RECURSIVE REASONING AS ATTRACTOR LANDSCAPE SEARCH: MECHANISTIC DYNAMICS OF THE TINY RECURSIVE MODEL

Andreas Efstathiou[§]
School of Engineering and Applied Science
University of Pennsylvania
andas@seas.upenn.edu

Aishwarya Balwani[§]
Department of Developmental Neurobiology
St. Jude Children's Research Hospital
aishwarya.balwani@stjude.org

ABSTRACT

Current reasoning systems predominantly rely on chain-of-thought approaches that generate explicit reasoning tokens, though recent work has begun exploring alternative architectures. Latent reasoning models such as the Tiny Recursive Model (TRM) recursively refine continuous hidden states rather than producing readable intermediate steps, and have demonstrated competitive performance with large language models while using orders of magnitude fewer parameters. However, these models remain poorly understood mechanistically – most evaluations focus on behavioral outcomes rather than the internal dynamics that produce them, limiting our ability to improve their reliability and efficiency. We therefore conduct a mechanistic analysis of TRM's recursive dynamics on Sudoku-Extreme, revealing three critical insights: (i) Sparse autoencoder analysis shows models form initial hypotheses early, after which feature activations converge to approximately periodic patterns that primarily stabilize rather than refine solutions; (ii) Reasoning trajectories in latent principal component space diverge within early recursion steps and descend into local minima determined by initialization, validating similar findings in the Hierarchical Reasoning Model; (iii) During inference, failed runs plateau at stable high-loss attractors rather than continuing exploration. Leveraging these insights, we demonstrate that simple inference-time interventions, i.e., adding noise to answer initialization and scaling guess attempts, enable escape from local minima, achieving near-state-of-the-art performance on Sudoku-Extreme using vanilla TRM without retraining. Our findings suggest recursive reasoning operates as adaptive search through an attractor landscape rather than incremental refinement, opening new avenues for test-time scaling.

1 INTRODUCTION

Modern reasoning systems predominantly rely on chain-of-thought (CoT) prompting, where large language models (LLMs) generate explicit step-by-step reasoning tokens before producing final answers (Wei et al., 2023). While effective, this approach scales reasoning depth through token generation, inheriting the computational costs and architectural constraints of autoregressive language models. Recently, however, a growing body of work has begun exploring alternative architectures that perform reasoning in continuous latent space rather than through explicit token sequences (Hao et al., 2024; Shen et al., 2025; Wang et al., 2025). These latent reasoning models recursively refine continuous hidden states, achieving competitive performance with large language models while using orders of magnitude fewer parameters – for instance, the Tiny Recursive Model (TRM) achieves 87% accuracy on Sudoku-Extreme with only 5M parameters (Jolicoeur-Martineau, 2025) compared to billions required for LLM-based approaches.

This dramatic efficiency advantage raises a fundamental question: how do these compact models achieve such strong reasoning performance? Current evaluations focus primarily on behavioral outcomes—accuracy on benchmark tasks—leaving the internal mechanisms that produce this per-

[§]Work conducted with Algoverse AI Research, cohort of Winter 2025.

formance poorly understood. Recent mechanistic analysis of the Hierarchical Reasoning Model (HRM) revealed surprising dynamics: rather than incrementally refining solutions through reasoning steps, HRM appears to "guess" fixed points in latent space, with trajectories either converging to correct solutions or becoming trapped in spurious attractors corresponding to incorrect answers (Ren & Liu, 2026). Understanding these dynamics is crucial for several reasons, e.g., enabling principled improvements to model reliability rather than black-box hyperparameter tuning, addressing whether recursive latent reasoning represents a fundamentally different computational paradigm from token-level reasoning or merely a more parameter-efficient implementation, and for informing strategies for test-time compute scaling.

TRMs offer an ideal testbed for investigating these questions. Compared to HRM's (Wang et al., 2025) dual 4-layer networks operating at different temporal frequencies, TRM uses a single 2-layer network that recursively updates both an answer embedding and a reasoning state. This architectural simplicity allows clearer isolation of fundamental recursive reasoning dynamics. Moreover, the fixed-point dynamics observed in HRM suggest connections to classical work on recurrent neural network optimization and implicit differentiation (Sussillo & Barak, 2013; Maheswaranathan et al., 2019; Driscoll et al., 2024), making mechanistic investigation particularly tractable.

