparable compute. Panels (b,c) show the corresponding budget comparisons on Maze-Hard and ARC-AGI-1.

4.2 What information do residual profiles provide?

We ask whether residual profiles provide useful information beyond a channel-wise summary of the current latent state. We assess this through the model's next update, which reflects how it has learned to revise its representation.

We train linear predictors of how the next update's energy (squared magnitude) is distributed across channels. All predictors include the update's position within the outer step, or *recurrent phase*. We compare state energy alone with state plus residual energy on held-out puzzles.

Table 2: Predicting the next atomic update's channel-energy allocation. Held-out MSE ($\times 10^{-7}$; lower is better). Every probe includes recurrent phase; the shuffled control is independently refitted.

Probe input	LRFeed (w/o feedback)	LRFeed
Phase only	4.660	5.405
State energy	2.868	3.753
State + residual energy	2.155	3.309
State + shuffled residual energy	2.928	3.784

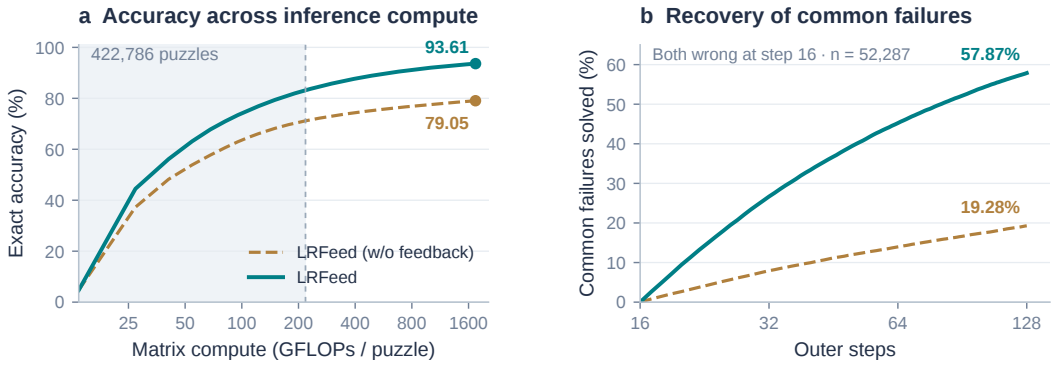


Figure 3: **Productive inference beyond the training budget.** (a) Accuracy at every endpoint of one continuous rollout over all 422,786 Sudoku test puzzles versus matrix compute; shading marks the training budget (16 outer steps, approximately 218 GFLOPs per puzzle). (b) Completion of the same 52,287 puzzles that both models miss at step 16. No subsampling or best-over-steps selection is used.

State energy improves prediction over recurrent phase alone (Table 2). Adding residual energy further reduces test MSE by 24.86% for LRFeed (w/o feedback) and 11.83% for LRFeed. This additional gain disappears when residuals are shuffled across puzzles at the same phase and predictors are refitted with the same capacity.

These results show that residual profiles provide additional information about how the next update is distributed across channels. Conditioning only on pooled state features may therefore miss useful cues for subsequent computation. This motivates supplementing the current state with residual feedback, giving the shared transition direct access to these cues. Details and further controls are in Appendix D.

4.3 Accuracy and Inference-Time Scaling

We next test how residual feedback affects task performance by comparing LRFeed with LRFeed (w/o feedback) on Sudoku. The latter uses the same Transformer backbone without the feedback pathway. Both models use the same training data, optimization settings, number of optimizer updates, and training-time halting policy. We evaluate them with fixed-depth rollouts on the same test puzzles.

Adding residual feedback improves accuracy across the evaluated compute budgets (Figure 3a). The advantage is already visible at the training limit and grows as inference continues beyond it. Timing measurements also show higher accuracy at shorter inference times (Appendix C.6).

To examine these gains beyond the training horizon, we track the same puzzles that both models fail to solve at the training limit of 16 outer steps. Figure 3b shows that LRFeed solves a substantially larger fraction of these puzzles as computation increases. This comparison focuses on new solutions after the training limit, rather than answers already correct at that limit.

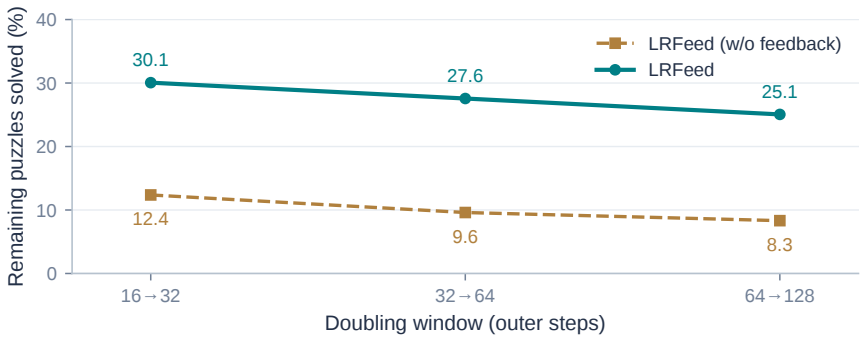


Figure 4: **Solving remaining puzzles with additional steps.** Fraction of each model's remaining puzzles solved over doubling windows on the complete Sudoku test set (Equation 9). Denominators are model-specific, and later windows contain more steps. A puzzle counts as solved if its answer is correct at the window's end.

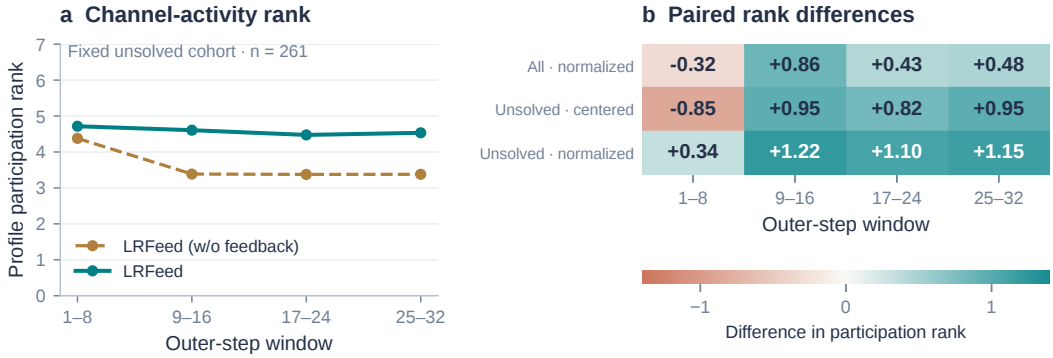


Figure 5: **Effective rank of channel-activity profiles.** (a) Mean effective rank (participation rank) of temporally centered, row-normalized state RMS profiles on the fixed 261 puzzles unsolved by both models throughout 33 steps. Each window contains eight outer steps; the maximum rank is seven. (b) Paired rank differences on all 3,072 puzzles and on the fixed cohort, including the centered-only normalization control. The first-window contrast depends on normalization and cohort; later contrasts are positive.

Figure 4 further measures the fraction of remaining puzzles solved each time the inference budget doubles. Let $C_m(t)$ indicate whether model m answers a puzzle correctly at step t. For a window $(a, b]$, we measure

$$q_m(a, b) = \Pr(C_m(b) = 1 \mid C_m(a) = 0)\,. \tag{9}$$

Unlike the fixed common-failure cohort in Figure 3b, the remaining set is specific to each model and updated at every window start. LRFeed consistently solves a larger share of that set, showing that its gains extend to continued problem solving beyond the training horizon.

These gains persist at much greater depths: the same LRFeed checkpoint continues to improve during a single rollout to 2,048 outer steps, 128 times the training limit (Appendix C.5).

4.4 Observations on Reasoning Dynamics

We next look for patterns in internal activity that may help understand the different gains from continued inference. We focus on how channel activity and answer distributions vary across iterations.

At each outer step t, the state RMS profile $s_{t,c} = \sqrt{S^{-1}\sum_i z_{t,i,c}^2}$ measures activation strength in each channel. Within each eight-step window, we subtract the mean profile and normalize each centered profile to unit length. Let X contain these profiles as rows and $G = XX^\top$. We measure

their effective rank using the participation rank

$$r_{\mathrm{PR}}(G) = \frac{\mathrm{tr}(G)^2}{\mathrm{tr}(G^2)}. \tag{10}$$

A low rank means that variation in channel activity is concentrated in a few directions; a higher rank means that it is spread more evenly across directions.

Figure 5 reveals a difference on the fixed set of puzzles that neither model solves throughout inference. Without feedback, mean rank drops from 4.38 in the first window to about 3.38 and stays low. LRFeed instead maintains a higher rank of about 4.5 in the later windows, including those beyond the training horizon. On these unresolved puzzles, we therefore observe more concentrated activity patterns without feedback and more varied activity with LRFeed. In a channel-level decomposition, most of the rank gap corresponds to less correlated channel trajectories, rather than simply more active channels (Appendix E.4).

Does the difference reach the readout? On a disjoint test subset, we apply the same rank measure to answer probabilities on puzzles both models still answer incorrectly. LRFeed also has higher output-profile rank, including at steps 121–128, even though its output changes are smaller in magnitude.

Together, these observations show that stronger continuation gains coincide with more varied changes in latent activity and answer distributions. This suggests a possible link between sustained activity diversity and the usefulness of additional reasoning steps. Measurement details, normalization controls, and replication results are provided in Appendix E.

5 Discussion and Limitations

Feedback for shared computation. Parameter sharing need not imply fixed feature processing at every iteration. LRFeed uses recent latent changes to condition channel gates while retaining the same Transformer parameters. This suggests a design direction beyond choosing how many iterations to run: making information from preceding updates explicitly available to shared computation.

Different tasks benefit at different depths. Sudoku highlights the value of continued inference, while Maze and ARC show benefits at smaller budgets. Both low-budget accuracy and gains from additional computation therefore matter when evaluating a looped model.