Consequently, we conduct a mechanistic analysis of TRM's reasoning dynamics on Sudoku-Extreme, revealing three critical insights into how recursive latent reasoning operates:

- **Front-loaded reasoning with periodic stabilization:** Using sparse autoencoders trained on TRM's latent activations, we find reasoning is not uniformly distributed across recursion steps. The model forms initial hypotheses early ($\sim 58\%$ of puzzles solved in first 1-2 supervision steps), after which SAE features enter approximately periodic firing patterns that persist for remaining recursion. This suggests later recursive steps primarily stabilize rather than refine the early hypothesis.
- **Latent attractor dynamics determined by initialization:** Visualizing reasoning trajectories in principal component (PC) space, we observe successful and failed runs diverge within early recursion steps. Varying answer initializations reveal the models descend toward local minima determined by initial conditions; some puzzles exhibit broad success basins, while others have compact failure regions corresponding to spurious attractors. This validates and extends the HRM findings by Ren & Liu (2026) to the TRM's architecture.
- **Practical escape from local minima via inference-time perturbation:** Since reasoning failures stem from initialization-dependent local minima rather than insufficient recursive depth, simple inference-time interventions enable escape. By perturbing answer initialization with noise and scaling parallel guess attempts, we achieve 97.8% accuracy on Sudoku-Extreme using vanilla TRM without retraining or architecture modification.

Taken together, our findings suggest that recursive latent reasoning operates as adaptive search through an attractor landscape rather than incremental refinement. This perspective has important implications: the compact size of latent reasoning models (orders of magnitude smaller than LLMs) makes inference-time exploration strategies particularly practical and cost-effective, opening new avenues for test-time compute scaling. Moreover, the comparable mechanistic patterns we observe between HRM and TRM—despite significant architectural differences—suggest these dynamics may reflect fundamental properties of recursive reasoning systems rather than architecture-specific artifacts.

2 Background and Experimental Methods

2.1 Background on the Tiny Recursive Model

The Tiny Recursive Model (TRM) (Jolicoeur-Martineau, 2025) is a latent reasoning model that recursively refines two latent states: an embedded solution z_h and a reasoning state z_l. In each *cycle*, the model performs 6 updates to z_l followed by one update to z_h; three cycles constitute one *supervision step*. During inference, TRM chains up to 16 supervision steps, carrying forward latent states between steps. An output head decodes the final z_h to produce the predicted solution $\hat{y}$. Training employs deep supervision with gradients computed only for the final cycle of each supervision step, and adaptive computation time (ACT) enables early halting when solutions are

reached. We analyze the 5M parameter MLP variant. Full architectural details are provided in Appendix A.

2.2 Experimental Setup

Sparse Autoencoders. To interpret TRM's latent dynamics, we train a sparse autoencoder (SAE) on residual stream activations with a gated-ReLU architecture and expansion factor $k = 32$, projecting the 512-dimensional hidden state into 16,384 sparse features (see Appendix B).

Dataset and Training. We train our TRMs on the Sudoku-Extreme dataset, consisting of 1,000 base puzzles each augmented 1,000 times through rule-preserving permutations (1,001,000 training examples total). Testing and SAE analysis are performed on the held-out test set.