Scope and limitations. Our experiments focus on compact models for structured reasoning. Whether residual feedback offers similar benefits in large pretrained language models remains untested. The current feedback also summarizes only the latest update: pooling discards position-specific changes and their signs, while normalization suppresses overall magnitude. Retaining some of this information or a longer update history may provide richer feedback, but would require balancing its value against additional cost.

6 Conclusion

We introduced LRFeed, which uses changes between consecutive latent states to condition shared computation in Looped Transformers. Across three structured reasoning benchmarks, LRFeed achieves a favorable accuracy–compute trade-off. Matched-backbone ablations show stronger gains from continued inference, including beyond the training horizon on Sudoku. These results highlight latent changes as a useful source of feedback for improving weight-shared reasoning.

Reproducibility statement

Section 3 defines the model, with implementation details in Appendix A. Training, evaluation, and compute accounting are specified in Appendix B. Appendix C provides supplementary performance results. Residual-information analyses and dynamic-profile analyses are documented in Appendices D and E, respectively.

AI use statement

Generative AI tools assisted with manuscript drafting and language polishing, literature retrieval and discovery, research ideation, model and evaluation code, analysis of experiment outputs, and figure preparation. They also assisted in developing mathematical derivations and proofs for the residual-information and effective-rank analyses (Appendices D and E.4). The authors are responsible for the final text, code, analyses, and mathematical claims.

Ethics statement

We adhere to the ICLR Code of Ethics. This work studies latent recurrent reasoning on established Sudoku-Extreme, Maze-Hard, and ARC-AGI-1 benchmarks. It does not involve the collection of personal or sensitive user data or studies with human participants. Improved reasoning capabilities may have dual-use implications when incorporated into broader AI systems. Any deployment beyond these benchmarks should include task-appropriate validation and safeguards, with human or domain-expert oversight in high-stakes settings.

References

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. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/331c41353b053683e17f7c88a797701d-Abstract-Conference.html.

Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/54d6a55225cebbdc16fbb0e45c5bdf2b-Paper-Conference.pdf.

Junyeob Baek, Mingyu Jo, Minsu Kim, Mengye Ren, Yoshua Bengio, and Sungjin Ahn. Generative recursive reasoning, 2026. URL https://arxiv.org/abs/2605.19376.

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

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. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/7f70331dbe58ad59d83941dfa7d975aa-Abstract-Conference.html.

Jay Bear, Adam Prügel-Bennett, and Jonathon Hare. Rethinking deep thinking: Stable learning of algorithms using lipschitz constraints. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/afbe068bd0469f4cd778c0f8106181b6-Abstract-Conference.html.

John J. Benedetto and Matthew Fickus. Finite normalized tight frames. *Advances in Computational Mathematics*, 18:357–385, 2003. doi: 10.1023/A:1021323312367. URL https://link.springer.com/article/10.1023/A:1021323312367.

Yilong Chen, Zitian Gao, Yihao Xiao, Jason Klein Liu, Xinyu Yang, Yifan Luo, Haoming Luo, Zhengmao Ye, Tingwen Liu, Ran Tao, and Bryan Dai. Polymorphic universal transformer. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 39001–39013, 2026. doi: 10.18653/v1/2026.acl-long.1809. URL https://aclanthology.org/2026.acl-long.1809/.

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

Liam Davis, Leopold Haller, Alberto Alfarano, and Mark Santolucito. Lattice deduction transformers, 2026. URL https://arxiv.org/abs/2605.08605.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://research.google/pubs/universal-transformers/.

Yunlong Deng, Boyang Sun, Yan Li, Zeyu Tang, Lingjing Kong, Kun Zhang, and Guangyi Chen. Selection, reflection and self-refinement: Revisit reasoning tasks via a causal lens. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/879d773f3321bcea7fa72b77b069ec5e-Abstract-Conference.html.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: pure attention loses rank doubly exponentially with depth. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 2793–2803, 2021. URL https://proceedings.mlr.press/v139/dong21a.html.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Learning iterative reasoning through energy minimization. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 5570–5582, 2022. URL https://proceedings.mlr.press/v162/du22d.html.

Jacob Fein-Ashley and Paria Rashidinejad. Solve the loop: Attractor models for language and reasoning, 2026. URL https://arxiv.org/abs/2605.12466.

Richard Freinschlag, Timo Bertram, Erich Kobler, Andreas Mayr, and Günter Klambauer. Symbol-equivariant recurrent reasoning models, 2026. URL https://arxiv.org/abs/2603.02193.

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

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

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

Jaemu Heo, Eldor Fozilov, Hyunmin Song, and Taehwan Kim. RingFormer: Rethinking recurrent transformer with adaptive level signals, 2025. URL https://arxiv.org/abs/2502.13181.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Hu_Squeeze-and-Excitation_Networks_CVPR_2018_paper.html.

Benhao Huang, Zhengyang Geng, and Zico Kolter. Equilibrium reasoners: Learning attractors enables scalable reasoning, 2026. URL https://arxiv.org/abs/2605.21488.

Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. LoopFormer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10009450.

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

Kenji Kubo, Shunsuke Kamiya, Masanori Koyama, Kohei Hayashi, Yusuke Iwasawa, and Yutaka Matsuo. C-Voting: Confidence-based test-time voting without explicit energy functions. In *International Conference on Learning Representations*, 2026. URL https://iclr.cc/virtual/2026/poster/10007778.

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, 2026. URL https://arxiv.org/abs/2606.18206.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Peebles_Scalable_Diffusion_Models_with_Transformers_ICCV_2023_paper.html.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11671. URL https://ojs.aaai.org/index.php/AAAI/article/view/11671.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped transformers. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2502.17416.

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

Shaowen Wang, Ge Zhang, Kairong Luo, Yuhao Wu, Shaofan Liu, Jiaheng Liu, Wenhao Huang, Shen Yan, and Jian Li. SMELT: Scaling laws for compute-matched MoE looped transformers, 2026. URL https://arxiv.org/abs/2609.01343.

Kevin Xu and Issei Sato. On expressive power of looped transformers: Theoretical analysis and enhancement via timestep encoding. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 69613–69646, 2025. URL https://proceedings.mlr.press/v267/xu25x.html.

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

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

Rui-Jie Zhu, Zixuan Wang, Kai Hua, Tianyu Zhang, Ziniu Li, Haoran Que, Boyi Wei, Zixin Wen, Fan Yin, He Xing, Lu Li, Jiajun Shi, Kaijing Ma, Shanda Li, Taylor Kergan, Andrew Smith, Xingwei Qu, Mude Hui, Bohong Wu, Qiyang Min, Hongzhi Huang, Xun Zhou, Wei Ye, Jiaheng Liu, Jian Yang, Yunfeng Shi, Chenghua Lin, Enduo Zhao, Tianle Cai, Ge Zhang, Wenhao Huang, Yoshua Bengio, and Jason Eshraghian. Scaling latent reasoning via looped language models, 2025. URL https://arxiv.org/abs/2510.25741.

A Model and implementation details

A.1 Why use a single-state backbone?

A matched looped Transformer. The single-state backbone repeatedly applies shared Transformer blocks to one state, replacing TRM's separate reasoning and answer states (Jolicoeur-Martineau, 2025). TRM provides an external performance comparison; LRFeed (w/o feedback) provides the matched-backbone ablation. The latter comparison separates the feedback benefit from the change in recurrent structure.

Alternating versus single-state updates. TRM updates its reasoning state several times while holding its answer state fixed, then updates the answer state. The single-state backbone updates the same state at every internal step, removing this alternation.

For joint state $Z = (z^L, z^H)$, an update that holds z^H fixed has

$$\Phi_L(Z) = \big(f(z^L, z^H, x), z^H\big), \qquad J_L = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix}, \tag{11}$$

where $A = \partial f/\partial z^L$ and $B = \partial f/\partial z^H$. The identity block gives $\rho(J_L) \geq 1$ and $\|J_L\|_2 \geq 1$, so this substep cannot strictly contract the full joint state. This does not determine stability over a complete cycle. The single-state backbone removes the frozen-state constraint without imposing convergence.

Local spectral behavior. In Figure 6, the mean for LRFeed (w/o feedback) falls below one after the early updates, whereas the TRM mean remains close to or above one. These are local diagnostics: stability over many updates depends on the product of their Jacobians.

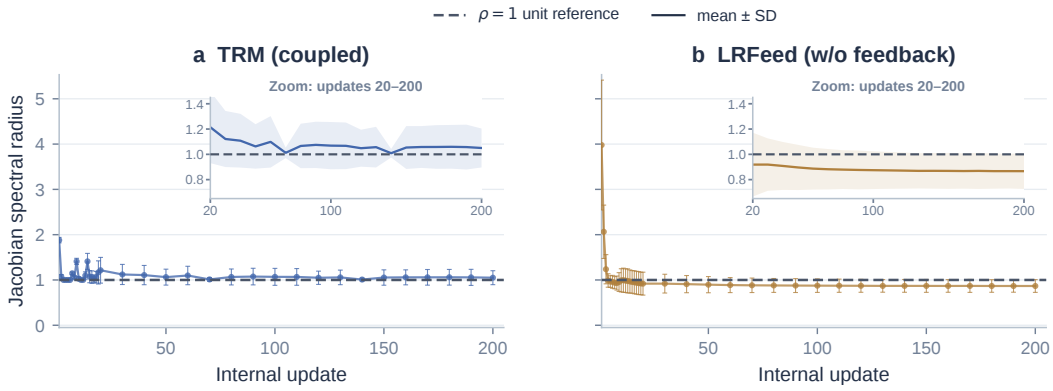


Figure 6: **Local spectral behavior of TRM and LRFeed (w/o feedback).** Mean estimated spectral radius at 38 internal update indices: steps 1–20, followed by every tenth step through 200. Error bars show standard deviations. The dark dashed line marks a spectral radius of one. Both panels use the same axes; insets show updates 20–200 at a common magnified vertical scale of 0.65–1.45.