3 Results And Analysis

3.1 Recursive computation rapidly collapses to a stable internal regime

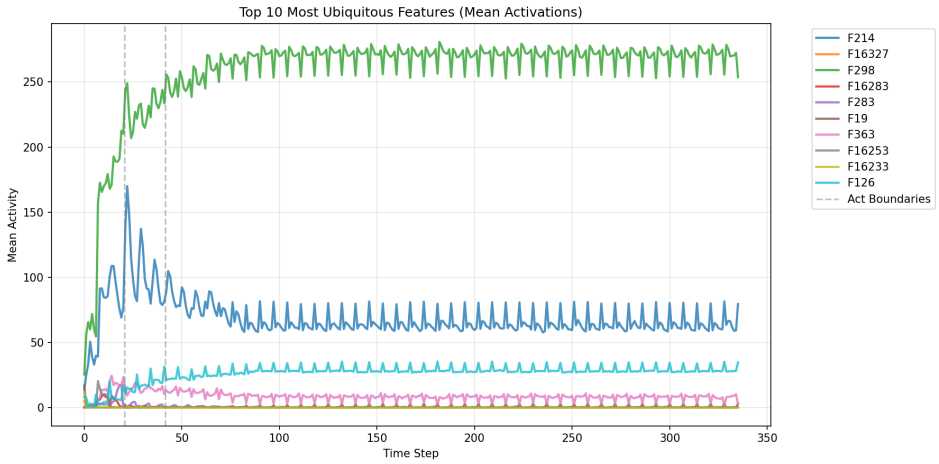

Figure 1: *Periodic features emerge after early transient dynamics.* Total activation over time for the ten most active SAE features during inference. The x-axis shows time steps across multiple supervision steps (vertical dashed lines mark ACT boundaries).

The most active SAE features exhibit a consistent two-phase structure during inference (Figure 1). Following a brief early transient in which feature magnitudes re-rank and spike (first ~ 50 steps), the same dominant features enter approximately periodic firing patterns across cycles that persist for the remainder of recursion, even when additional ACT steps are available. This indicates that, at the level of salient internal features, TRM inference rapidly settles into a stable internal loop rather than continually constructing new representations. Later recursive computation primarily re-expresses this stabilized feature configuration, consistent with early commitment followed by dynamical reinforcement.

Table 1: Early solution statistics showing front-loaded convergence across ACT steps, evaluated on the first 10k puzzles of the test set.

Step	Cumulative Percentage Solved
1	15.6%
2	53.9%
3	65.6%
5	75.1%
10	83.4%
16	87.2%

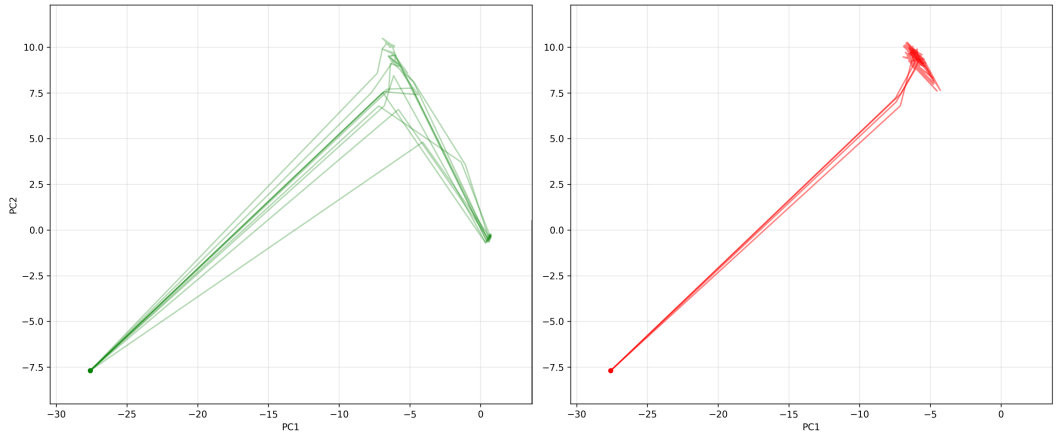

Figure 2: *Latent Reasoning Manifolds.* 2D PCA projection of the reasoning state z_L across 16 ACT steps. Left: Successful (green) trajectories. Right: Failing (red) trajectories converge to 'attractors'.

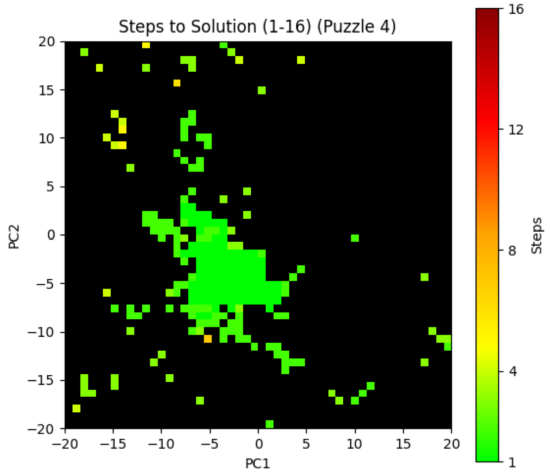

Figure 3: *Latent-space visualization of inference trajectories under varying* h_{init}. Each point represents an initialization colored by steps-to-solution as per the key, or black if no solution is reached after 16 ACT steps.

Table 1 shows that the majority of successful solutions are reached within this early transient phase, aligning feature stabilization with solution discovery. Over half of all solved puzzles (53.9%) reach correct solutions by step 2, with diminishing returns from additional computation—steps 11-16 contribute only 3.8% additional accuracy. This front-loaded convergence pattern suggests that TRM's reasoning capacity is concentrated in the initial steps, while later steps primarily consolidate rather than fundamentally alter the solution trajectory.

3.2 The evolution of z_l across time steps

Visualizing the evolution of z_l in PCA space reveals that successful and failed reasoning trajectories diverge within the earliest recursion steps (Figure 2). Successful trajectories (green) continue progressing through latent space toward the ground-truth solution, while failed runs (red) rapidly converge to stable attractor points and remain trapped thereafter (more on the dynamics can be seen in Appendix C).

3.3 ANSWER INITIALIZATION DETERMINES BASIN OF ATTRACTION

To probe the dependence on answer initialization, we vary h_{init} and visualize the resulting inference trajectories in PCA space (Figure 3). Multiple initializations lead to successful convergence, occupying clustered regions where solutions are reached in few steps (bright green/yellow). Initializations outside these clusters consistently fail (black), revealing a basin-of-attraction structure in the latent space.

This behavior reflects how the TRM is trained: deep supervision enforces loss minimization at every recursion step, causing inference to behave as a local loss optimizer. Rather than exploring the global landscape, trajectories descend toward nearby minima determined by initialization. Once trapped in a local minimum (whether corresponding to the correct solution or a spurious attractor), the model lacks mechanisms to escape, explaining why additional recursive depth provides diminishing returns beyond early steps.

This mirrors the attractor dynamics observed in HRM (Ren & Liu, 2026), validating that initialization-dependent local minima persist across different latent reasoning architectures. The rapid early divergence suggests that reasoning failures stem from unfavorable initializations rather than insufficient recursive depth.

3.4 INITIALIZATION PERTURBATION ENABLES ESCAPE FROM LOCAL MINIMA

Sections 3.1 through 3.3 demonstrated that vanilla TRM exhibits diminishing returns from scaling recursive depth, with failures stemming from initialization-dependent basins of attraction rather than insufficient reasoning capacity. This suggests a path forward: scale exploration breadth rather than depth. As proof-of-concept, we perturb h_{init} with Gaussian noise and execute multiple short forward passes, reducing the number of ACT steps from 16 to 5, as compared to the vanilla TRM.

For each puzzle, we select the trajectory with highest ACT confidence as the final prediction. This simple inference-time intervention—requiring no retraining or architectural modification—yields substantial performance gains, achieving 97.8% when evaluated on the first 10k puzzles from the test set, representing a 10.6% increase in accuracy as compared to the vanilla TRM. (Table 2).

Table 2: Performance gains from initialization perturbation on Sudoku-Extreme test set (N=10,000 puzzles). Baseline uses single forward pass; perturbation method samples 50 diverse initializations.

ACT Steps	Baseline	+Perturbation	Gain
1	15.6%	15.6%	+0.0%
2	53.9%	77.1%	+23.3%
3	65.6%	94.7%	+29.1%
5	75.1%	97.8%	+22.7%

4 CONCLUSION

Our mechanistic analyses lead to the conclusion that TRM's recursive reasoning operates as adaptive search through an attractor landscape rather than incremental refinement as it was idealized. Three key findings frame this conclusion: (1) reasoning is front-loaded, with features rapidly settling into periodic patterns after early hypothesis formation; (2) trajectories diverge early based on initialization, descending toward local minima in latent space; (3) failures stem from entrapment in spurious attractors rather than insufficient recursive depth. Parrallely, through these insights a second, practical conclusion arises: Much of the TRM's reasoning capability remains latent and can be unlocked through principled manipulation of inference dynamics.