Variation in update magnitude. Figure 7 tracks $\Delta_t = \|z_{t+1} - z_t\|_2$. TRM shows repeated short-period fluctuations, including on correctly solved puzzles. LRFeed (w/o feedback) is smoother: correct trajectories reach smaller updates, while incorrect trajectories retain larger updates. These are fluctuations in update magnitude, not evidence of a periodic orbit in the full latent state.

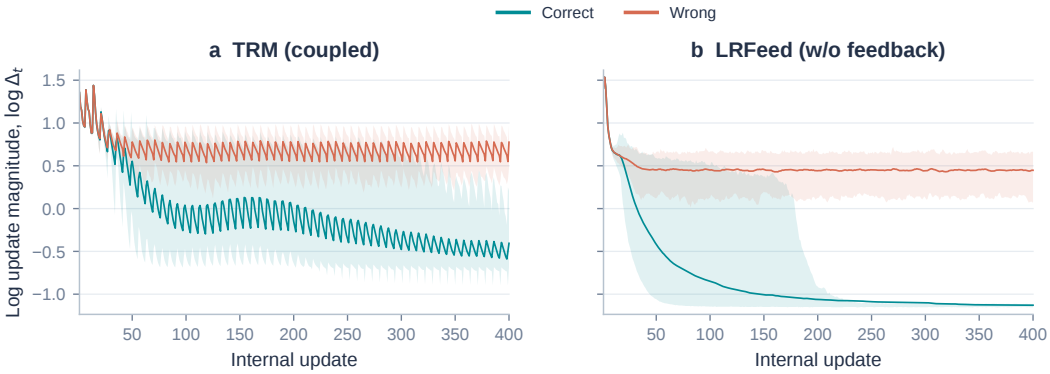


Figure 7: **Latent update magnitudes over 400 internal updates.** Curves show the mean log update magnitude for correctly and incorrectly solved puzzles in TRM and LRFeed (w/o feedback). Shaded bands show p05–p95 bounds. Both panels use the same axes. The horizontal axis counts internal updates, not the outer steps used for the main accuracy curves.

These observations motivate the single-state backbone. LRFeed adds residual feedback to this same shared transition, so their comparison isolates the feedback pathway from the choice of backbone.

A.2 Implementation details

Backbone. The backbone contains two Transformer blocks with width $D = 512$, eight attention heads, rotary positional embeddings, and SwiGLU intermediate width 1,536. The same two blocks are reused at every atomic transition. For Sudoku, 81 grid tokens and 16 puzzle-embedding positions give $S = 97$. We set $L_{\text{cycles}} = 9$ and $H_{\text{cycles}} = 1$, yielding ten atomic transitions per outer step.

Residual history and normalization. For a new puzzle, we initialize the current and preceding states to the same value, making the first residual zero. Each atomic transition saves the latent states before and after the update; their difference supplies the feedback for the next transition. We retain both states across outer-step boundaries, so the feedback history is not reset between outer steps. Both states are detached at each boundary, preventing gradients from propagating into the preceding outer step. The RMS profile in Equation 4 uses $\epsilon = 10^{-5}$ and the fixed factor $1/\sqrt{2}$. Squaring removes the residual's sign, and normalization suppresses its overall magnitude. We detach the normalized profile before passing it to the feedback MLP.

Feedback MLP and gate initialization. The feedback MLP maps the normalized profile to a D-dimensional vector using two 512-to-512 linear layers:

$$f_\phi(v) = W_2 \,\text{Dropout}_{0.05}\,[\text{SiLU}(W_1 v + b_1)] + b_2. \tag{12}$$

Dropout is active only during training. We initialize W_2 and each modulation vector w^r from a truncated normal with standard deviation 0.02 and bounds $[-0.04, 0.04]$. Their biases are zero: $b_2 = 0$ and $b^r = 0$. Each gate bias starts at $q^r = -2$. The diagonal gate coefficient a^r uses a truncated normal with standard deviation 0.001 and bounds $[-0.002, 0.002]$. It is a $[1, D]$ vector applied elementwise, not a dense projection. Gate logits use a scale factor of 1.

Supervision and halting. Within each outer step, the first two atomic transitions run without gradient tracking; the remaining eight retain gradients. We supervise the final prediction of each outer step with token-level Stablemax cross-entropy and a binary halting loss:

$$\mathcal{L} = \mathcal{L}_{\text{stablemax}} + 0.5\mathcal{L}_{\text{halt}}. \tag{13}$$

The halting target is one when the entire predicted puzzle is correct and zero otherwise. With `no_ACT_continue=true`, we use no separate continuation loss. During training, a sample stops when its halt logit is positive and it has completed its assigned minimum number of outer steps (Appendix B.1). Every sample stops by step 16. Gradients from the halting head to the latent state are multiplied by 0.7, without changing the forward logits. The stopping decisions themselves are not differentiated.

B EXPERIMENTAL PROTOCOLS AND COMPUTE ACCOUNTING

B.1 EVALUATION AND TRAINING PROTOCOL

Data and execution. We use 1,000 training puzzles, each retained in its original form and augmented 1,000 times, yielding 1,001,000 training examples. The TRM preprocessing applies digit relabeling, optional transposition, and valid row-band, within-band, column-stack, and within-stack permutations. The generated dataset is fixed across controlled runs; test puzzles are not augmented. The primary full-test evaluation covers 422,786 puzzles through 128 steps at batch size 128. Paired ablation analyses use 3,072 puzzles, with a separate 512-puzzle replication set. The primary models use identical ordered inputs, EMA weights, batch size 32, eager bfloat16 inference, and forced fixed-depth continuation. Each subset rollout records all 512 outer-step endpoints. Batch shape can affect finite-precision execution, so the two protocols need not be numerically identical.

Optimization. The matched Sudoku runs use global batch size 768 and four-GPU DDP. Dense parameters use AdamATan2 with betas $(0.9, 0.95)$; puzzle embeddings use distributed sparse sign SGD. Both use learning rate 10^{-4} and weight decay 1.0, with a 2,000-update warmup followed by a constant learning rate. EMA decay is 0.999. A training step is an optimizer update, not a completed puzzle episode. Matching this progress aligns backbone calls, but architectures with additional modules can still have different total training FLOPs.

Training seeds. For LRFeed with and without feedback, Table 1 reports means and sample standard deviations over training seeds 0, 1, and 2 on Sudoku and Maze. The detailed Sudoku curves and trajectory analyses use the matched seed-0 models.

Training-time step floor. Matched Sudoku runs use a maximum of 16 outer steps. Each new sample has a persistent minimum-step floor: with probability 0.25 it is sampled uniformly from $\{3, \dots, 16\}$, and otherwise it is 3. The floor stays fixed until that sample halts. Evaluation disables this exploration and records fixed-depth endpoints after each outer step.

Uncertainty and endpoint correctness. Subset accuracy differences use paired bootstrap intervals with 5,000 resamples of whole puzzles jointly across models. The atomic prediction probe also uses 5,000 held-out-puzzle resamples. Paired gain bands in Figure 10a are pointwise normal intervals (mean paired correctness difference ± 1.96 standard errors), not simultaneous bands. Full-test curves are exact benchmark scores without sampling intervals; all reported accuracies are endpoint correctness at the stated depth.

B.2 COMPUTE ACCOUNTING

Parameter count. LRFeed (w/o feedback) has 6,828,034 learned parameters; LRFeed has 7,359,494. The additional 531,460 parameters comprise the conditioning MLP and four gate/modulation sites: $2D^2 + 2D + 4(3D + 1)$ for $D = 512$. The constant-input control has the same parameter count as LRFeed. These counts include the shared puzzle embedding and output heads.

Matrix operations. We count a multiply–add as two operations. For a dense attention block with sequence length S, width D, and SwiGLU intermediate width I, the dominant matrix cost is

$$C_{\text{block}} = 8SD^2 + 4S^2D + 6SDI. \tag{14}$$

The three terms cover attention projections, attention matrix products, and feed-forward matrices. We multiply by the actual block count and atomic calls. The conditioning MLP adds $4D^2$ operations per atomic call; the accounting also includes a small $2D$ gate approximation. Embeddings, output heads, RoPE, normalization, pooling, other elementwise operations, and runtime overhead are excluded. The constant-input control still evaluates its MLP.

This proxy gives 13.608, 13.618, 28.577, 25.631, and 13.744 GFLOPs per outer step for LRFeed (w/o feedback), LRFeed, TRM, EqR, and HRM. Thus equal outer depth does not imply equal compute across architectures. LRFeed at 16 steps uses 0.1251 times the estimated cost of LRFeed (w/o

feedback) at 128 steps. Budget comparisons charge actual integer-depth computation, not unused budget; breadth charges every candidate trajectory. These estimates are not latency measurements. FPRM uses two Transformer blocks over 97 tokens and a depthwise 3×3 convolution over the 81 grid tokens. Counting both attention products and the convolution gives 1.361540096 GFLOPs per fixed-point update, excluding the output head as for the other models.

C SUPPLEMENTARY PERFORMANCE RESULTS

C.1 EXTERNAL-CHECKPOINT AND REPORTED-RESULT COMPARISONS

Figure 2a compares single-trajectory checkpoint evaluations over the displayed compute range, alongside reported SR^2 and C-voting reference points. The protocols for these comparisons are given below.

The displayed LRFeed and FPRM curves start near 109 GFLOPs and follow an approximately doubling budget grid up to the cost of 128 LRFeed outer steps, matching the full-test ablation horizon. EqR retains its native inference noise and FPRM its random initialization. Their curves average three inference seeds, each using one trajectory. ItrSA++ accuracies are digitized from the original figure, as detailed below.