Indeed, by perturbing initialization and sampling multiple inference trajectories, hence exploiting the compact model size that makes parallel sampling tractable, we achieve substantial performance gains without retraining. More broadly, our findings suggest that understanding the geometric structure of latent reasoning spaces enables more principled test-time strategies (Snell et al., 2024), as well as new training-time approaches that leverage these latent dynamics we reveal.

The comparable dynamics observed across HRM and TRM architectures suggest these may reflect fundamental properties of recursive latent reasoning, opening avenues for geometry-aware design of future reasoning systems.

REFERENCES

Laura N Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *Nature Neuroscience*, 27(7):1349–1363, 2024.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. Scaling and evaluating sparse autoencoders. *arXiv preprint arXiv:2406.04093*, 2024.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

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

Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. *Advances in neural information processing systems*, 32, 2019.

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

Xuan Shen, Yizhou Wang, Xiangxi Shi, Yanzhi Wang, Pu Zhao, and Jiuxiang Gu. Efficient reasoning with hidden thinking. *arXiv preprint arXiv:2501.19201*, 2025.

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

David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation*, 25(3):626–649, 2013.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

A TRM ARCHITECTURE DETAILS

A.1 FORWARD PASS

The TRM operates on an input puzzle x, an embedded solution state z_h, and a latent reasoning state z_l (initially set to h_{init} and l_{init} values, respectively). We denote the update applied to z_l by $f_l(\cdot)$ and the update applied to z_h by $f_h(\cdot)$.

A single *cycle* of the TRM consists of 6 updates of the latent reasoning state z_l:

$$z_l^{(i+1)} = f_l(x, z_h, z_l^{(i)}), \qquad i = 0, \ldots, 5,$$

followed by an update of the embedded solution z_h:

$$z_h \leftarrow f_h(z_h, z_l).$$

Three cycles ($T = 3$) constitute one *supervision step*. During inference, a forward pass consists of 16 supervision steps chained together, where the z_l and z_h values from the previous step are carried forward as initial conditions for the next step. At the end of the forward pass, an output head decodes the final z_h to produce the predicted solution $\hat{y}$.

A.2 TRAINING PROCEDURE

During training, deep supervision is applied across supervision steps. The first $T - 1$ cycles in each supervision step run without gradient computation; only the final cycle has gradients enabled. The final z_h is decoded to produce a prediction $\hat{y}$, and cross-entropy loss is computed against the ground-truth solution, with gradients backpropagating only through that cycle. The resulting states (z_l, z_h) are then detached from the computational graph and used to initialize the next supervision step.

This process repeats for up to 16 supervision steps during training. Adaptive computation time (ACT) enables early halting when a supervision step produces a correct solution, determined by the condition $q_{\textit{halt-logits}} > q_{\textit{continue-logits}}$. At inference time, the model always executes the full sequence of 16 supervision steps.

B SPARSE AUTOENCODER DETAILS

To interpret the latent dynamics of TRM, we train a sparse autoencoder (SAE) on the residual stream activations collected across all positions and layers during forward passes. We employ a gated-ReLU architecture (Gao et al., 2024) with an expansion factor of $k = 32$, projecting the 512-dimensional hidden state into a 16,384-dimensional sparse feature space.

The SAE is trained to minimize reconstruction loss while encouraging sparsity through the gated-ReLU activation function, which enables better feature disentanglement compared to standard ReLU SAEs. Training is performed on activations collected from both successful and failed reasoning trajectories to ensure comprehensive coverage of the latent space. Running it on the

C ADDITIONAL ANALYSIS OF REASONING DYNAMICS

Figure 4 shows cross-entropy loss evolution across recursive cycles for both successful and failed inference attempts. Successful runs (green, individual traces shown as thin lines) exhibit continuous loss reduction, with the averaged trajectory (thick green line) monotonically decreasing to zero within approximately 10 cycles. In contrast, failed runs (red) demonstrate a markedly different pattern: after an initial steep descent in the first 2-3 cycles, loss plateaus at stable values around 11-12 and remains flat for the remainder of inference. This plateau indicates convergence to suboptimal solutions in the answer space—spurious attractors from which the model cannot escape despite continued recursive computation. The vertical dashed lines demarcating supervision step boundaries (every 3 cycles) show that this divergence between success and failure modes occurs within the first supervision step, consistent with our finding that reasoning is front-loaded and early initialization determines trajectory outcomes. Figure 5 shows us something similar in the latent space.

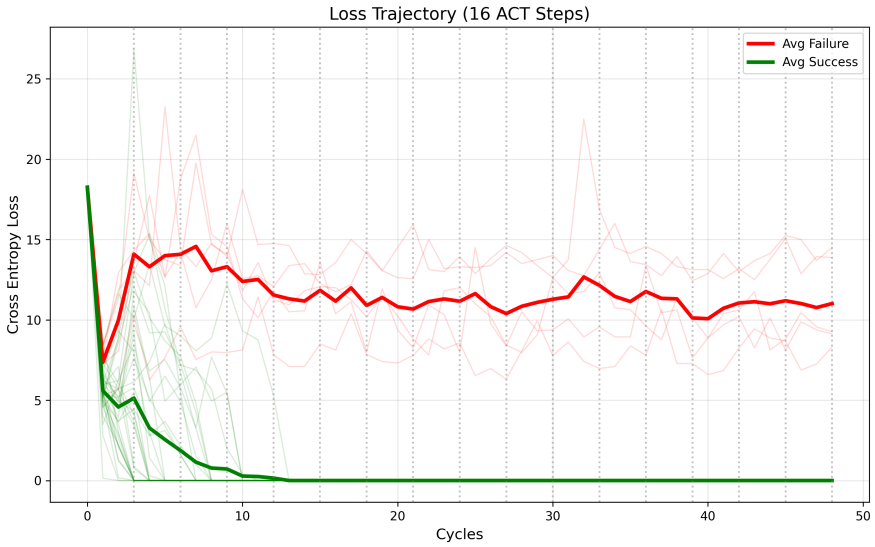

Figure 4: *Loss trajectories during recursive inference.* Green curves correspond to successful runs and red curves to failed runs; thick lines show averages over runs. Vertical dashed lines mark step boundaries (three update cycles each). Successful runs (green) continuously minimize cross-entropy loss until reaching zero, while failed runs (red) rapidly plateau at stable high-loss values around 11-12, indicating convergence to suboptimal solutions.

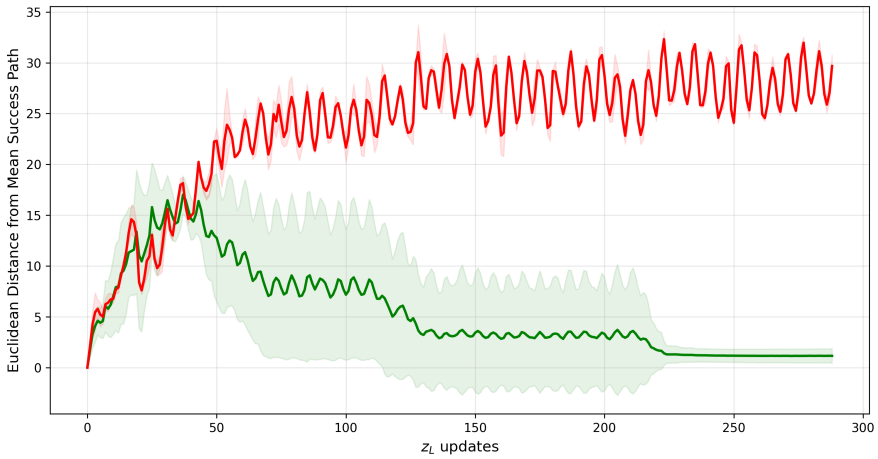

Figure 5: *Divergence from mean success path.* Red curve shows the mean Euclidean distance in latent space between failed attempts and the centroid of all successful trajectories at each z_l update step. The green curve is indicative of the solved puzzles and provides a control. Shaded regions indicate standard deviation. The mean success trajectory is calculated as the per-step centroid of the internal activation vectors across all puzzles that the model successfully solves. Shaded regions indicate standard deviation. Failed runs diverge sharply from the success manifold within the first 50 updates and continue diverging thereafter, while successful runs remain clustered around the mean success path. The persistent divergence of failed runs, which never reconverge to the success manifold despite hundreds of additional z_l updates, provides further evidence that reasoning failures stem from initialization-dependent basins of attraction rather than insufficient recursive depth.