SR^2 reported reference. The star in Figure 2a shows the published Sudoku-Extreme accuracy of 66.63% (Deng et al., 2026), not a measurement on our analysis cohort. We estimate matrix compute from the public implementation: 82 tokens, width 512, SwiGLU intermediate width 1,536, and one shared Transformer block. Each supervision step contains 16 block calls; the evaluation terminates after 16 supervision steps. The resulting 256 calls cost $256 \times 0.57266176 = 146.60141056$ GFLOPs under our accounting convention. The paper reports peak evaluation accuracy; without its checkpoint, we cannot verify the exact score-to-configuration correspondence.

ItrSA++ with C-voting reported references. The hollow hexagons use accuracies digitized from Figure 3(b) of Kubo et al. (2026), which evaluates 1,000 Sudoku-Extreme puzzles at $T = 32$ iterations per candidate. C-voting selects the candidate with the largest mean top-1 token confidence. We digitize six points at $K \in \{2, 4, 8, 16, 32, 64\}$ candidates, with approximate accuracies of 68.5, 73.3, 77.6, 81.6, 85.2, and 88.4%. The main figure starts at $K = 4$; the line connects reported points. Digitization tolerance is approximately 0.2 percentage points, not a statistical confidence interval. The reported architecture has width $D = 384$, four self-attention calls, one cross-attention call, and one SwiGLU per iteration. For an illustrative matrix-compute estimate, we use $N = 81$ tokens, assume SwiGLU intermediate width $I = 4D$, and do not cache cross-attention keys or values. This gives

$$C_{\text{step}} = 5(8ND^2 + 4N^2D) + 6NDI \approx 0.8148 \text{ GFLOPs}, \qquad C_{\text{total}} = KTC_{\text{step}}.$$

The corresponding budgets are approximately 52, 104, 209, 417, 834, and 1,669 GFLOPs. The source does not specify the exact SwiGLU width or optimized execution; this estimate excludes geometry-aware positional transformations, readout, and elementwise operations. These reference points are not checkpoint evaluations on our common cohort.

HRM. We evaluate the official Sudoku-Extreme checkpoint on the same 3,072 puzzles, retaining both latent states across outer steps. The released configuration uses two high-level cycles, two low-level cycles, and four Transformer layers in each module. Each outer step executes four low-level and two high-level module calls, or 24 Transformer blocks. It uses 82 tokens, including one puzzle token. We bypass the ACT wrapper for the fixed-depth rollout, as in the other depth curves. At the seven shared compute caps, HRM uses 3, 7, 15, 32, 63, 126, and 237 outer steps and achieves 24.02%, 47.04%, 55.05%, 59.02%, 60.45%, 61.88%, and 63.02%, respectively.

FPRM. We use the released Sudoku EMA weights at step 78,120 and the model file distributed with them. Shared layers and solver code come from the official `dev/training` release; all checkpoint parameters and buffers load strictly. We use the model-card inference settings (damping decay 0.997, patience 10), retaining the native random initialization. We average seeds 0, 1, and 2 without voting or selecting a seed. The shared caps allow 40, 80, 160, 330, 640, 1280, and 2400 fixed-point updates. Mean accuracies are 63.34%, 70.49%, 75.00%, 79.72%, 84.87%, 88.91%, and 90.61%. Each seed uses one continuous rollout with persistent solver state. Native whole-batch stopping readouts match the fixed-depth readouts at these caps. These are capped evaluations, not a reproduction of the model card's 35,000-update endpoint.

EqR. We evaluate the released EqR EMA checkpoint at training step 50,000 with its original inference noise level 0.5. Table 3 uses the same 3,072-puzzle analysis subset and averages inference

seeds 0, 1, and 2. There is no cross-seed voting or best-seed selection. Training data processing, architecture, and optimization are not matched to LRFeed. The compute ledger uses each implementation's actual recurrence: 21 atomic calls per EqR outer step versus ten for LRFeed.

Table 3: Single-trajectory exact accuracy (%) under common estimated matrix-compute caps. Integer depths need not exhaust the cap. Difference intervals pair puzzles and average EqR inference seeds.

LRFeed steps	EqR steps	LRFeed	EqR	Difference [95% CI]
4	2	63.12	38.00	25.12 [23.58, 26.74]
8	4	75.16	55.41	19.75 [18.26, 21.27]
16	8	82.49	67.76	14.72 [13.33, 16.08]
33	17	87.83	77.96	9.86 [8.54, 11.17]
64	34	90.89	85.57	5.32 [4.11, 6.52]
128	68	93.13	90.58	2.55 [1.51, 3.67]
240	127	94.92	93.86	1.06 [0.14, 1.96]

C.2 MULTI-BUDGET BENCHMARK CURVES

We collect the complete budget sweeps used to form the main comparison. The horizontal coordinate is the number of executed atomic module calls; the tables also report the corresponding matrix-compute budget. Sudoku numbers below use the full 422,786-puzzle test set. The Maze and ARC tables use the same evaluation endpoints as the curves in Figure 8.

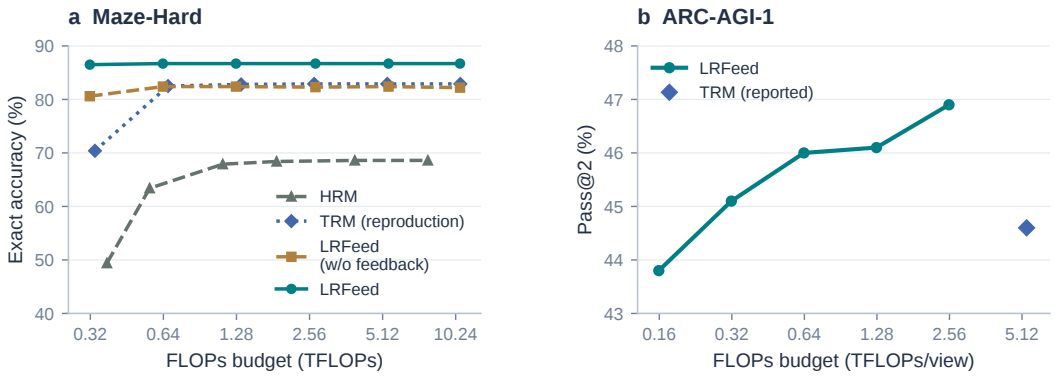


Figure 8: **Inference budgets beyond Sudoku.** (a) Maze exact accuracy versus matrix-FLOPs budget. (b) ARC accuracy versus budget per augmented input. The isolated TRM point is its reported 44.6% at 5.350 TFLOPs, not a reproduced curve. Lines connect evaluated endpoints only; they do not imply that the architectures use identical integer iteration counts.

Table 4: Sudoku-Extreme exact accuracy (%) on all 422,786 test puzzles. Each model uses the largest complete outer-step budget below the atomic cap. TRM uses 21 atomic calls per outer step; LRFeed uses 10.

Atomic budget	84	168	336	672
TRM	57.77	66.53	70.82	73.57
LRFeed	75.34	83.18	88.42	91.66

Table 5: Full-test Sudoku-Extreme accuracy at common matrix-compute budgets at batch size 128. Each row is evaluated by one continuous rollout; no candidate selection is used. Depth is counted in atomic transitions: each outer step contains 21 transitions for TRM and 10 for LRFeed.

Budget (≈ TFLOPs/puzzle)	0.218	0.436	0.872	1.743	9.288
TRM atomic transitions	147	315	630	1260	6825
LRFeed atomic transitions	160	320	640	1280	6820
TRM accuracy (%)	65.33	70.50	73.36	75.44	79.09
LRFeed accuracy (%)	83.18	88.24	91.48	93.61	96.32

On the complete Sudoku test set, LRFeed at 16 outer steps reaches 83.18%, exceeding TRM's 75.63% at 64 steps while using 11.9% of its matrix compute. At a common cap of 1.743 TFLOPs per puzzle, their accuracies are 93.61% and 75.44%, a difference of 18.17 percentage points (Table 5). Appendix C.5 reports the extended-budget comparison.

C.3 CROSS-BENCHMARK REFERENCE PROTOCOLS

Budget convention. Table 1 counts the configured maximum matrix computation. Fixed-depth models are charged for their complete rollout; adaptive solvers are charged at their iteration cap, without an early-stopping discount. Multi-candidate inference charges all candidates. We count a multiply–add as two FLOPs and use the matrix-operation scope in Appendix B.2. These budgets are not measured runtime. Figure 2(b,c) visualizes the selected Maze-Hard and ARC-AGI-1 budgets. The LRFeed points, with and without feedback, use seed 0; Table 1 additionally aggregates three

Table 6: Maze-Hard accuracy at the tested atomic budgets. The reported matrix costs use the nearest complete outer iteration for each reference budget.

Atomic calls	21	42	84	168	336	672
TRM accuracy (%)	70.4	82.5	82.8	82.9	82.9	82.9
TRM cost (≈ TFLOPs)	0.334	0.669	1.338	2.675	5.350	10.700
LRFeed accuracy (%)	86.5	86.7	86.7	86.7	86.7	86.7
LRFeed cost (≈ TFLOPs)	0.318	0.637	1.274	2.707	5.414	10.669

Table 7: ARC-AGI-1 accuracy at the tested LRFeed budgets. The TRM entry is a single reported reference point rather than a reproduced sweep; † marks the reported value.

Atomic transitions	10	20	40	80	160	336
LRFeed accuracy (%)	43.8	45.1	46.0	46.1	46.9	—
LRFeed cost (≈ TFLOPs)	0.159	0.318	0.637	1.274	2.548	—
TRM accuracy (%)	—	—	—	—	—	44.6†
TRM cost (≈ TFLOPs)	—	—	—	—	—	5.350

training seeds on Maze. Open markers denote published references and filled markers denote our evaluations. The upper-left shading indicates the preferred direction: lower compute and higher accuracy.

Backbone budgets and curves. TRM, LRFeed (w/o feedback), and LRFeed use two-block modules of width 512, with SwiGLU width 1,536 and $900 + 16 = 916$ sequence positions on these tasks. One block costs 7.961608192 GFLOPs. TRM uses 21 module applications per outer step, whereas LRFeed (w/o feedback) and LRFeed use ten. LRFeed additionally evaluates a two-layer conditioning MLP, costing $4(512)^2$ FLOPs per module application. Thus four LRFeed outer steps cost 0.636971 TFLOPs, and 16 TRM outer steps cost 5.350201 TFLOPs. The ratio is 0.119055. For the Maze curves, the original reference grid is $21, 42, 84, 168, 336, 672$ module applications; LRFeed (w/o feedback) and LRFeed use the nearest complete outer iteration. Their plotted costs use the corresponding rounded depth, rather than treating a reference label as an exact executed count. HRM uses 2, 3, 6, 10, 21, and 42 outer steps at the six reference points. Its released Maze configuration has 901 sequence positions and 24 block applications per outer step, costing 0.187285 TFLOPs per step. The table uses 21 steps, the first tested depth reaching 68.6%, at 3.932991 TFLOPs; the curve also includes 42 steps at 7.865982 TFLOPs. Maze HRM and TRM scores are our evaluated results, not their original papers' scores.

ARC depth and voting. The LRFeed curve is 43.8, 45.1, 46.0, 46.1, and 46.9% at 1, 2, 4, 8, and 16 outer steps, respectively. We use the same test-augmentation and voting protocol as TRM, reporting two final answers per test input. LRFeed (w/o feedback) reaches 45.0% at 16 outer steps; with ten module applications per outer step and two blocks per application, this costs 2.547715 TFLOPs per augmented input. ARC budgets are per augmented input grid, not per selected answer. If a query evaluates A views, its prediction-pool budget is A times the per-view cost; pass@2 does not imply $A = 2$. The native HRM/TRM protocol describes roughly 1,000 test-time views. The released SR^2 builder requests 1,000 augmentations in addition to the original view, with duplicates removed. FPRM uses the augmented ARC build; SE-RRM's stated eight dihedral augmentations concern training, not an established test-time view count. We therefore do not infer a common whole-task budget for every reported ARC score.

HRM's released configuration uses one puzzle token, giving 901 positions, and executes 24 Transformer-block applications per outer step. At the 16-step cap, $384C_{\text{block}}(901, 512, 1536)$ gives 2.996565 TFLOPs per augmented input. Its 40.3% and TRM's 44.6% are reported ARC results (Wang et al., 2025; Jolicoeur-Martineau, 2025).

SR^2. The reported Maze-Hard and ARC-1 scores are 93.7% and 44.3% (Deng et al., 2026). Both released recipes have a 256-shared-block maximum, with 901 positions, width 512, and actual SwiGLU width 1,536. This gives 1.997710 TFLOPs per input. Maze uses 16 refinement updates per

supervision segment and 16 segments; the separate ARC implementation repeats its shared block 16 times in each of 16 ACT segments. The paper's main table labels its metrics pass@1, while its ARC benchmark description specifies two attempts and pass@2. We mark this metric ambiguity with ‡ beside the published 44.3%.

ItrSA++ and C-voting. Figure 3(c) of Kubo et al. (2026) reports Maze-Hard results for $K = 1$ through 512 candidates at 64 iterations per candidate. The single-candidate score is approximately 76.7%, read from the vector figure with 0.2-point precision; the reported 512-candidate endpoint is 78.6%. We use 900 positions and width 384. Each iteration has four self-attention applications, one input cross-attention, and one SwiGLU. With feed-forward width 1,536, as in our Sudoku estimate, its cost is

$$C_{\mathrm{ItrSA}}(K) = 64K\left[5(8SD^2 + 4S^2D) + 6SDI\right]. \tag{15}$$

The resulting budgets are 0.941713 and 482.157055 TFLOPs at $K = 1$ and 512. The paper does not specify I; a sensitivity calculation using $I = 256$ gives 0.771844 and 395.183967 TFLOPs. These are architectural assumptions, not confidence intervals. Cross-attention key/value projections are charged each iteration. No ARC result is reported.

FPRM. The released model recipes use two width-512 blocks over 916 positions, plus a depth-wise width-four convolution over the grid. Including its padded convolution gives 15.926915072 GFLOPs per solver update. The reported Maze 87.0% uses a 35,000-update cap, costing 557.442 TFLOPs; ARC 47.5% uses a 1,000-update cap, costing 15.927 TFLOPs per view (Movahedi et al., 2026). The Maze score is the best of six evaluation-restart seeds (mean approximately 86.6%), not a six-candidate inference ensemble. Each solver iteration makes one two-block evaluation; unused hierarchical-cycle configuration fields do not multiply this cost.

SE-RRM. The reported Maze and ARC scores are 88.8% and 45.3% (Freinschlag et al., 2026). Its symbol-axis representation requires different accounting from a standard Transformer. With $S = 900$, $D = 256$, $I = 768$, and C symbol channels, a layer costs

$$C_{\mathrm{SE}} = 12SCD^2 + 4CS^2D + 2SC^2D + 6SCDI. \tag{16}$$

This includes position attention and narrower, 128-wide symbol attention. The released recurrence has $3 \times 4 = 12$ two-layer applications per outer step and a 16-step cap. Maze uses $C = 6$, giving 5.994263 TFLOPs; ARC uses $C = 12$, giving 12.001267 TFLOPs per input. Disabling symbol equivariance for Maze changes the embedding/readout, but does not remove the symbol-attention operation.

Coverage and unavailable budgets. Attractor's 27M variant reports 93.1% on Maze-Hard (Fein-Ashley & Rashidinejad, 2026). We retain this score, but the paper does not identify its corresponding solver cap and resolved configuration; an example repository default cannot establish that score's budget. EqR's reported Maze result uses Maze-Unique, not Maze-Hard, and no ARC result was located (Huang et al., 2026); it is therefore excluded from both task columns. These rules cover all external methods in our Sudoku comparisons without importing scores from different datasets. Other ARC pass@k and multi-trajectory studies remain related-work context (Gao et al., 2025; Baek et al., 2026).

LLM reference protocols. The CoT references from HRM's Figure 1 are o3-mini-high, Claude 3.7 (8K), and DeepSeek-R1 score 34.5%, 21.2%, and 15.8% on ARC-AGI-1, respectively, and 0% on Sudoku-Extreme and Maze-Hard (Wang et al., 2025). HRM evaluated Sudoku and Maze through model APIs and quoted the official leaderboard for ARC. Those ARC results use the semi-private split, distinct from our public-evaluation split (Chollet et al., 2025). We mark them with $*$ in Table 1 as cross-split references, not a direct ranking against our ARC scores. Their matrix-compute budgets are unavailable.

The LDT study evaluates GPT-5.4, Claude Opus 4.6, and DeepSeek V4-Pro zero-shot with prompts requiring a completed grid or path, and reports 0% for each on Sudoku-Extreme and Maze-Hard in Tables 1 and 3 (Davis et al., 2026). These are reference observations, not controlled accuracy–compute comparisons: comparable per-puzzle token, reasoning, and FLOP budgets are not established. Zero-shot prompting does not bound LLM ability with different tools or budgets. We omit these scores from the common-budget curve.

Lattice Deduction Transformer reference. LDT's 800K-parameter Sudoku model was reported at 100%, but the authors' repository note corrects the complete-test solve rate to approximately 99.96% after an evaluation inadvertently skipped a full pass (Davis et al., 2026). On Maze-Hard, its 1.8M-parameter model reports 99.3% with one training solution per puzzle and 99.9% with 512 sampled shortest-path training solutions per puzzle. Maze scoring accepts any valid optimal path, whereas our common comparison uses exact match to the reference path; the repository flags this mismatch for HRM/TRM baselines. Beyond multiple inference trajectories, which C-voting also uses, LDT changes the learning and solver protocol: it represents per-cell candidate sets as a lattice, supervises state-dependent candidate elimination and conflict detection on evolving search states, and uses explicit branching, conflict-triggered resets, and parallel search chains at inference (Davis et al., 2026). Ground-truth solutions supervise training and score outputs post hoc; they do not select a winning inference chain. We report LDT separately because its candidate-set search protocol and Maze scoring differ from the latent-refinement comparisons above. These endpoints do not establish a matched-compute ranking.

C.4 Same-backbone ablations and replication

Table 8: Endpoint accuracies (%) for the full-test ablation in Figure 3a. Both models are evaluated in one continuous rollout on all 422,786 Sudoku test puzzles at training update 97,650. Each column gives a common matrix-compute cap; 16 outer steps is the training cap.

Budget (TFLOPs/puzzle)	0.218	0.436	0.872	1.743
Outer steps	16	32	64	128
LRFeed (w/o feedback)	71.16	74.73	77.15	79.05
LRFeed	83.18	88.24	91.48	93.61

Continuation metrics. Figure 3b fixes the 52,287 full-test puzzles that both models fail at the training limit of 16 outer steps. At step 128, LRFeed solves 57.87% of this cohort, compared with 19.28% for LRFeed (w/o feedback). The cohort stays fixed across all subsequent readouts.

Figure 4 instead conditions on each model's remaining errors at the start of each window, as defined by Equation 9. Over the full-test windows 16–32, 32–64, and 64–128, LRFeed solves 30.06%, 27.56%, and 25.06% of its remaining puzzles, respectively; LRFeed (w/o feedback) solves 12.37%, 9.60%, and 8.32%. Later windows contain more steps. The equivalent per-step rates $1 - (1 - q_m(a, b))^{1/(b-a)}$ decrease for both models; the window rates do not imply a constant per-step probability of success.

Table 9: Extended-depth ablation: exact accuracy (%) on 3,072 puzzles at training update 97,650. This table preserves the longer rollout used for paired uncertainty estimates.

Outer steps	16	32	64	128	256	512
LRFeed (w/o feedback)	71.58	75.29	77.47	79.59	81.45	82.78
LRFeed	82.49	87.60	90.89	93.13	95.02	95.80

Table 10: Replicated ablation: exact accuracy (%) on 512 disjoint puzzles at training update 97,650.

Model	16 steps	33 steps	128 steps	512 steps
LRFeed (w/o feedback)	66.02	70.70	76.56	78.52
LRFeed	81.84	89.26	93.95	96.48

Conditioning-input control. The constant-input control retains LRFeed's conditioning MLP, gates, normalization, modulation scale, dropout, and training-time halting policy. It replaces $\widehat{p}_t$ with the fixed vector $\mathrm{RMSNorm}(\mathbf{1})/\sqrt{2}$ at every call.

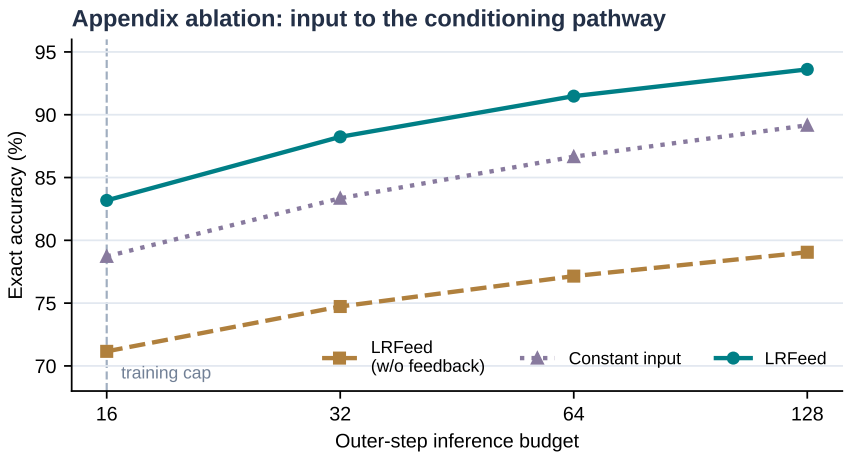


Figure 9: **Conditioning-input ablation.** Exact accuracy on all 422,786 Sudoku test puzzles at training update 97,650. The control keeps the conditioning pathway but replaces the residual profile with a fixed input. Points show the evaluated endpoints.

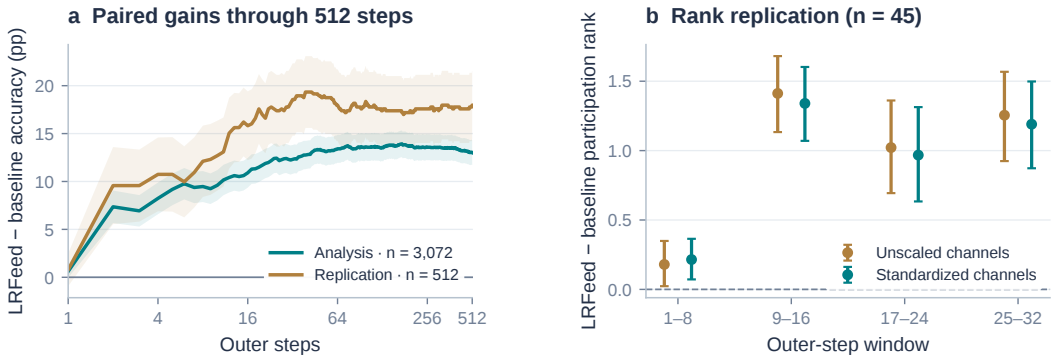


Figure 10: **Complementary replication checks.** (a) Paired accuracy gains of LRFeed over LRFeed (w/o feedback) through 512 steps, including the final 256–512 interval; bands are pointwise 95% normal intervals over paired puzzles. (b) Rank differences on the 45 puzzles unsolved throughout both 33-step replication trajectories. Channel scales are either unchanged or fitted on a separate puzzle set before evaluation; error bars are paired bootstrap 95% intervals.

The parameter-matched constant-input control does not reach LRFeed's accuracy at any of these budgets (Figure 9); the added modulation pathway alone does not account for the full performance.

In the 256–512 window on the 3,072-puzzle subset, LRFeed resolves 15.69% of its remaining puzzles, compared with 7.19% for LRFeed (w/o feedback). These conditional populations differ between models; the paired accuracy differences in Figure 10a use identical puzzles.

C.5 Continuous long-horizon inference

We extend the fixed LRFeed checkpoint to 2,048 outer steps on all 422,786 Sudoku-Extreme test puzzles. We use batch size 128 and the same endpoint readout as the full-test comparison. Each puzzle follows one continuous trajectory. We do not restart at intermediate budgets or select among multiple trajectories. Figure 11 shows the accuracy of the current answer at each step.

Accuracy increases from 93.61% at 128 steps to 96.01% at 512 steps and 97.16% at 2,048 steps. The checkpoint, test set, and rollout policy are fixed throughout, with no noise injection, label-dependent stopping, or candidate selection. The first 128 steps reproduce the full-test accuracy curve in Table 8 exactly.

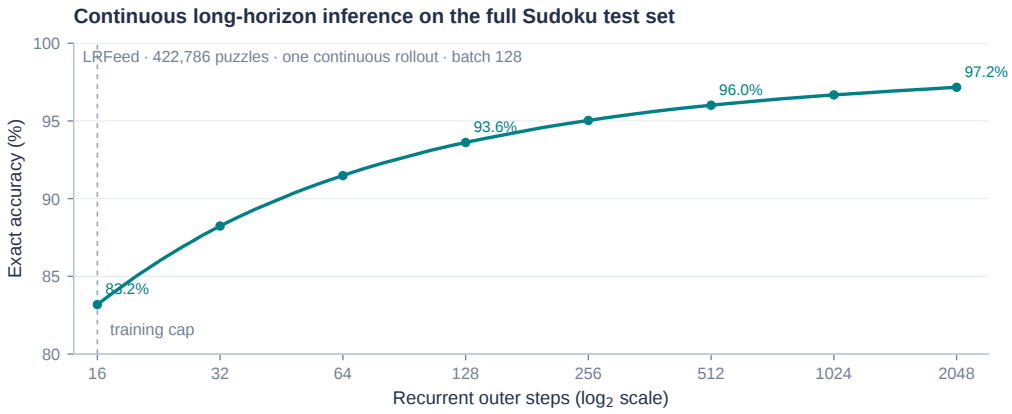


Figure 11: **Continuous long-horizon inference.** Exact accuracy during one uninterrupted rollout on the complete 422,786-puzzle test set. The curve starts at the 16-step training cap, marked by the dashed line. All later points exceed that cap. The horizontal axis is logarithmic.

Table 11: Endpoint accuracy during the same continuous rollout.

Steps	16	32	64	128	256	512	1024	1536	2048
Accuracy (%)	83.18	88.24	91.48	93.61	95.03	96.01	96.67	96.97	97.16

Extended comparison with TRM. We compare continuous full-test rollouts of LRFeed through 2,048 outer steps and TRM through 975 outer steps on the same 422,786 puzzles. LRFeed uses batch size 128 and TRM batch size 1,024. The largest common matrix-compute cap is approximately 27.89 TFLOPs per puzzle: LRFeed uses 27.891 TFLOPs and TRM uses 27.862 TFLOPs, the largest complete depths within that cap. At these endpoints, LRFeed reaches 97.16% and TRM 80.85%, a difference of 16.31 percentage points. Table 12 also shows intermediate budgets from the same continuous rollouts.

Table 12: **Full-test accuracy at extended compute budgets.** Each model uses the largest integer number of outer steps within the specified estimated matrix-compute cap per puzzle. Accuracies are current-answer endpoint percentages, without candidate or step selection.

Budget (TFLOPs)	0.218	0.436	0.872	1.743	3.486	6.973	9.288	27.891
TRM outer steps	7	15	30	60	121	243	325	975
LRFeed outer steps	16	32	64	128	256	512	682	2048
TRM accuracy (%)	65.36	70.54	73.37	75.45	77.12	78.56	79.08	80.85
LRFeed accuracy (%)	83.18	88.24	91.48	93.61	95.03	96.01	96.32	97.16

C.6 Measured inference latency

Full-panel timing. We time both update-97,650 checkpoints on the same RTX 3090. We use eager bfloat16 inference, batch size 32, and one CPU thread. Each model first runs three warm-up rollouts of 128 steps. At each budget, we then time all 128 batches of the 4,096-puzzle panel once. We randomize the order of batch–budget pairs and the model order within each pair. GPU synchronization brackets every timing interval. Inputs are already on the GPU. Timing includes carry initialization and readout, but excludes loading, data transfer, and scoring.

At the reported operating points, LRFeed reaches 81.98% at 16 steps in 46.36 seconds, while LRFeed (w/o feedback) reaches 77.66% at 128 steps in 324.36 seconds. At equal depth, LRFeed takes approximately 14–15% more forward time (Table 13).

Table 13: **Measured inference on all 4,096 replication puzzles.** Total resident forward wall time in seconds and exact accuracy in percent. Each entry includes 128 different batches, not repeated timing of one batch.

	Time (s)		Accuracy (%)	
Outer steps	LRFeed (w/o feedback)	LRFeed	LRFeed (w/o feedback)	LRFeed
16	40.47	46.36	69.26	81.98
32	81.07	92.59	73.05	87.38
64	161.38	184.30	75.59	90.55
128	324.36	370.79	77.66	92.72

D INFORMATION IN RESIDUAL FEEDBACK

D.1 WHAT INFORMATION THE FEEDBACK RETAINS

Let $\Delta_t = z_t - z_{t-1} \in \mathbb{R}^{S\times D}$. Its uncentered second moment is $Q_t = \Delta_t^\top \Delta_t / S$, and the feedback module's squared RMS profile is exactly $e_t = \mathrm{diag}(Q_t)$ in real arithmetic. Define the relative energy profile $\pi_t = e_t/(\mathbf{1}^\top e_t)$. For any nonzero e_t, our normalized input satisfies

$$\frac{\widehat{p}_{t,c}^2}{\sum_j \widehat{p}_{t,j}^2} = \pi_{t,c}. \tag{17}$$

This identity also holds with nonzero normalization ϵ: the common scale factor cancels. The input retains an ϵ-dependent attenuation, which can matter near zero updates; bfloat16 adds numerical approximation. The feedback module therefore observes relative channel changes directly, without reconstructing them from the absolute latent state.

A common additive component cancels: $(z_t + b) - (z_{t-1} + b) = \Delta_t$. This distinguishes change feedback from state-energy pooling; the full state remains available to the backbone.

Geometry of the feedback direction. For nonzero profiles, let $v_t = \widehat{p}_t/\|\widehat{p}_t\|_2$. Since RMS pooling gives nonnegative entries, the preceding identity yields

$$\begin{aligned} v_t &= \sqrt{\pi_t}, \qquad v_t^\top v_u = \sum_c \sqrt{\pi_{t,c}\pi_{u,c}}, \\ \|v_t - v_u\|_2^2 &= 2\left(1 - \sum_c \sqrt{\pi_{t,c}\pi_{u,c}}\right), \end{aligned} \tag{18}$$

where the square root is elementwise. Thus the feedback direction encodes how update energy is allocated across channels: identical allocations give the same direction, while disjoint allocations give orthogonal directions. This direction is unchanged by a common positive rescaling of the update, even with nonzero ϵ. The magnitude of the actual MLP input retains the ϵ-dependent attenuation described above.

D.2 RESIDUAL ENERGY AND PREDICTION

For a differentiable loop without feedback and with fixed input x, write $\delta_t = \mathrm{vec}(\Delta_t)$. The fundamental theorem of calculus gives the exact secant relation

$$\delta_{t+1} = \overline{J}_t \delta_t, \qquad \overline{J}_t = \int_0^1 J_{\Phi_x}(\mathrm{vec}(z_{t-1}) + u\delta_t)\, du. \tag{19}$$

If B_c selects channel c across positions, then

$$e_{t+1,c} = \frac{1}{S}\delta_t^\top \overline{J}_t^\top B_c^\top B_c \overline{J}_t \delta_t. \tag{20}$$

The next update energy depends on the preceding change and the intervening Jacobian. RMS pooling retains its per-channel energy while discarding directions and cross-channel terms. This relation in the unconditioned backbone motivates testing the predictive value of the retained energy.

D.3 PUZZLE-HELD-OUT ATOMIC PREDICTION

We use 128 outcome-independent puzzles from the disjoint replication capture. Both models are evaluated at batch size 32. Each of 33 outer steps contains ten atomic calls. For calls $j = 1, \ldots, 9$, predictors use state energy and/or incoming residual energy at call $j - 1$ to predict residual energy at call j (297 rows per puzzle); the first call of each outer step is excluded. These are pooled-energy features, not the full latent state, and the target is energy allocation rather than signed displacement.

All predictors include a one-hot known atomic phase. Additional features are the current state-energy profile, incoming residual-energy profile, or both, each normalized to unit channel-energy sum. Feature centering and scaling are fitted only on training puzzles. Ridge penalties are selected

from $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 10^3, 10^4\}$ using puzzles 0–47 for fitting and 48–63 for validation, followed by refitting on 0–63. Puzzles 64–127 are held out from these fits. Table 2 reports their average per-puzzle MSE. Bootstrap intervals use 5,000 paired resamples of the 64 held-out puzzles, not individual transitions. Adding incoming residual energy reduces test MSE relative to state energy alone by 24.86% for LRFeed (w/o feedback) (paired 95% CI [23.30, 26.41]) and 11.83% for LRFeed ([11.31, 12.42]). Both predictors also include recurrent phase.

The capacity control permutes residuals across puzzles independently at each phase and separately within the training, validation, and test partitions. It retains the same features and ridge-selection procedure. A second control permutes only the test residual supplied to the fitted state-plus-residual probe. This tests the probe's use of its input, not the reasoning model's dependence on online feedback.

Table 14: Additional prediction controls. Held-out MSE ($\times 10^{-7}$). These complement the four primary feature comparisons in Table 2.

Predictor	LRFeed (w/o feedback)	LRFeed
Phase + residual energy	2.161	3.377
Incoming-profile persistence	4.168	6.332
State + residual probe, test residual shuffled	4.753	4.937

For the refitted capacity control, shuffled-minus-correct MSE is 7.72×10^{-8} for LRFeed (w/o feedback) (95% CI $[7.06, 8.42] \times 10^{-8}$) and 4.75×10^{-8} for LRFeed ($[4.52, 5.00] \times 10^{-8}$). The result supports information in the sample-aligned residual beyond the chosen state descriptor. These are exploratory probes fitted with puzzle-level holdouts from the trajectory analysis data.

E INTERNAL AND OUTPUT DYNAMICS

E.1 CHANNEL-PROFILE PARTICIPATION RANK

We use matched batch-32 captures on 3,072 puzzles and a disjoint 512-puzzle replication set. The state profile is the per-channel RMS of the latent state over sequence positions at each outer-step endpoint. It is distinct from the residual profile supplied to the feedback MLP. Each window has eight steps: 1–8, 9–16, 17–24, and 25–32. For its profile matrix $P \in \mathbb{R}^{8\times D}$, set

$$Y_{t,c} = P_{t,c} - \frac{1}{8}\sum_{u=1}^{8} P_{u,c}, \qquad X_{t,:} = \frac{Y_{t,:}}{\max(\|Y_{t,:}\|_2, 10^{-12})}. \tag{21}$$

Equation 10 is evaluated on $XX^\top$; the centered-only control uses $YY^\top$. Temporal centering bounds the rank by seven. The statistic summarizes spectral concentration within a window; it is invariant to row permutations and does not encode temporal ordering.

The fixed unsolved cohort contains 261 main-set and 45 replication puzzles for which neither model is correct at any of the 33 endpoints. This cohort is selected using future outcomes and is descriptive. In the unselected full 3,072-puzzle population, normalized rank differences over the four windows are -0.32, $+0.86$, $+0.43$, and $+0.48$. For the fixed cohort, centered-only differences are -0.85, $+0.95$, $+0.82$, and $+0.95$. Thus, the first-window contrast changes with cohort and normalization.

For the scale-control replication, per-channel standard deviations are fitted separately in each model on the 3,072-puzzle capture, then applied to the disjoint 512 puzzles before centering and normalization. Scales are floored at the larger of 5% of their median and 10^{-8}. Difference intervals use 5,000 paired puzzle bootstrap resamples.

E.2 STATE-ACTIVITY GEOMETRY OF THE TRM REPRODUCTION

We capture both TRM state streams and the LRFeed state on the 512-puzzle replication set. At atomic cap $21k$, $k = 1, \ldots, 64$, TRM is read at outer step k and LRFeed at $\lfloor 21k/10 \rfloor$. The maximum rounding gap is nine atomic calls; no interpolation is used. Each spectral window contains eight sampled endpoints. TRM's answer state is the primary readout comparison, with its reasoning state and concatenated streams retained as sensitivity checks.

On the fixed cohort, TRM's answer-state profile rank remains approximately 3.0–3.35, while LRFeed's is 4.44–4.89 (Figure 12). At caps 189–336, their means are 3.080 and 4.685 (paired difference 1.604, 95% CI [1.003,2.111]). The contrast remains without row normalization (2.626 versus 4.066) and using TRM's reasoning state. TRM's rank is comparatively flat; on all 512 puzzles the first-window ordering reverses, while the later-window ordering remains.

E.3 READOUT-SPACE REPLICATION

Protocol. We randomly selected 4,096 test inputs disjoint from the 3,072- and 512-puzzle analysis sets, without using model outcomes. Both models use update-97,650 checkpoints, batch size 32, and continuous 128-step inference. The primary comparison covers steps 121–128 on common failures at step 120. Secondary windows 17–24 and 57–64 use common failures at steps 16 and 56. Each comparison pairs identical puzzles; cohorts can differ between windows.

From latent profiles to answer distributions. Let v_t concatenate the decoder's Stablemax probabilities over all cells and vocabulary entries. In each eight-step window W, we center these vectors as $v_t - \bar{v}$, where $\bar{v} = |W|^{-1}\sum_{t\in W} v_t$. We normalize each row and compute participation rank with Equation 10. Its maximum is seven. We report variation magnitude separately:

$$A_W = \left(\frac{1}{|W|}\sum_{t\in W} \|v_t - \bar{v}\|_2^2\right)^{1/2}. \tag{22}$$

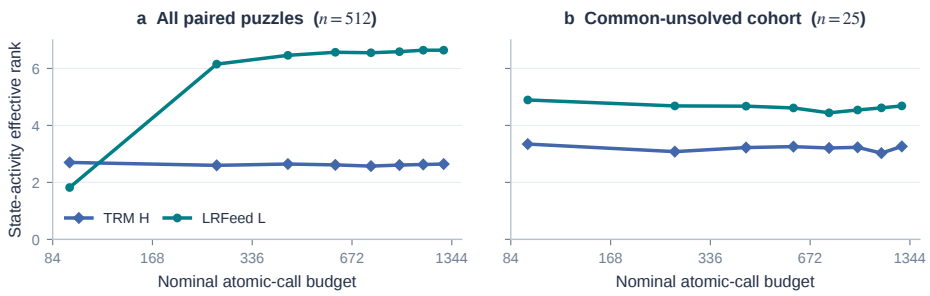


Figure 12: **External-model state-activity comparison.** Participation rank at aligned atomic budgets on (a) all 512 paired puzzles and (b) the fixed 25 puzzles unsolved by both models at every sampled endpoint. Curves show cohort means; this comparison also changes the recurrent architecture.

Table 15: **Output-profile rank on disjoint replication puzzles.** Each cohort contains common failures before the window. Differences are LRFeed minus LRFeed (w/o feedback); intervals use 2,000 paired-puzzle bootstrap samples. The final row is the primary comparison.

Window	n	LRFeed (w/o feedback)	LRFeed	Difference	95% CI
17–24	572	3.97	4.21	+0.23	[0.10, 0.37]
57–64	300	4.04	4.59	+0.55	[0.41, 0.70]
121–128	220	4.05	4.63	+0.58	[0.41, 0.75]

Results and controls. LRFeed has higher output-profile rank in all three windows. On the primary cohort, its state-profile rank is also higher (4.40 versus 3.34). Output variation is smaller (0.683 versus 1.009; paired difference -0.326 $[-0.372, -0.277]$). The rank difference reflects how variation is distributed across directions, not an increase in magnitude.

The primary difference is similar without row normalization ($+0.58$, 95% CI [0.42, 0.75]) and on initially blank cells ($+0.58$, [0.41, 0.75]). In subsets matched to within two initial errors, the difference is positive at steps 121–128 ($+0.44$, [0.11, 0.77], $n = 61$) and 57–64 ($+0.66$, [0.40, 0.92], $n = 98$), but uncertain at steps 17–24 ($+0.15$, $[-0.13, 0.42]$, $n = 159$).

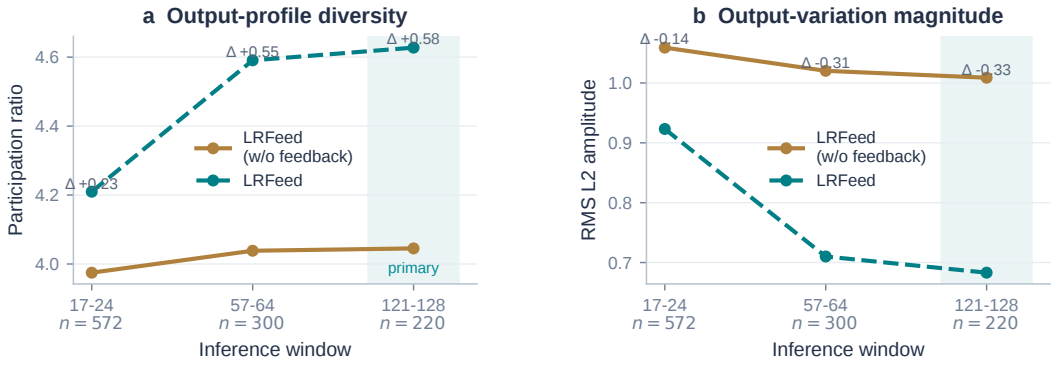


Figure 13: **Readout variation on common unresolved puzzles.** (a) Participation rank of centered, row-normalized answer probabilities. (b) Magnitude of centered variation (Equation 22). Labels give paired mean differences; shading marks the primary window. Table 15 gives intervals for rank differences.

E.4 Spectral geometry of channel profiles

The following identities help interpret the observed rank differences in terms of channel energy and correlated activity. Participation rank summarizes spectral concentration within a window and is invariant to permutations of its steps.

E.4.1 Channel energy and correlated activity

For the same temporally centered, row-normalized profile matrix X used in Section 4.4, set $C = X^\top X$ and $e_c = C_{cc}$. Define channel-energy diversity E and energy-weighted squared channel overlap κ (for $E > 1$) as

$$E = \frac{(\sum_c e_c)^2}{\sum_c e_c^2}, \qquad \kappa = \frac{\sum_{c \neq d} C_{cd}^2}{\sum_{c \neq d} e_c e_d}. \tag{23}$$

The latter is a weighted average of squared cosines between columns of X; zero-energy pairs contribute nothing. To separate energy allocation from correlation, write $T = \sum_c e_c = \mathrm{tr}(C)$. Then

$$\begin{aligned} \mathrm{tr}(C^2) &= \underbrace{\sum_c e_c^2}_{\text{energy concentration}} + \underbrace{\sum_{c \neq d} C_{cd}^2}_{\text{channel overlap}} \\ &= \frac{T^2}{E} + \kappa \left(T^2 - \frac{T^2}{E} \right). \end{aligned} \tag{24}$$

Since $XX^\top$ and C have the same nonzero eigenvalues, for $E > 1$ this gives

$$r_{\mathrm{PR}}(XX^\top) = \frac{1}{E^{-1} + (1 - E^{-1})\kappa}. \tag{25}$$

For $E = 1$ the rank is one and κ is unnecessary. Thus broader energy allocation increases participation rank at fixed $\kappa < 1$. Broad allocation alone is insufficient when channel trajectories are strongly correlated.

On the 512-puzzle replication capture, the fixed cohort unsolved throughout contains 45 puzzles. At steps 9–16, state-profile participation rank is 3.41 for LRFeed (w/o feedback) and 4.82 for LRFeed. Energy diversity is 247 versus 300, while weighted channel overlap is 0.306 versus 0.210. The same later-window ordering holds on all 512 puzzles and without row normalization. An arithmetic decomposition of the per-puzzle rank difference averages the two orders of substituting E and κ into Eq. (25). It assigns 0.010 of the 1.412 mean gap to E and 1.402 to κ. In this algebraic decomposition, the spectral difference is predominantly associated with less correlated channel trajectories, not simply more active channels.

How residual energy relates to state activity. At an atomic transition $z_{t+1} = z_t + \Delta z_{t+1}$, write $s_{t,c}^2 = S^{-1} \sum_i z_{t,i,c}^2$ and $p_{t+1,c}^2 = S^{-1} \sum_i \Delta z_{t+1,i,c}^2$. Then

$$s_{t+1,c}^2 - s_{t,c}^2 = \frac{2}{S} \sum_i z_{t,i,c} \Delta z_{t+1,i,c} + p_{t+1,c}^2, \qquad |s_{t+1,c} - s_{t,c}| \leq p_{t+1,c}. \tag{26}$$

The identity follows by expansion and the bound by the reverse triangle inequality. State-energy change depends on both residual energy and state–update alignment. Residual magnitude also bounds the displacement of the state RMS profile. LRFeed retains the relative channel-energy pattern through normalization. These identities describe atomic transitions; the state-profile figures sample the ends of outer steps.

E.4.2 Participation rank as directional redundancy

Proposition 1 (Directional redundancy and compression) *Let $X \in \mathbb{R}^{n \times D}$, $n \geq 2$, contain nonzero, temporally centered profiles subsequently normalized to unit row norm. Write $r = r_{\mathrm{PR}}(XX^\top)$ and $\mathcal{R} = [n(n-1)]^{-1} \sum_{i \neq j} (x_i^\top x_j)^2$. Then*

$$r = \frac{n}{1 + (n-1)\mathcal{R}} \leq \min(D, n-1), \qquad \frac{\|X - X_k\|_F^2}{\|X\|_F^2} \geq \left[1 - \sqrt{k/r} \right]_+, \tag{27}$$

where X_k is a best rank-k approximation and $[v]_+ = \max(v, 0)$.

Proof of Proposition 1. Let Y be the temporally centered profile matrix, with row norms $a_i = \|Y_{i,:}\|_2 > 0$, and $X = \mathrm{Diag}(a)^{-1}Y$. Centering gives $Y^\top \mathbf{1} = 0$, hence $X^\top a = 0$ and $\mathrm{rank}(X) \le \min(D, n-1)$. Importantly, normalizing the rows does not generally preserve $X^\top \mathbf{1} = 0$. For $G = XX^\top$, each diagonal entry is one. Expanding the frame potential (Benedetto & Fickus, 2003) gives

$$\mathrm{tr}(G) = n, \qquad \mathrm{tr}(G^2) = \sum_{i,j} (x_i^\top x_j)^2 = n + n(n-1)\mathcal{R}. \tag{28}$$

Substitution gives the first identity. If λ_j are the nonzero eigenvalues of G, Cauchy–Schwarz gives $(\sum_j \lambda_j)^2 \le \mathrm{rank}(G) \sum_j \lambda_j^2$, proving the upper bound. Nonnegativity also gives $r \ge 1$.

For the compression bound, order the eigenvalues decreasingly and write $\pi_j = \lambda_j / \mathrm{tr}(G)$, padding with zeros as needed. The singular-value decomposition gives

$$\frac{\|X - X_k\|_F^2}{\|X\|_F^2} = 1 - \sum_{j=1}^{k} \pi_j \ge 1 - \sqrt{k \sum_j \pi_j^2} = 1 - \sqrt{k/r}. \tag{29}$$

Combining this inequality with nonnegativity proves the result. Retaining at least $1-\delta$ of the squared Frobenius energy therefore requires $k \ge r(1-\delta)^2$. This is a necessary, not sufficient, condition. □

The geometry at maximum rank. The rank bound gives an unavoidable overlap:

$$\mathcal{R} \ge \frac{1}{(n-1)^2}. \tag{30}$$

For $D \ge n-1$, equality is attainable. Suppose $r = n-1$. The $n-1$ nonzero eigenvalues must be equal. Since $Ga = 0$ and $\mathrm{tr}(G) = n$,

$$G = \frac{n}{n-1}\left(I - \frac{aa^\top}{\|a\|_2^2}\right). \tag{31}$$

Its unit diagonal forces $a_i^2 = \|a\|_2^2/n$. All a_i are positive, so

$$G = \frac{n}{n-1}\left(I - \frac{\mathbf{1}\mathbf{1}^\top}{n}\right), \qquad x_i^\top x_j = -\frac{1}{n-1} \quad (i \ne j). \tag{32}$$

The profiles therefore form a regular simplex. For eight-step windows, this configuration has rank seven and pairwise inner products $-1/7$. Maximal participation rank thus corresponds to evenly spread profile directions. The nonzero overlap follows from temporal centering, even in this maximally spread configuration.

The proposition assumes every centered profile has norm at least 10^{-12}, the normalization floor used in the analysis; no profile in the replication capture reaches that floor.